

NEURAL CORRELATES OF ADAPTIVE BEHAVIOR:
STRUCTURE, DYNAMICS, AND INFORMATION PROCESSING

Steven C. Williams

Submitted to the faculty of the University Graduate School
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy
in the Cognitive Science Program
and the School of Informatics, Computing, and Engineering,
Indiana University

October 2019

Accepted by the Graduate Faculty, Indiana University,
in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Doctoral Committee

Chair (Cognitive Science) Randall D. Beer, PhD

Chair (Informatics) Luis M. Rocha, PhD

John M. Beggs, PhD

Olaf Sporns, PhD

Larry S. Yaeger, MS

October 7, 2019

Copyright © 2019
Steven C. Williams

*To Kay,
for her patience and support*

Acknowledgments

Many thanks go to my committee members—Randy Beer, Luis Rocha, John Beggs, Olaf Sporns, and Larry Yaeger—for their guidance in the development of this dissertation. I would especially like to thank Larry, who served as my primary research advisor for many years, and without whom the model that this work is based on would not exist. My gratitude goes to Tim Menzies for introducing me to both the field of cognitive science and the academic research process. I am indebted to the kind and hard-working administrative staff at Indiana University, especially Beverly Diekhoff, Linda Hostetter, and Susan Palmer. Lastly, I would like to thank my parents for their decades of support, for setting an example of high personal character that I still strive to live up to, and for fostering a love of learning that has led to this work.

This research was supported in part by National Academies Keck Futures Initiative grant number CS22 and by National Science Foundation Integrative Graduate Education and Research Traineeship award number 0903495. Much of the analysis described here was carried out on Indiana University’s high-performance parallel computing environment Big Red II, which was supported in part by Lilly Endowment, Inc., through its support for the Indiana University Pervasive Technology Institute, and in part by the Indiana METACyt Initiative. The Indiana METACyt Initiative at Indiana University was also supported in part by Lilly Endowment, Inc.

Steven C. Williams

NEURAL CORRELATES OF ADAPTIVE BEHAVIOR:
STRUCTURE, DYNAMICS, AND INFORMATION PROCESSING

Traditionally, cognitive science has focused on symbol-manipulation tasks in isolated problem domains. Implicit in this approach is the identification of intelligence with high-level, algorithmic, and uniquely human abstract reasoning. Arguably more essential, however, is the coordinated perception and action required of even the simplest organisms, whose survival depends on reacting appropriately and in real time to the constantly changing conditions of a messy and often hostile environment. Embracing this latter view, some modern approaches to cognitive science identify intelligence with adaptive behavior: an intelligent organism is one whose actions support the survival of the individual or species.

In this dissertation, I seek to characterize the neural basis of intelligence by investigating the neural correlates of adaptive behavior. This work is performed in the context of Polyworld, an agent-based artificial life model. In Polyworld, a population of agents evolves in a simulated ecology. Each agent is endowed with a rudimentary sense of vision and is capable of a set of simple actions controlled by a neural network. The population's long-term survival depends on the evolution of adaptive behavior via rewiring of these networks over successive generations. I identify the neural mechanisms underlying adaptive behavior by investigating trends in neural properties, both over evolutionary time and across experimental conditions.

Applying the tools of graph theory, dynamical systems theory, and information theory, I analyze Polyworld's neural networks using metrics with demonstrated relevance in other studies of complex systems. Many properties of interest are amplified over evolutionary time: the emergence of a successful survival strategy is accompanied by trends toward small-world neural structure, critical neural dynamics, and effective neural information processing. These trends are statistically significant when compared to a neutrally evolving null model, suggesting correlations with adaptive behavior. More direct evidence for these properties' adaptive significance comes from correlations with environmental complexity: as experimental conditions become increasingly difficult, each property is further amplified, fostering the increasingly complex adaptations required for survival.

Chair (Cognitive Science) Randall D. Beer, PhD

Chair (Informatics) Luis M. Rocha, PhD

John M. Beggs, PhD

Olaf Sporns, PhD

Larry S. Yaeger, MS

Contents

1	Introduction	1
1.1	Justification	1
1.2	Organization	4
2	Background	5
2.1	Prehistory	5
2.2	Neural Networks	7
2.3	Artificial Life	10
2.4	Evolutionary Algorithms	15
2.5	Conclusion	19
3	Model	20
3.1	Overview	20
3.2	Physiology	22
3.3	Actions	23
3.4	Neural Model	24
3.5	Genetic Model	26
3.6	Prior Work	27
3.7	Conclusion	33
4	Experiments	35
4.1	Overview	35
4.2	High-Level Results	37

4.3	<i>In Vitro</i> Analysis	45
4.4	Conclusion	47
5	Analysis of Structure	48
5.1	Foundations	48
5.2	Methods	51
5.3	Results	53
5.4	Discussion	54
5.5	Conclusion	55
6	Analysis of Dynamics	57
6.1	Foundations	57
6.2	Methods	60
6.3	Results	62
6.4	Discussion	62
6.5	Conclusion	68
7	Analysis of Information Processing	70
7.1	Foundations	70
7.2	Methods	77
7.3	Results	81
7.4	Discussion	84
7.5	Conclusion	86
8	Synthesis	87
8.1	Introduction	87
8.2	Methods	88
8.3	Results	97
8.4	Conclusion	97
9	Conclusion	100

A	Worldfiles	102
A.1	Parameters	102
A.2	Legacy	122
A.3	Simple	123
A.4	Forage	125
A.5	Poison	126
B	Proof of Equation 3.9	127
	Bibliography	130
	Curriculum Vitae	

List of Figures

3.1	Polyworld	21
3.2	Typical neural network	25
3.3	Complexity vs. timestep (prior work)	29
3.4	Complexity vs. timestep (prior work: driven and passive)	31
3.5	Complexity vs. timestep (prior work: driven, passive, and C-max)	34
4.1	Typical Legacy simulations	38
4.2	Population vs. timestep (Legacy)	39
4.3	Food consumption rate and birth rate vs. timestep (Legacy)	40
4.4	Complexity vs. timestep (Legacy)	41
4.5	Evolutionary selection vs. timestep (Legacy)	43
4.6	Strength gene vs. timestep (Legacy)	45
4.7	<i>In vitro</i> complexity vs. <i>in vivo</i> complexity (Legacy)	47
5.1	Friendship network	49
5.2	Graphs for the Watts–Strogatz model	50
5.3	Efficiency vs. rewiring probability (Watts–Strogatz)	53
5.4	Local and global efficiency vs. timestep (Legacy)	54
5.5	Small-worldness vs. timestep (Legacy)	55
5.6	Typical neural networks (Legacy)	56
5.7	Small-worldness vs. synaptic weight and density (Legacy)	56
6.1	Time series for the logistic map	58

6.2	Numerical calculation of the maximal Lyapunov exponent	60
6.3	State space expansion vs. timestep (Legacy)	62
6.4	Typical bifurcation diagrams (early Legacy)	64
6.5	Typical bifurcation diagrams (middle Legacy)	65
6.6	Typical bifurcation diagrams (late Legacy)	66
6.7	Onset of criticality vs. timestep (Legacy)	69
6.8	State space expansion vs. onset of criticality (Legacy)	69
7.1	Information content vs. probability	71
7.2	Relationships among information-theoretic metrics	76
7.3	Causal relationships in a neural network	77
7.4	Active information storage	78
7.5	Apparent and complete transfer entropy	79
7.6	Collective transfer entropy	80
7.7	Active information storage vs. timestep (Legacy)	82
7.8	Apparent and complete transfer entropy vs. timestep (Legacy)	82
7.9	Collective transfer entropy vs. timestep (Legacy)	83
7.10	Positive and negative separable information vs. timestep (Legacy)	83
7.11	Differential entropy vs. timestep (Legacy)	85
7.12	Relationships between differential entropy and other metrics (Legacy)	85
7.13	Local separable information vs. timestep (Legacy)	86
8.1	Typical Simple, Forage, and Poison simulations	89
8.2	Small-worldness and state space expansion (Simple, Forage, and Poison)	98
8.3	Complexity and active information storage (Simple, Forage, and Poison)	98
8.4	Positive and negative separable information (Simple, Forage, and Poison)	99

Chapter 1

Introduction

What are the properties of neural networks that allow them to produce adaptive behavior? Less formally, what makes a brain smart—able to help an organism navigate a complex and potentially treacherous environment? This is a central question in cognitive science, having implications not only for the development of more advanced artificial intelligences, but also for the better understanding of real-world biological intelligence. In this dissertation, I synthesize multiple approaches to this question in an attempt to provide a comprehensive explanation of how adaptive behavior is produced by neural networks.

1.1 Justification

A number of decisions must be made in order to translate this question into a concrete research methodology. The following sections justify the particular decisions made in the development of this dissertation.

1.1.1 Why Adaptive Behavior?

In this work, adaptive behavior is defined as any action supporting the goal of survival for an individual or species. Why consider such goal-directed behavior to be an indication of intelligence? The traditional approach in cognitive science has instead typically focused on abstract reasoning, considering adaptive behavior to be a mere complication of some more fundamental computational process, or at most an unexplored substrate upon which abstract reasoning might be built. In the traditional view, intelligence is to be found in algorithms, whether implemented in a brain or on a computer: to be smart is to be adept at manipulating symbols. Behaving adaptively is simply a matter of transducing the proper symbolic

representations of the world over which such an algorithm may operate.

Difficulties with the traditional approach have led to a reversal of these ideas. Alternative approaches consider adaptive behavior to be the more fundamental concept in the study of intelligence. In such views, abstract reasoning is just another example—albeit an impressively complex one—of the capabilities of intelligent organisms to survive and adapt. Ultimately, computation is only useful to an individual or species insofar as it assists in producing goal-directed action.

This focus on adaptive behavior emphasizes that the brain is not a pure symbol processor at the seat of intelligence. Rather, it is only one part of an extended information-processing system that is embodied within an organism and embedded within an environment. Adaptive behavior is inherently situated: behavior is not adaptive per se, but with respect to its context. Intelligence must be understood as the product of an entire brain–body–environment system.

1.1.2 Why Artificial Neural Networks?

The types of studies required by this line of research are often not feasible in the natural world. Detailed investigation of brain anatomy and neural activity in biological organisms can be invasive and impractical. In contrast, artificial neural networks are easily implemented in computer models and may be instrumented in arbitrarily fine detail with comparatively little effort. Straightforwardly analogous to biological brains, such models provide a convenient formalism for the study of adaptive behavior.

1.1.3 Why Agent-Based Modeling?

The inherent situatedness of adaptive behavior requires modelers to carefully consider both the behavior itself and the context in which it unfolds. Agent-based modeling addresses this concern neatly by separating these two elements—behavior and context—into distinct components: agents (where goal-directed action is localized) and their environment (where this action occurs and is made sense of). Embodiment and embeddedness are placed front and center by such an approach.

1.1.4 Why Simulated Evolution?

Human-designed models are limited by the motives, abilities, and imaginations of their designers. If constructed haphazardly, a model may simply reflect the designer's biases rather than explaining the phenomenon

under investigation. Various techniques help to remediate this human bias by removing aspects of the model from the designer's explicit control. In evolutionary algorithms, these aspects are instead placed under the control of a process that mimics biological evolution, thereby incorporating the same mechanism by which adaptive behavior appears in the real world.

1.1.5 Why Polyworld?

The work described in this dissertation is performed in the context of Polyworld, an artificial life model that combines the elements described thus far: adaptive behavior, artificial neural networks, agent-based modeling, and simulated evolution. In Polyworld, a population of agents evolves in a simulated ecology. Each agent is endowed with a rudimentary sense of vision and is capable of a set of simple actions controlled by a neural network. The population's long-term survival depends on the evolution of adaptive behavior via rewiring of these networks over successive generations.

Polyworld is unique among related models for its biological verisimilitude: its design was driven by the desire for a true-to-life yet high-level model. As a result, it is sophisticated enough to produce interesting ecological dynamics and modestly complex neural networks, yet it is simple enough that simulation and analysis are not overly cumbersome. Crucially, Polyworld's neural networks are evolved via a genetic algorithm with no fitness function other than survival and reproduction. The resulting network topologies are therefore determined by natural selection alone, rather than being designed into the model.

1.1.6 Why Metrics?

While computer models of adaptive behavior can be visually compelling, scientific rigor requires measurable results. For this work to be more than eye candy, metrics must be applied in order to quantify adaptive behavior and the mechanisms that produce it. Prior studies using Polyworld have addressed this need by investigating evolutionary trends in quantitative properties of its neural networks, most notably their complexity. The framework introduced by these prior studies—and relied on in the present work—includes a null model that allows statistical significance testing of such trends.

1.1.7 Why Structure, Dynamics, and Information Processing?

The tools of graph theory, dynamical systems theory, and information theory provide three different lenses through which neural networks may be viewed. In this work, I use these tools to quantify the structure, dynamics, and information-processing capabilities of Polyworld’s networks using metrics with demonstrated relevance in other studies of complex systems. Through the use of these metrics, I investigate the properties typically exhibited by adaptive networks. I discuss evolutionary trends in the context of real-world observations and, as in prior work, assess these trends for statistical significance. The ultimate goal of this work is an explanation of how neural network structure, dynamics, and information processing combine to facilitate the production of adaptive behavior.

1.2 Organization

The rest of this dissertation is organized as follows. Chapter 2 provides theoretical background by reviewing neural networks, artificial life, and evolutionary algorithms. Each of these fields is discussed in some detail, and relevant prior studies that inform this dissertation’s work are highlighted. Chapter 3 introduces Polyworld—the physics and ecology of the model, the physiology of its agents, and its neural and genetic models. Qualitative and quantitative results from prior Polyworld-based studies are discussed, as is the null model that allows significance testing of evolutionary trends. Chapter 4 describes the experiments on which the bulk of this dissertation is based and presents some of their high-level results.

Chapters 5, 6, and 7 constitute the core of this work, describing the methods and results of the structural, dynamical, and information-processing analyses carried out on Polyworld’s evolved neural networks. I measure evolutionary trends in relevant neural properties using the tools of graph theory, dynamical systems theory, and information theory. As in prior work, I assess these trends for statistical significance. Additionally, I explore the individual-level basis for these population-level trends in an effort to determine how they are implemented. Chapter 8 synthesizes these findings by investigating how neural properties are correlated with behavioral adaptation to environments of varying complexity. Chapter 9 concludes this dissertation, summarizing its major contributions and suggesting directions for future work.

Chapter 2

Background

This chapter presents a historical narrative focused primarily on neural networks, artificial life, and evolutionary algorithms. The development of each of these fields is discussed, and enough context is provided to support the material that follows in the rest of this dissertation. For other reviews of these topics, see [Langton \(1989\)](#), [Cliff \(2003\)](#), and [Downing \(2006\)](#).

2.1 Prehistory

Throughout much of the 1970s and early 1980s, the field of artificial intelligence (AI) was dominated by symbolic approaches. This “good old-fashioned AI” attempted to model and explain cognition as a process of formal symbol manipulation. [Newell and Simon \(1976\)](#) were among the first to formalize this approach. They claimed that, in order for a system to exhibit human-like intelligence, it must be a physical symbol system—“a machine that produces through time an evolving collection of symbol structures” through “processes of creation, modification, reproduction and destruction.” They further stated that such a system solves problems “by generating and progressively modifying symbol structures until it produces a solution structure” (i.e., via heuristic search). Essentially, they hypothesized that cognition is computation.

This computationalist view was centered on two claims. First, cognition requires internal representations: as the mind cannot directly operate over real-world objects, it must instead make use of symbols that represent those objects. Second, cognition is algorithmic: the mind operates by following a set of discrete steps or rules for manipulating the symbols it contains. The problem for AI, then, was simply to bring together the right representations and the right rules in the right context.

Newell and Simon were by no means the first proponents of computationalism. A decade prior to their work, Chomsky (1965) proposed that a formal grammar accounts for the productivity of language: our ability to understand and create an unlimited set of sentences—despite our limited set of linguistic symbols and rules—depends on language having a syntax whose rules hold regardless of the content of the symbols. Fodor (1975) posited that thought itself is a formal linguistic process, carried out in a kind of “mentalese” having its own grammar. These ideas led to implementations such as Schank and Abelson’s (1977) conceptual dependency theory, a framework for deriving canonical representations of sentences from their natural language equivalents. E.g., the sentence “John told Mary that Bill was happy” would be represented as “John **MTRANS**(Bill **BE MENT.ST**(5)) to Mary.”

There is no doubt that computationalism succeeded in at least mimicking certain aspects of cognition. Consider the “computer therapist” ELIZA (Weizenbaum, 1966)—a program that loosely imitates a Rogerian psychotherapist using simple natural language processing—to see how a little symbol manipulation can go a long way. However, in its preoccupation with high-level, uniquely human tasks, computationalism failed to make good on its efforts to fully explain cognition, and it became the target of philosophical objections throughout the 1980s.

A notable example of this criticism is Searle’s (1980) Chinese room thought experiment, in which an English-speaking man converses in Chinese by consulting a rule book. Via purely syntactic symbol manipulation, he converts questions (in Chinese) to answers (in Chinese) based only on the instructions he has been given (in English) for correlating the “meaningless squiggles” of the prompts that he receives with the responses that he produces. I.e., the man implements a formal symbol processor that speaks Chinese, yet he understands nothing of what is being said.

While this scenario stretches the limits of intuition, it raises a valid point: computationalism has proposed a syntax for cognition but no semantics. The representations it invokes are abstract and arbitrary, bearing no systematic relationship to their referents. This symbol-grounding problem (Harnad, 1990) points out that computationalism gives no account of how internal representations are grounded in the external world, nor does it explain how real-world objects are transduced into symbols in the first place. Further, there is no empirical evidence for such processes in the brain.

There is also a kind of circular reasoning in the computationalist account. If intelligence is defined in terms of symbolic tasks—the ability to “prove theorems, make and execute plans, and summarize newspaper stories” (Nilsson, 2007)—then intelligent systems will necessarily be symbol systems. But this ignores

arguably more important aspects of cognition such as real-time, coordinated perception and action, where the need for symbolic processing is dubious.

Consider the task of visually guided reaching. My ability to, say, catch a ball does not require a representational or algorithmic explanation; it may be cashed out more simply in terms of coordinated dynamics among the ball, my hand, and my brain. Granted, a computational explanation could be envisioned: first I measure the positions of the ball and my hand, then I calculate the displacement between the two, then I issue motor commands to reduce this displacement, and so on. But this account would seem lacking, as it ignores the continuous, nonsequential nature of the process.

There are no discrete, identifiable steps in which one representation gets transformed into another. . . . Rather, input, internal activity, and output are all happening continuously and at the very same time, much as a radio is producing music at the very same time as its antenna is receiving signals. ([van Gelder, 1995](#))

Such a system cannot be computational, as it lacks both internal representations and an algorithm for manipulating them.

2.2 Neural Networks

As symbolic approaches to AI were stagnating, connectionism was beginning to gain favor. This paradigm focused on a unique form of computation carried out by sets of simple, interconnected processing units operating in parallel. Studies of this “parallel distributed processing” became widely popular thanks in part to the PDP Research Group ([Rumelhart et al., 1986b](#); [McClelland et al., 1986](#)), whose work laid the foundations for the modern field of artificial neural networks.

While this work helped to usher in a new era for AI, it was not truly novel: PDP represented the revival of a line of research extending back to the 1940s when [McCulloch and Pitts \(1943\)](#) used networks of binary-threshold units to calculate first-order logic functions. However, connectionism had been dealt a blow decades later when [Minsky and Papert \(1969\)](#) proved the limitations of the commonly studied single-layer perceptron ([Rosenblatt, 1962](#)). Despite the fact that these limitations were not observed in multilayer models, [Minsky and Papert](#)’s work effectively curbed neural network research for many years, with this “AI winter” lasting until connectionism’s revival in the 1980s.

2.2.1 Fundamentals

Inspired by the structure of biological brains, an artificial neural network consists of a set of processing units (analogous to neurons) and a pattern of connectivity among these units (analogous to synapses between pairs of neurons). The processing units themselves are relatively simple, each mapping an input (the *excitation* received from other units, the environment, or even the unit itself) to an output (the unit's *activation*), which is then propagated through the network based on its connectivity. The input–output mapping (the unit's *activation function*) is often a simple binary threshold or sigmoidal curve. However, more complicated relationships may be defined, allowing the unit to generate “spikes” of activity analogous to the action potentials observed in biological brains. Generally, certain sets of units are designated as the network's inputs and outputs, defining an interface through which the network may interact with its environment.

Details of neural network models vary widely. At one extreme, there are the complicated spiking neurons of [Hodgkin and Huxley \(1952\)](#). Modeled by four equations and dozens of parameters that correspond to actual biological quantities, these neurons can faithfully reproduce certain types of activity observed in real-world nervous systems. At the other extreme, there are the simple binary-thresholded, symmetrically-connected networks due to [Hopfield \(1982\)](#). These almost completely dispense with the biological metaphor, though they exhibit interesting computational properties such as content-addressable memory (i.e., convergence to a previously trained state when later provided with a portion of that state). Between these extremes, there are myriad compromises: models which can produce biologically plausible activity in a computationally efficient manner ([Izhikevich, 2003](#)), rate-coded models in which the states of processing units more accurately correspond to short-term action potential frequencies ([Fukai and Tanaka, 1997](#)), and so on.

2.2.2 Learning

In most neural network models, each connection may be assigned a weight. Analogous to synaptic efficacy in biological brains, this weight determines the degree of influence that the activation of one processing unit exerts on the excitation of another. Hand-specification of these weights is impossible in all but the simplest of cases. Instead, learning rules may be applied in which weights are modified based on the activity of the network or in response to performance on a task ([Hinton, 1992](#)). These learning rules are as varied as the models they operate over, and the selection of a particular rule depends on the context.

If the desired mapping of network inputs to outputs is known in advance, then *supervised learning* may

be used. The network is trained by iteratively presenting inputs, comparing the actual output to the desired output, and adjusting weights such that errors are reduced. A successfully trained network is capable of generalizing to novel inputs, though overtraining must be avoided. Typically, supervised learning involves gradient descent methods like backpropagation (Rumelhart et al., 1986a). These have been used to great effect, but they are biologically implausible and less straightforward to apply to deeply layered or recurrent networks (i.e., those in which connectivity cycles are permitted).

In *reinforcement learning*, there is no unitwise comparison between actual and desired outputs. Instead, training is based on a payoff (reward or punishment) that corresponds to the network's overall performance in response to a given input. A variety of algorithms exist for adjusting weights with the goal of payoff maximization (Sutton and Barto, 1998). This type of learning may be used as long as the appropriateness of each network output can be assessed, even if a detailed input–output mapping is not known.

In *unsupervised learning*, no external feedback of any kind is provided. Instead, weights are adjusted based on the activity of the network itself, usually as a result some process involving competition or correlation among processing units. Canonical examples include Hebbian learning (Hebb, 1949) and the related spike-timing-dependent plasticity (Abbott and Nelson, 2000). Unsupervised learning is often associated with statistical methods involving transformation of inputs into a more efficient form (e.g., dimensionality reduction techniques such as principal components analysis).

2.2.3 Representation

Philosophically, connectionism offers a radically different account of cognition from that of computationalism. In symbol systems, knowledge is stored in collections of monolithic, static tokens. In neural networks, knowledge is stored in a matrix of connection weights and “represented” in a distributed fashion—distributed across both space and time (i.e., implemented as patterns of activation involving many processing units) (Hinton et al., 1986). This many-to-many relationship between processing units and representations offers advantages such as damage resistance and graceful degradation: localized destruction results in gradual performance decline rather than catastrophic failure. These features are difficult to explain in computationalist frameworks. (Compare the results of removing a processing unit from a neural network to those of deleting a line of code from a computer program.)

It is not merely the *structure* of connectionist representations that is unique, but also the *nature* of the represented content. While computationalist symbols are abstract, arbitrary, and ungrounded, connectionist

representations are modal and analogical—grounded in the perceptions (i.e., input patterns) that originally produced them. In the view proposed by Barsalou’s (1999) framework of perceptual symbol systems, when I think of a chair, my mind simulates what it is like to perceive a chair in modality-specific ways. As opposed to an amodal symbol that is merely *associated* with the object it describes, a perceptual symbol is inherently *connected* to its referent, capturing (in this case) aspects of what it is like to actually see, feel, and sit in a chair. Such representations could be employed in much the same way as amodal symbols, but given their analogical nature, their manipulation could be circumscribed by meaning-respecting constraints.

Whether patterns of activation can truly be considered representations (in the sense of computational symbols) is the subject of much debate. Connectionism seems to locate itself somewhere between traditional symbolic AI and more radical dynamical views which seek either to remove explanatory focus from internal representations or to dispense with them altogether (Clark, 1997). But the account summarized here provides a neat response to many of the philosophical issues encountered in computationalism.

2.2.4 Caveats

The connectionist approach is not without its disadvantages. The knowledge stored in a neural network—being a matrix of connection weights—is essentially opaque to a human observer. Intuitions about a network’s behavior are thus vulnerable to a teleological bias: the observer may assign purpose or meaning where there is none. Before making any claims, careful analysis is needed to determine how and why a given model behaves the way it does.

2.3 Artificial Life

The inception of the field of artificial neural networks in the 1940s was part of a broader scientific movement, a period of unparalleled interdisciplinarity that took place during and after World War II. Cybernetics, as this movement came to be called, was “the scientific study of control and communication in the animal and the machine” (Wiener, 1948). Its primary objects of study were regulatory systems that made use of feedback loops in the implementation of goal-directed behavior.

Cybernetics included much work on the modeling and simulation of lifelike behavior, and was thus a precursor to the modern field of artificial life (ALife). Some of these models were implemented in the physical world, like Walter’s (1950, 1951) robotic “tortoises” whose behavior gave “an eerie impression

of purposefulness, independence and spontaneity.” Some were implemented in virtual worlds, like [von Neumann’s \(1966\)](#) self-reproducing automata.

This latter model was one of the first applications of the cellular automaton, a lattice of finite state machines where the inputs to each machine are given by the states of machines at neighboring lattice points ([Ulam, 1962](#)). Seeking to abstract the logical form of self-reproduction from its natural implementations, [von Neumann](#) developed a 29-state cellular automaton that was capable of universal construction: given the description of any automaton, [von Neumann’s](#) model could replicate it in an adjacent portion of the lattice. When given a description of itself, it would construct an identical “offspring” automaton and, with the help of a description copier, provide this clone with its own copy of the description, thus enabling further self-reproduction.

This synthetic approach—the development of models as constructive proofs using “bottom-up, parallel, local-determination of behavior” ([Langton, 1989](#))—has become one of the hallmarks of ALife. While starkly different from a more standard analytic approach that focuses on reductive investigation of actual living systems, ALife still has the potential to inform our understanding of real-world biology. E.g., central to [von Neumann’s](#) model was the realization that descriptions must be used in two ways: as interpreted instructions in the automaton construction process, and as uninterpreted data in the description copying process. This mirrors the two functions of nucleic acids in the processes of transcription, translation, and replication—mechanisms which were not fully understood until after [von Neumann’s](#) death.

Work on cellular automata continued in the ensuing decades, with Conway’s Game of Life ([Gardner, 1970, 1971](#)) extending interest in the field beyond academia. Later studies provided detailed analyses of the space of possible one- and two-dimensional cellular automata ([Wolfram, 1983](#); [Packard and Wolfram, 1985](#)). But as was the case with neural networks, research into the modeling of fundamental lifelike behavior saw less activity throughout the mid- to late-20th century.

2.3.1 Philosophy

As with connectionism, ALife’s development was in part the product of shifting philosophies. By the mid-1980s, it had become apparent that “designing agents that can interact with the real world with the versatility and robustness of even simple animals has turned out to be considerably more subtle and difficult than originally realized, and approaches developed for disembodied agents have not in general translated well to the noisy, unpredictable, open-ended and dynamic nature of the real world” ([Beer, 1995](#)). After

decades of focusing on high-level symbolic tasks, AI began to embrace “insect level behavior as a noble goal” with Brooks’s (1985, 1986) pair of seminal papers introducing behavior-based robotics. Traditional sense–plan–act paradigms were to be replaced with layered architectures wherein each layer (responsible for some behavior) had access to the system’s inputs and outputs and to lower-level competencies. These “subsumption architectures” were successfully used in the development of robots that exhibited robust behavior in dynamic environments. Whether it was implemented in software (Beer, 1990) or hardware (Brooks, 1991), intelligence was coming to be interpreted much more inclusively as adaptive behavior broadly construed.

Another philosophical shift that informed ALife’s development was the conceptual extension of the mind beyond its traditional boundary of the brain to include aspects of the organism’s body and of its interactions with the environment. Rooted in the concepts of embodiment (Clark, 1996) and situatedness (Gibson, 1979), this extended cognition (Clark and Chalmers, 1998) emphasized that intelligent activity never occurs in isolation. Cognition is not a product solely of the brain, but rather of the entire brain–body–environment system (Beer, 1997).

2.3.2 Fundamentals

ALife came to prominence with the organization of its inaugural conference in 1987. In the proceedings of that conference, Langton (1989) defined the field as follows:

Artificial Life is the study of man-made systems that exhibit behaviors characteristic of natural living systems. It complements the traditional biological sciences concerned with the *analysis* of living organisms by attempting to *synthesize* life-like behaviors within computer and other artificial media. By extending the empirical foundation upon which biology is based *beyond* the carbon-chain life that has evolved on Earth, Artificial Life can contribute to theoretical biology by locating *life-as-we-know-it* within the larger picture of *life-as-it-could-be*.

ALife thus looks to exploit the “law of uphill analysis and downhill invention” (Braitenberg, 1984). A synthetic approach is, first of all, more practical than an analytic one: it is much easier to create a mechanism that exhibits lifelike behavior than it is to start from the outside and guess what that mechanism is. There is also a theoretical advantage to such an approach: the study of biological life is necessarily limited to those organisms we happen to be able to observe. ALife, in contrast, can investigate the principles of life divorced from the peculiarities of the particular instantiations that we see around us.

Distinctions may be made between the *soft*, *hard*, and *wet* approaches to ALife (Bedau, 2007), depending on whether models are implemented in software, hardware (i.e., robots), or biochemistry. This dissertation

focuses on soft approaches, which often employ agent-based software simulations. In these models, agents interact with their environment (which may include other agents) via some suite of sensors (which transfer information from the environment to the agent) and effectors (which give the agent some degree of control over its body and surroundings). During the course of a simulation, agents may live, die, and be replaced with new agents.

Usually the agent population is tasked with some objective, which may be implicit in the definition of the simulation or environment. This often involves competition for resources needed in order to survive or for opportunities to reproduce. Depending on the difficulty of the task, the environment may be defined so as to impede or to encourage success in this objective. Generally, the goal of a particular study is to devise a simulation such that the population succeeds, then to analyze and explain how it did so.

2.3.3 Adaptation

The success of a simulation is often predicated on the ability of the agent population to adapt to its environment or to the objective it has been tasked with. Such adaptation can occur on multiple levels. Not all of these levels are present in all ALife models, and even when they are, they are not always easily distinguishable from one another.

Phylogenetic adaptation requires that certain properties of each agent are specified (either directly or indirectly) in the agent's genome. When a new agent is introduced into the population, its genome is generated by some process that takes as its input the genomes currently present in the population. This process is often an evolutionary algorithm, though other kinds of search may also be used. The properties that are encoded in the genome are thus inherited (possibly with alterations) from previous generations of agents. More will be said about this type of adaptation in Section 2.4.

Ontogenetic adaptation involves a nontrivial process of translation between an agent's genotype (full hereditary information available in the genome) and its phenotype (observable characteristics). When this type of adaptation is used, the genome does not directly specify the characteristics of an agent. Rather, it acts as a kind of recipe for the agent's growth.

Epigenetic adaptation requires an agent to be capable of modifying its own behavior. I.e., it must be able to learn from experience. Such adaptations are typically not inherited by future generations in Lamarckian fashion. However, learning may still aid populations in an evolutionary context by allowing agents to explore their local fitness landscape (Hinton and Nowlan, 1987).

2.3.4 Examples

Given the open-endedness of “life-as-it-could-be,” the explosion of early ALife models is perhaps unsurprising, though there is much similarity among them. Most are simple ecologies implemented in one- or two-dimensional environments where the main task is the accumulation of resources necessary for reproduction. Models such as Evolve (Conrad and Pattee, 1970), Bugs (Packard, 1989), and Echo (Holland, 1994) all fall into this category, with experiments and subsequent analysis focusing primarily on evolutionary dynamics.

A variation on this theme is Ray’s (1991) Tierra, which demonstrates particularly well the utility of coevolution in ALife models. In Tierra, the environment is a virtual computer with a limited instruction set which acts on a chunk of memory (the abiotic “soup”) populated by “creatures.” These agents—represented by write-protected blocks of memory containing programs in the virtual computer’s machine code—compete for execution time with the ultimate goal of propagating their “species” to future generations by copying themselves into freshly allocated memory blocks.

When the Tierran environment is inoculated with a single self-replicating ancestral agent, its descendants rapidly fill the soup, with subsequent competition leading to reciprocal innovations (i.e., mutations) resembling parasitism, immunity to parasites, hyperparasitism, and social behavior. Such coevolutionary “arms races” (Dawkins and Krebs, 1979) are typical in models where competing agents evolve via *natural selection* (i.e., those in which evolution is driven solely by the ability to survive and reproduce). In such simulations, an agent’s fitness is a function of its environment, which includes the rest of the population. Agents thus continuously and reciprocally shape each others’ fitness landscapes.

Artificial selection, in contrast, implies an externally imposed fitness function based on criteria identified as relevant by a human observer. Significant difficulty is often encountered in defining such a fitness function: too harsh and it may squash promising but as yet ineffective variations; too forgiving and it may fail to provide sufficient selection pressure. Natural selection provides a kind of automatically sliding difficulty scale for a population and can act over criteria which affect survival and reproduction but are not readily observable. Coevolution thus has the capacity to encourage continuing innovation, not simply adaptation toward some optimum. However, care must be taken to avoid thinking of this innovation as a kind of global progress (McShea, 1991). Evolutionary dynamics are not guaranteed to be monotonic, and descendant forms are therefore not necessarily superior to ancestral ones.

Perhaps one of the most compelling of the early ALife models is [Sims's \(1994a, 1994b\)](#) virtual creatures. By evolving morphologies in concert with the neural systems controlling them, [Sims](#) produces agents which engage in various forms of goal-directed behavior, from simple locomotion to more complex tasks such as following a light source or competing for control of an object. The resulting strategies range from the uncannily natural to the bizarrely alien, emphasizing the ability of evolution to find solutions unaffected by the biases of human expectation and unfettered by the limits of human imagination.

Another central concept in ALife is that of emergence ([Chalmers, 2006](#)), the spontaneous arising of large-scale patterns from low-level mechanisms. A quintessential example is [Reynolds's \(1987\)](#) Boids, in which agents seek to stay close to their neighbors, matching their velocities to agents in the immediate area while simultaneously trying to avoid collisions. Despite the simplicity of this individual-level behavior, complex population-level flocking patterns emerge. Such large-scale complexity is rarely deducible from the underlying mechanisms. ALife's bottom-up focus, however, facilitates the implementation and analysis of such behavior.

2.3.5 Caveats

Given that a computer can in theory simulate anything, working simulations that reproduce lifelike behavior cannot necessarily be taken as providing evidence for or against theories of life.

In particular, whether we use natural or artificial environments, we must allow only universal physical laws and the theory of natural selection to restrict the evolution of artificial life. This means that simulations that are dependent on ad hoc and special-purpose rules and constraints for their mimicry cannot be used to support theories of life. ([Pattee, 1989](#))

“Theory-free simulations” that ignore biology run the risk of turning ALife into a “fact-free science” ([Horgan, 1995](#)). While high-quality simulations and graphics can be equal parts compelling and distracting, the trade-off is nearly limitless flexibility in experimental design and the ability to measure almost any phenomenon of interest.

2.4 Evolutionary Algorithms

Evolutionary algorithms are a family of metaheuristic optimization algorithms inspired by biological evolution ([Holland, 1975](#); [Goldberg, 1989](#)). They implement an approach to problem-solving based on iterative optimization of a group of candidate solutions.

The most popular variation of this paradigm is the genetic algorithm, which makes heavy use of biological metaphor and is routinely employed in ALife models. Each candidate solution represents a single *individual* in the *population* of all solutions under consideration. Characteristics of the solution are encoded in the individual's *genome*, with individual characteristics represented by separate *genes*. Variants of a particular characteristic constitute the corresponding gene's *alleles*. Solutions are optimized over successive *generations* using operators such as *crossover* and *mutation*.

2.4.1 Fundamentals

The basic implementation of a genetic algorithm proceeds as follows:

1. Initialize the first generation of the population either randomly or by seeding individuals in selected areas of the search space.
2. Evaluate the fitness of each individual. This implies the existence of a fitness function and therefore the use of artificial selection to distribute reproductive opportunities based on externally imposed criteria. Note that, in many ALife models, reproduction is built into the simulation, allowing implicit fitness evaluation and natural selection to take place. See the notes on coevolution in Section 2.3.4.
3. Select individuals to act as parents. This selection should be carried out differentially according to fitness (so that fitter individuals are more likely to reproduce) yet stochastically (so that the population does not prematurely converge).
4. Breed children from the parents selected in the previous step. Common genetic operators used in this reproductive process include crossover (selecting genes from one or the other parent in order to combine their characteristics) and mutation (altering inherited genes randomly in order to introduce genetic diversity into the population).
5. Construct the next generation by replacing low-fitness individuals or the entire population with the children bred in the previous step. Optionally, implement elitist selection by preserving some proportion of the fittest individuals from the previous generation without alteration.
6. Iterate until reaching some termination criterion (e.g., number of generations, fitness threshold, search stagnation, etc.).

Genetic algorithms are similar to other stochastic search algorithms like simulated annealing (Kirkpatrick et al., 1983). They amount to a parallel generate-and-test search of a solution space, where each individual in the population represents a separate point of search. An individual's genome (consisting of n genes) may be thought of as a particular location in an n -dimensional fitness landscape. A properly configured genetic algorithm should result in a population that converges to this landscape's global optimum. No assumptions are made about the shape of the landscape, meaning that genetic algorithms can be used in situations where simpler approaches such as hill-climbing struggle to escape local optima. This wide applicability can come at a cost, though: genetic algorithms may be inefficient in cases where there is exploitable structure in the problem domain.

2.4.2 Tuning

The performance of a genetic algorithm is sensitive to the many choices that must be made in its implementation. The following considerations demonstrate that much careful tuning is required in order to ensure a successful and efficient search.

How should characteristics be encoded in the genome? Bit strings or arrays of floating-point numbers are common choices. Direct encoding of characteristics is often the most straightforward technique, but in cases where the phenotype calls for repetition, the lack of a developmental process for genotype–phenotype mapping can lead to an explosion in genome length. This problem can be partially addressed by allowing genome length to vary among individuals, but this stopgap creates its own complications, as finding appropriate crossover points for arbitrary pairs of genomes becomes nontrivial.

How should the genome be laid out? Genes that are closer to one another are less likely to be split during crossover, though this is sensitive to the type of crossover used. Single-point, multiple-point, and uniform crossover (where each gene has an equal probability of being selected from either parent) are all options, as are more exotic possibilities in which more than two parents are used in breeding children.

How should the rate and magnitude of mutation be set? Overly aggressive mutation leads to loss of good solutions (unless elitist selection is used) and, in the extreme case, causes degeneration to random search. Insufficient mutation fails to maintain diversity in the population, leading to premature convergence.

2.4.3 Neuroevolution

Neuroevolution is a type of evolutionary algorithm in which the solutions being evolved are neural networks. Given its generic nature, neuroevolution is more widely applicable than the learning rules discussed in Section 2.2.2. The following examples indicate some of the variety in potential applications. As with genetic algorithms, much care must be taken in designing and tuning a search.

In one of the earliest examples of neuroevolution, [Montana and Davis \(1989\)](#) used a genetic algorithm to learn weights for fixed-topology (fully connected) networks tested on a classification task. Crossover was constrained to occur between sets of weights. I.e., input weights for each neuron were treated as atomic units in the breeding process. This method outperformed an early version of backpropagation.

[Kitano \(1990\)](#) addressed scalability issues in this approach using an indirect genotype–phenotype mapping: network topologies were encoded in the genome using a context-free grammar. Weights for these topologies were initially randomized, then updated via backpropagation. Crossover was completely unrestricted, and the mutation rate was modified on a per-child basis, varying in inverse proportion to the genetic similarity of the parents. Experiments indicated improvements in performance and scalability over direct encoding techniques.

[Ackley and Littman \(1991\)](#) used paired networks to achieve a kind of reinforcement learning for agents in an ALife simulation. An action network (responsible for the agent’s behavior) mapped sensory input to behavioral output. An evaluation network (responsible for the agent’s “goals”) mapped sensory input to a reinforcement signal. Initial weights for both networks were encoded in the genome, but weights for the action network were updated during an agent’s lifetime using a reinforcement learning algorithm based on the evaluation network’s output. The use of both evolution and learning was found to be more successful than either technique alone in producing adaptive behavior.

A common problem in neuroevolution is that of competing conventions: a given functional unit (i.e., a portion of the network responsible for certain patterns of activity) can appear in different areas of the network for different individuals. Two functionally equivalent networks may thus be structurally quite dissimilar. Consider two parent networks—both containing a particular functional unit—where the unit is encoded at the start of one parent’s genome and at the end of the other’s. Crossover between these two genomes can lead to redundancy (when the unit is selected from both parents) or to loss of functionality (when it is selected from neither). A variety of techniques have been developed for dealing with this problem.

Moriarty and Miikkulainen's (1997) symbiotic adaptive neuroevolution (SANE) skirts the competing conventions problem by evolving single neurons (rather than entire networks) in a constrained topology. Evolved neurons are placed in a hidden layer between inputs and outputs, with the connections to these other layers (both their topology and weights) specified genetically. Neurons are evaluated based on their ability to cooperate within groups selected randomly from the population of neurons being evolved. Experiments show SANE to be an efficient and adaptive technique, in addition to maintaining a high level of diversity in the population.

Stanley and Miikkulainen's (2002) neuroevolution of augmenting topologies (NEAT) addresses the competing conventions problem more directly. NEAT evolves variable-length genomes that encode both topology and weights. Networks are incrementally grown from minimal structure, resulting in gradual complexification and tending to produce simpler solutions: genomes are only likely to increase in length when the additional complexity confers an evolutionary advantage on the corresponding networks. During crossover, these variable-length genomes are aligned based on historical markers that are associated with each gene at the time it is appended to the genome. This simple but effective technique can appropriately identify homologous genes as potential crossover points. NEAT outperforms several other methods of neuroevolution when tested on a pole-balancing task.

2.5 Conclusion

This chapter has introduced three fields—neural networks, artificial life, and evolutionary algorithms—that constitute the foundation of the work presented in this dissertation. With this background in place, the specifics of this dissertation's work may now be discussed, beginning with the model on which it is based.

Chapter 3

Model

This chapter introduces Polyworld and discusses prior work which is based on it. The official Polyworld source code repository is available at <https://github.com/polyworld/polyworld>. This dissertation is based on a fork of that repository available at <https://github.com/smailliwcs/polyworld>. The simulations described in this work make use of Git revisions [bdc7060](#) and [9057d6c](#), which have since been merged into the official repository. Polyworld was originally reported in the proceedings of the third Artificial Life conference (Yaeger, 1994).

3.1 Overview

As shown in Figure 3.1, Polyworld’s ecology consists of trapezoidal agents and rectangular food items on a flat ground plane that may be partitioned into separate regions by vertical barriers. Each agent is controlled by a neural network. The inputs to this network are mainly visual: at each timestep, the world is rendered from the agent’s point of view (as shown in the small boxes—one for each agent—at the bottom of Figure 3.1). The resulting pixels are passed to the network as if they were light falling on a retina. The network’s outputs control the agent’s actions, such as eating, mating, and moving.

The basic task for Polyworld agents involves navigating through the environment in response to visual input via the Move and Turn actions. Agents must forage for food and consume it by coming into contact with it while expressing the Eat action. Due to the finite lifespan of individual agents, long-term survival of the population is only possible through reproduction, which is initiated by two collocated agents simultaneously expressing the Mate action.

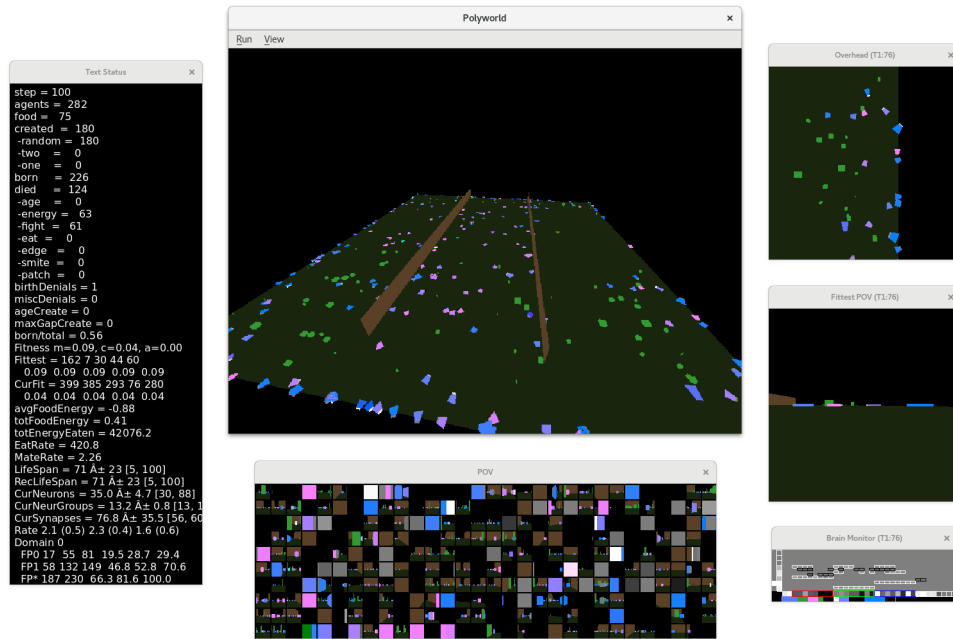


Figure 3.1: Polyworld running on Debian GNU/Linux. Left: Basic statistics for the simulation in progress. Center: Overhead view of the world (top) and points of view for each agent (bottom). Right: Overhead view (top), point of view (middle), and neural network (bottom) for a single agent.

An agent's genome specifies characteristics of its physiology and neural structure. When two agents mate, an offspring agent is produced whose genome is generated from the parent genomes subject to crossover and mutation. Agents evolve under the influence of natural selection alone; in its standard mode of operation, Polyworld has no fitness function other than survival and reproduction.

Agents expend energy with all actions. Even when completely inactive, an agent's energy is subject to a fixed nominal drain. Energy is also lost by coming into contact with another agent that is expressing the Fight action. When its energy is exhausted or when it reaches a maximum lifespan, the agent dies, potentially leaving behind a food item representing its dead carcass. Food also grows randomly in localized patches, providing a naturally occurring source of energy.

Polyworld thus amounts to a complex energy-balancing problem where agents must maximize consumption and retention of energy while minimizing its expenditure. In general, there is significant difficulty associated with designing a simulation such that agents succeed in solving this problem without overpopulating the environment, and so Polyworld provides a number of ways to facilitate energy balance. Many of these are population control mechanisms in which individual agents' energy expenditure increases or decreases in response to the population-wide agent count. Polyworld also incorporates a more traditional

genetic algorithm in which the population can be artificially maintained at or above some threshold by forcing births from parents selected according to an ad hoc fitness function. This latter mechanism is undesirable, however, as it introduces an element of artificial selection, and so it has been disabled for the experiments discussed in this work.

Many aspects of Polyworld simulations are controlled by *worldfiles* which specify population limits, food growth parameters, barrier positions, and so on. The list of available parameters is extensive, as Polyworld supports a wide variety of experiments. For the sake of simplicity, this chapter ignores some aspects of Polyworld that are not relevant to the work presented in this dissertation. For a full description of available worldfile parameters and the particular settings used in this work, see Appendix A. The bulk of the results reported here refer to experiments conducted using the [Legacy](#) worldfile.

At the beginning of a simulation, Polyworld is seeded with an agent population that is simple, genetically uniform, and relatively poorly adapted to the environment. This population is not viable and must evolve in order to avoid dying out. There is thus significant selection pressure on the agents' genomes, at least until the population becomes capable of sustaining itself. Populations generally adapt behaviorally to the [Legacy](#) environment within the first several thousand timesteps. After this point, a typical agent exhibits evolutionarily adaptive behaviors such as foraging for food and seeking mates for reproduction.

3.2 Physiology

Certain physiological characteristics can vary among Polyworld agents. Most of these are genetically specified and thus under direct evolutionary control. Others are controlled by the agent itself.

The *Size* gene affects the physical size of an agent, its energy capacity, its energy expenditure due to moving and turning, and the damage it inflicts when fighting. The *Strength* gene also affects fight damage and, more notably, directly scales all energy usage. The *MaxSpeed* gene determines how quickly an agent can move and, like the *Size* gene, affects energy expenditure due to moving and turning. These genes introduce important tradeoffs in potential survival strategies: smaller, weaker, and slower agents can more easily conserve energy, while larger, stronger, and faster agents have a predatory advantage.

The *MateEnergyFraction* gene determines the proportion of an agent's energy that is transferred to its offspring at birth. Each agent actually stores two different classes of energy—*health* energy and *food* energy—each having the same capacity. At birth, both classes of energy are initialized to the same level

(as determined by the parents' health energy levels and their `MateEnergyFraction` genes). Both classes of energy are depleted by the agent's activity, and both are replenished by eating. However, only an agent's health energy is depleted by being attacked (i.e., by a collocated agent expressing the `Fight` action). At death, an agent's remaining food energy is normally converted to a food item, though this behavior is disabled in the `Legacy` condition in order to facilitate tracking and maintenance of the world's food supply. Though this eliminates the possibility of predation, the `Fight` action still provides a potentially useful way for agents to compete for resources.

The `ID` gene is mapped to the green color component of an agent's body. The other color components (blue and red) are mapped to the neural activations corresponding to the agent's `Mate` and `Fight` actions. An agent's shape is affected by the `MaxSpeed` gene, which determines the proportions of the agent's trapezoidal body: slow agents are short and wide, while fast agents are long and thin.

3.3 Actions

The suite of primitive actions available to Polyworld agents consists of eating, mating, fighting, moving, turning, lighting, and focusing. All actions expend energy based on the activation of the corresponding output neuron (representing the agent's volition toward the action) and a per-action scale factor specified in the worldfile (`EnergyUseEat`, `EnergyUseMate`, and so on). The `Eat`, `Mate`, and `Fight` actions all require volition to exceed some threshold before the action becomes effective.

The `Eat` action restores an agent's energy via consumption of a portion of a collocated food item. The amount of energy consumed is proportional to volition.

The `Mate` action initiates reproduction via crossover and mutation with a collocated agent who is also expressing the `Mate` action. Both parents are then prohibited from mating again until `MateWait` timesteps have elapsed. Mating may also be denied due to population limits or an insufficient energy level in either potential parent.

The `Fight` action reduces the health energy of a collocated agent. The amount of energy reduction depends on volition and on the attacker's health energy, size, and strength.

The `Move` action advances an agent along its current direction of travel by a distance proportional to volition. Motion is impeded by barriers and the edges of the world. The `Turn` action changes the orientation of this forward motion by an angle proportional to volition.

The `Light` action adjusts the brightness of the front surface of an agent’s body. The color of this surface is constrained to shades of gray, as all color components are mapped identically based on volition.

The `Focus` action allows an agent to control the angle of its visual field. Vision is implemented by first rendering the scene from the agent’s perspective using a horizontal field of view proportional to volition. A strip of pixels just above vertical center (i.e., directly on top of the ground plane) is then antialiased into the appropriate number of bins and provided to the visual input neurons. Red, green, and blue color channels are considered separately during this process, as each channel may have a different number of corresponding input neurons.

3.4 Neural Model

As shown in Figure 3.2, Polyworld’s neural networks are separated into five input groups, a variable number of internal groups, and seven output groups. Three of the input groups provide red, green, and blue Vision, with a variable number of neurons dedicated to each group. The other two input groups consist of single neurons: an Energy neuron (which tracks the agent’s current health energy level) and a Random neuron (which is activated randomly at each timestep). The seven output groups, each consisting of a single neuron, control the agent’s seven actions.

An agent’s genome specifies much of its neural structure, including the number of visual input neurons dedicated to each color, the number of internal groups, and the number of neurons within each internal group. Internal neurons are strictly excitatory or inhibitory: for a given neuron, outgoing connections have either all positive or all negative synaptic weights. Notably, individual synaptic weights are not specified genetically. Instead, connections between each pair of groups are described at a relatively high level in the genome using two parameters: connection density (the proportion of possible connections that are actually realized) and topological distortion (the degree to which connections are made randomly rather than sequentially). Synaptic weights are randomized before birth, then updated during an offline gestation period and throughout the agent’s lifetime via a Hebbian learning rule. Weights have a maximum magnitude w_{\max} that corresponds to the `MaxSynapseWeight` parameter. Since synapses may be excitatory or inhibitory, all weights are constrained to $[-w_{\max}, w_{\max}]$.

Normally, an agent incurs neural energy costs based on the number of neurons and synapses present in its network. However, such energy expenditure is disabled in the `Legacy` condition to avoid imposing

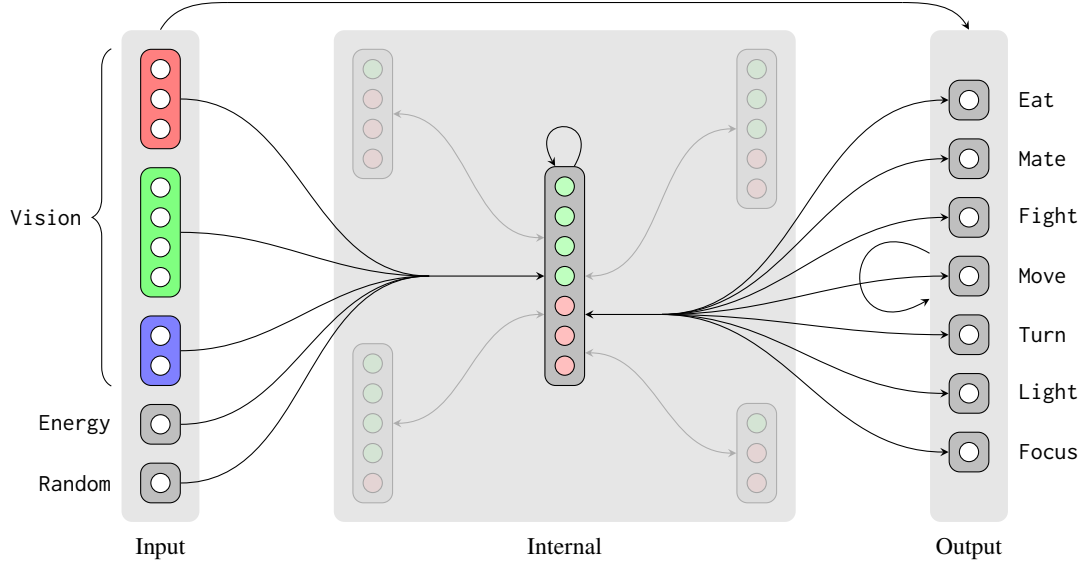


Figure 3.2: Typical neural network for a Polyworld agent. Input groups potentially project to internal and output groups. Internal and output groups potentially project to each other and recurrently to themselves. Only representative connections are shown; those involving internal groups other than the large central group are mostly suppressed.

physiological constraints on network topology.

Polyworld uses discrete-time, rate-coded neural networks with summing-and-squashing neurons. At each timestep t , the i th neuron's activation a_i is updated according to

$$a_i(t+1) = \sigma\left(\theta_i + \sum_{j=1}^n w_{ji}(t) a_j(t)\right), \quad (3.1)$$

where θ_i is the genetically specified bias of the group containing the i th neuron, n is the total number of neurons, w_{ji} is the synaptic weight from the j th neuron to the i th neuron, and σ is a squashing function. If there is no synapse from the j th neuron to the i th neuron, w_{ji} is assumed to be zero. It is often convenient to normalize synaptic weights by w_{\max} , allowing Equation 3.1 to be rewritten as

$$a_i(t+1) = \sigma\left(\theta_i + w_{\max} \sum_{j=1}^n \bar{w}_{ji}(t) a_j(t)\right), \quad (3.2)$$

where \bar{w}_{ji} represents a synaptic weight that has been normalized into $[-1, 1]$. The squashing function applied to neural excitation x is the logistic curve

$$\sigma(x) = \frac{1}{1 + e^{-\alpha x}}, \quad (3.3)$$

Meta-genetic	Physiological	Neural
<ul style="list-style-type: none"> • Mutation-Rate • Crossover-PointCount • LifeSpan 	<ul style="list-style-type: none"> • ID • Strength • Size • MaxSpeed • Mate-Energy-Fraction 	<ul style="list-style-type: none"> • Red, Green, and Blue visual input neuron count • InternalNeuronGroupCount • ExcitatoryNeuronCount and InhibitoryNeuronCount for each internal group • Bias for each group • ConnectionDensity, LearningRate, and TopologicalDistortion for each pair of groups

Table 3.1: Genes by type.

where α is the `LogisticSlope` parameter.

Throughout an agent’s lifetime, its synaptic weights are updated according to the Hebbian learning rule

$$w_{ji}(t + 1) = D\left(w_{ji}(t) + \eta_{ji} \left[a_j(t) - \frac{1}{2} \right] \left[a_i(t + 1) - \frac{1}{2} \right] \right), \quad (3.4)$$

where η_{ji} is the genetically specified learning rate of synapses connecting the groups containing the j th neuron and the i th neuron. The decay function D slowly weakens synapses whose weights have magnitudes greater than $\frac{1}{2}w_{\max}$:

$$D(w) = \begin{cases} w & \text{if } |w| \leq \frac{1}{2}w_{\max}, \\ w \left[1 - (1 - r) \frac{|w| - \frac{1}{2}w_{\max}}{\frac{1}{2}w_{\max}} \right] & \text{if } |w| > \frac{1}{2}w_{\max}, \end{cases} \quad (3.5)$$

where r is the `SynapseWeightDecayRate` parameter.

3.5 Genetic Model

Each gene in Polyworld is represented by a single byte whose value is interpolated into an applicable range, which is often controlled by worldfile parameters. E.g., the raw value of an agent’s `Size` gene is mapped from $[0, 255]$ into $[\text{MinAgentSize}, \text{MaxAgentSize}]$. A full list of genes is given in Table 3.1. In addition to the genes already discussed, this list includes three genes that provide a kind of meta-level genetics. They control the mutation rate and number of crossover points used during reproduction, as well as an agent’s maximum lifespan.

Genome length is fixed based on the maximum number of neural groups (controlled in part by the `MaxInternalNeuralGroups` parameter). This means that portions of an agent’s genome may be unused in the developmental process if the agent has fewer than the maximum number of neural groups. However,

this fixed-length scheme produces a smoother evolutionary landscape and provides a mechanism for neutral mutations to accumulate in the genome.

Previous versions of Polyworld evolved fixed-topology (fully recurrent) neural networks by explicitly encoding synaptic weights for all pairs of neurons in the genome. This resulted in a genome whose size scaled quadratically with the maximum number of neurons. While the group-based encoding described here results in far fewer neural genes, they still comprise the vast majority of the genome. To ensure recombination of non-neural genes, at least one crossover point is constrained to occur within the meta-genetic or physiological genes during reproduction.

Genomes for a simulation's initial population are seeded according to sensible defaults. E.g., agents are predisposed to mate at all times, to approach the color green (foraging for food), and to avoid the color red (fleeing attack). However, these seed genomes are of nearly minimal complexity in that they code for no internal neural groups. Furthermore, they are identical across all agents: the initial population of a [Legacy](#) simulation is completely genetically uniform.

3.6 Prior Work

A primary goal of many early Polyworld experiments was “the evolution of complex emergent behaviors from only the simple suite of primitive behaviors built into the organisms” ([Yaeger, 1994](#)). The environments used for these experiments produced a variety of successful survival strategies. Evolved agents could be qualitatively separated into different behavioral “species,” with the evolution of these species “mediated exclusively through the action of natural selection on the organisms’ nervous systems.”

Survival strategies were highly dependent on the specific parameters of the simulation. Wraparound (toroidal) worlds with plentiful food resulted in “frenetic joggers” that moved straight ahead continuously, always eating and mating—an effective strategy for such a simple environment. Worlds with impassable edges gave rise to “edge runners” that easily found mates by congregating near the world’s boundaries. Tabletop worlds—where contact with the world’s edges was fatal—produced “dervishes” that spun in tight circles, a strategy that allowed modest exploration of the local environment with minimal risk. In simulations where parents were not required to transfer energy to their offspring, “indolent cannibals” evolved to exploit this free source of energy by remaining essentially stationary and eating each other, circumventing the need to move about in search of food and mates.

More complex patterns of behavior were eventually evolved, including those that resembled fleeing, fighting back, grazing, foraging, following/chasing, and flocking/swarming. A theme throughout these early experiments was that “the gross energetics of the system have been observed to be crucial to the evolution of successful survival strategies” (Yaeger, 1994). The development of various energy balance mechanisms helped to facilitate such evolution.

Modern experiments using more complicated environments have shown that populations are capable of distributing themselves ideally with respect to resources (Griffith and Yaeger, 2006) and tracking a moving food patch. Other experiments have made use of recent modifications to Polyworld: new types of neural topologies and dynamics, additional sensors and actions, and more complex metabolisms. Videos of many of these experiments are available at <http://shinyverse.org/larryy/PolyworldMovies.html>.

3.6.1 Complexity

Much recent work on Polyworld has analyzed the information-theoretic properties of its evolved neural networks (Yaeger and Sporns, 2006; Yaeger et al., 2008; Yaeger, 2009). As shown in Figure 3.3, a trend toward increased complexity has been observed during the first several thousand timesteps of a simulation—roughly the period during which populations adapt behaviorally to the environment. This figure was generated by binning agents at 1000-timestep intervals based on their time of death. E.g., the leftmost data point in the figure represents the mean of all agents who died within the first 1000 timesteps. Note that similar figures in later chapters do not bin agents, instead taking the population-level value at a given timestep as the mean over all agents who were alive at that timestep.

The complexity metric used here is based on the neural complexity of Tononi et al. (1994), which measures the degree to which a system’s activity is globally integrated yet locally segregated. Neural complexity C_N is calculated using averages of multi-information M for k -subsets X_i^k of the system X :

$$C_N(X) = \sum_{k=1}^n \left[\frac{k}{n} M(X) - \langle M(X_i^k) \rangle_i \right], \quad (3.6)$$

where n denotes the system’s total size, and the subscript i indicates that the average is taken over all possible subsets of size k . Multi-information (also known as *total correlation*) is a generalization of mutual information to more than two variables, allowing quantification of the statistical dependency among an

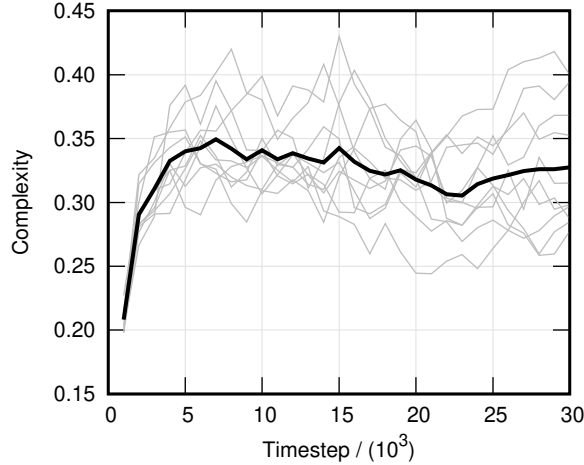


Figure 3.3: Complexity vs. timestep for ten simulations originally reported in [Yaeger et al. \(2008\)](#). Light lines represent population means for individual simulations, while the heavy line represents a meta-mean over all ten simulations.

arbitrary number of elements x_i :

$$M(X) = \sum_{x_i \in X} H(x_i) - H(X), \quad (3.7)$$

where H is the Shannon entropy.

A quantity that is related to neural complexity—but which does not require exhaustive subset analysis—is the interaction complexity C_I ([Tononi et al., 1998](#)), which measures the portion of a system’s entropy that is accounted for by interactions among its elements:

$$C_I(X) = H(X) - \sum_{x_i \in X} H(x_i | X \setminus \{x_i\}). \quad (3.8)$$

Note that by dividing the interaction complexity by the system’s size, we recover the penultimate term of neural complexity (i.e., the $k = n - 1$ term of the summation in Equation 3.6):

$$C_N^{n-1}(X) = \frac{1}{n} C_I(X). \quad (3.9)$$

Prior work has typically calculated complexity as in Equation 3.9 using rank-equivalent Gaussian noise matched to the time series of neural activations recorded during an agent’s lifetime. This is the quantity reported in Figure 3.3 and referred to simply as *complexity* throughout this dissertation. For a full proof of Equation 3.9, see Appendix B.

3.6.2 Driven and Passive

While Figure 3.3 indicates an evolutionary trend toward increased complexity, it is unclear whether this trend is actively selected for by evolution, or whether it is a consequence of the “left wall” of evolutionary simplicity (Gould, 1994). I.e., the initial population is so simple that it *must* complexify—there is no room for further simplification. Might the trend toward increased complexity be explained by random genetic drift away from a minimally complex starting point?

To investigate this question, a null model (passive) mode of simulation was developed to complement Polyworld’s standard (driven) mode (Yaeger et al., 2008). Here, *driven* and *passive* are used in the sense proposed by McShea (1994): the former indicates evolution by natural selection, while the latter refers to a random walk through genetic space. In driven mode, a standard simulation is run under the influence of natural selection, and the timing of each agent’s birth and death is recorded. In passive mode, a corresponding simulation is run in which evolutionary dynamics are identical to those of the driven simulation, yet stripped of adaptive significance. To accomplish this, natural births and deaths are suppressed in passive mode. Instead, these events occur randomly but in lockstep with the previously completed driven simulation. For every driven birth, two randomly selected passive agents are mated. For every driven death, a randomly selected passive agent is killed.

The passive simulation thus acts as a null model for the corresponding driven simulation, representing the intrinsic evolutionary dynamics of the model under neutral evolution. Differences between paired driven and passive simulations may be attributed to the action of natural selection: a driven trend is only adaptively significant if it outpaces its passive counterpart. A null model such as this “helps us to determine which aspects of the behavior of a [driven] run can be attributed to the adaptive significance of genotypes and which might reflect nothing more than the system’s underlying architecture or chance” (Bedau et al., 1998).

Predictably, agents evolved in passive simulations do not exhibit adaptive behavior. Since passive births and deaths occur randomly, fitter agents are not rewarded with increased reproductive opportunities as they are in driven simulations. In a passive simulation, an agent that forages for food and seeks mates for reproduction is no more likely to pass on its genes than any other agent. However, the evolutionary trend in driven/passive complexity is somewhat surprising. Figure 3.4 shows this trend, with statistical significance based on the p -value for a two-tailed, paired Student’s t -test of driven and passive means. Significance is calculated as $1 - p$, with large values (exceeding 0.95) indicating a significant difference ($p < 0.05$) between

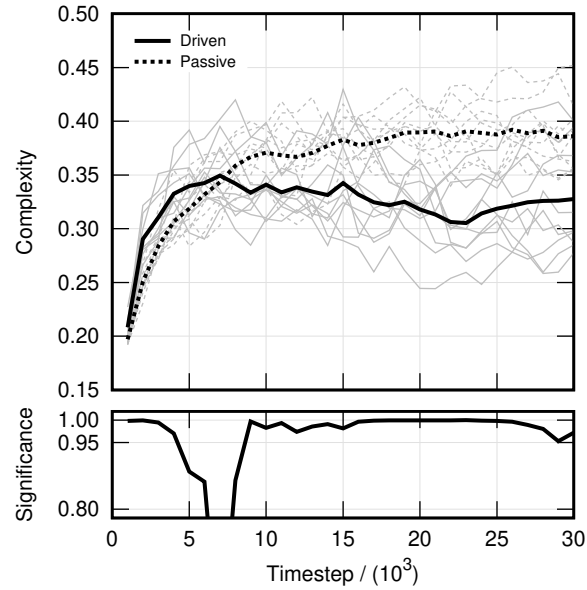


Figure 3.4: Complexity vs. timestep for ten driven/passive pairs of simulations originally reported in [Yaeger et al. \(2008\)](#). Light lines represent population means for individual simulations, while heavy lines represent meta-means over all ten simulations. Significance is calculated as $1 - p$ for a two-tailed, paired Student's t -test of driven and passive means.

the driven and passive simulations (i.e., a selection bias relative to neutral mutation).

Given its apparent association with adaptive behavior, we might expect complexity to be significantly higher in driven simulations. Figure 3.4 indicates that this is indeed the case during the first several thousand timesteps. After this period, however, driven complexity remains relatively constant—even decreasing in a number of cases—though a few simulations do produce further increases in complexity, as competition between agent subpopulations drives a kind of punctuated equilibrium ([Murdock and Yaeger, 2011](#)). Meanwhile, passive complexity exhibits unexpected continued growth for an additional 10,000 or so timesteps, ultimately attaining a much larger value, despite the fact that agents in these simulations never exhibit adaptive behavior.

[Yaeger et al. \(2008\)](#) explain these observations as follows (emphasis added):

The first thing to note is a statistically significant faster growth rate in complexity in the driven runs than in the passive runs during approximately the first 4000 time steps. *Evolution is clearly selecting for an increase in complexity during this early time period.* This makes sense intuitively since the seed population is known to be non-viable and must evolve or die out. Increases in complexity during this period are of a distinct evolutionary advantage, producing descendant populations that are more capable of thriving in this particular environment than their ancestors. During this time period, the evolution of complexity is clearly driven, with a bias towards increasing complexity.

The next thing to note is the early plateauing of complexity in the driven runs, allowing the randomly drifting complexity of the passive runs to catch and surpass them by around $t = 7000$. This is the result of evolution having found a solution that is “good enough”, and the concomitant spread of the genes producing this solution throughout the population. Seven out of the 10 natural selection simulations remain relatively stable around this modestly complex solution once it is found. *The intuition here is that any change away from this “good enough” solution is likely to be detrimental, hence evolution selects for stability.* Note that this actively suppresses genetic drift and, indeed, a statistically significant difference between driven and passive runs, with passive complexity now being the larger of the two, is maintained from about $t = 8000$ to the end of the runs at $t = 30,000$.

There is also a consistent, but less interesting, plateauing of complexity in the passive runs. *This is due solely to the individual bits of the underlying genome approaching a state of approximately 50% on, 50% off.* Effectively, the random walk has maximized variance as much as it can given the model parameters. Though generally higher than the driven mean, complexities in the passive/random model are nowhere near the maximum obtainable with the full range of gene values (as observed in the complexity-as-fitness function experiments discussed below); they just correspond to the range of complexities representable by the genome with an even mix of on and off bits. *Such larger values of complexity are potentially meaningful, but do not confer any evolutionary advantage on agents in these lockstep runs.*

In [Dennett’s \(1996\)](#) words, “the cheapest, least intensively designed system will be ‘discovered’ first by Mother Nature, and myopically selected.” Driven complexity thus stabilizes once the population becomes sufficiently well adapted to its environment. In contrast, passive complexity grows until reaching values typical of agents with random genomes. As discussed in [Section 3.6.3](#), complexity is not strictly tied to evolutionary fitness, thus it is unsurprising that this passive complexity growth is not accompanied by the production of adaptive behavior.

[McShea and Hordijk \(2013\)](#) provide further insight into the possible reasons for the observed driven complexity trend. In the Darwinian view, complexity evolves monotonically via a process of incremental addition: initially simple structures are steadily enhanced by the addition of new parts through natural selection, with each addition representing a marginal improvement to overall fitness. An alternative view of “complexity by subtraction” proposes that complexity grows early in evolution “perhaps on account of the spontaneous tendency for parts to differentiate” given the system’s initial simplicity. An overproliferation of structures results in a system that is functionally successful yet unstable and inefficient. A reduction in complexity follows, “perhaps driven by selection for effective and efficient function,” which requires simplification following the initially excessive complexity growth. This view provides a possible explanation for the decrease in complexity observed in the later stages of some driven simulations.

3.6.3 Complexity Maximization

The studies discussed thus far have been fairly typical in that they evolve behavior via natural selection to “solve” a particular environment, then analyze the resulting evolutionary trends in neural properties. The results of such studies can help to highlight the neural correlates of adaptive behavior—the properties of neural networks that support the implementation of successful survival strategies.

Polyworld also supports studies that invert this line of inquiry by evolving neural networks for specific properties, then analyzing the resulting evolutionary trends in behavior. Complexity-maximization (C-max) simulations are an example of this type of study which have seen frequent use in prior work. They represent a unique mode of operation—neither driven nor passive—in which complexity is taken as an objective function for optimization. Natural selection is dispensed with, and evolution is instead implemented via a steady-state (constant-population) genetic algorithm that takes complexity as its fitness function. Upon the death of any agent, it is replaced by the offspring of two parents selected from among the fittest (i.e., most complex) agents ever observed.

As shown in Figure 3.5, C-max simulations are capable of evolving agents with highly complex neural dynamics. Indeed, complexity values in these simulations far exceed those typically observed in driven and passive modes. Yet, as in passive simulations, typical patterns of behavior are not ostensibly adaptive. Agents evolved in C-max simulations appear to ignore both food and mates, instead exhibiting a tight turning behavior throughout much of the simulation. [Yaeger \(2009\)](#) proposes that “this behavior produces fairly high entropy while maintaining a significant degree of mutual information in the sensory units . . . thus producing a fairly high degree of complexity throughout the network.” In longer-running C-max simulations, agents begin to explore the environment more widely, though such results have not been analyzed extensively. These studies provide further evidence that complexity is not strictly correlated with adaptive behavior, and that both driven and passive trends plateau far from the maximum values possible in the model.

3.7 Conclusion

This chapter has introduced Polyworld—the model itself, the qualitative results of past studies, and the quantitative framework which has typically been used to perform analysis. Prior work has theorized a significant driven evolutionary trend toward the amplification of adaptive properties, specifically the complexity of agents’ neural dynamics. This trend takes place in the early stage of a simulation during which the

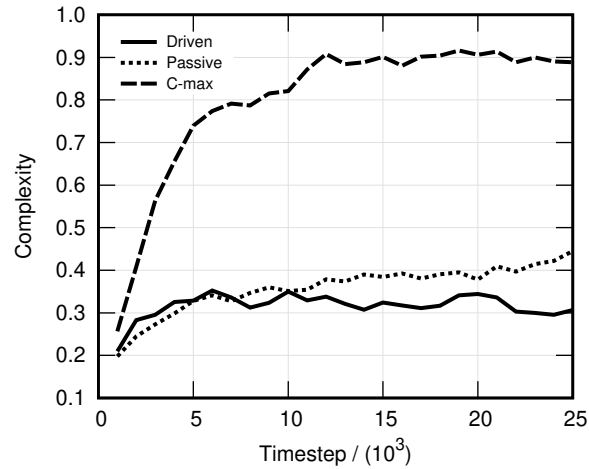


Figure 3.5: Complexity vs. timestep for typical driven, passive, and C-max simulations originally reported in [Yaeger et al. \(2008\)](#) and [Yaeger \(2009\)](#).

population is adapting behaviorally to its environment. Once a successful survival strategy arises, the driven trend plateaus as evolution begins to select for the stability of this strategy. Given this understanding of the model, it is now possible to introduce the details of the particular experiments upon which this dissertation is largely based.

Chapter 4

Experiments

This chapter provides an overview of the [Legacy](#) experimental condition and presents some of its high-level results. Similar experiments were first described in the proceedings of the eleventh Artificial Life conference ([Yaeger et al., 2008](#)). The [Legacy](#) condition has been reused in this dissertation in order to more directly extend these prior studies and to allow their results to provide a baseline for the work described here.

4.1 Overview

The [Legacy](#) environment is as previously shown in [Figure 3.1](#): food grows in two patches at opposite ends of the world, which is separated into three equal-area regions by two barriers. These barriers begin at the “northern” edge of the world and extend for 90% of its depth, meaning that agents may only move between regions at the world’s “southern” extreme. This physical separation of subpopulations introduces geographical niches and aids in the preservation of genetic diversity.

At the beginning of a simulation, the world is populated with 180 genetically uniform agents. The seed genome is of nearly minimal complexity but, in the interest of simulation viability, does code for some sensible default behavior. During the course of a simulation, the population is permitted to range from 90 to 300 agents. At 90 agents, a “population crash” occurs, triggering early termination of the simulation. At 300 agents, additional births are suppressed. Neither of these situations is common: in the ten driven simulations reported here, the population never dropped below 150 agents, and only four births were suppressed. The typically observed population range in these experiments is between 170 and 260 agents.

A minimum of 90 food items is enforced at all times, which is a relatively large value given the physical

distribution of food patches. This ensures a plentiful supply of energy and makes for a comparatively nonthreatening environment. To facilitate tracking and maintenance of the world's food supply, no carcass food items are created upon agent death.

Energy balance is accomplished via the `ApplyLowPopulationAdvantage` and `NumDepletionSteps` parameters. I.e., energy usage is increased or decreased linearly whenever the population is, respectively, above or below its nominal level of 180 agents. Substantial selection pressure is imposed by requiring agents to have a health energy level of 50% (relative to capacity) in order to reproduce. Between 20% and 80% of a parent's health energy is transferred to its offspring at birth, depending on the value of the parent's `MateEnergyFraction` gene.

Agents' neural networks contain a maximum of five internal neural groups with up to 16 excitatory and 16 inhibitory neurons per group. This yields a maximum of 32 neurons in each internal group, corresponding to an overall maximum of 160 internal neurons in each network. Network inputs are as previously shown in Figure 3.2: red, green, and blue Vision; the agent's health Energy level; and a source of Random noise. Each network output controls the agent's volition toward one of its possible actions: Eat, Mate, Fight, Move, Turn, Light, and Focus. Neural energy costs (normally assessed for each neuron and synapse) are suppressed in order to avoid imposing physiological constraints on network topology.

Prior work (Yaeger et al., 2008; Yaeger, 2009) has typically investigated evolutionary trends by running simulations for 30,000 timesteps. However, as discussed in Chapter 3, behavioral adaptation to the Legacy environment is generally achieved within the first several thousand timesteps, after which point most driven evolutionary trends usually stagnate. In the interest of computational economy, the experiments conducted here focus on this period of behavioral adaptation by analyzing only 15,000 timesteps.

Due to Hebbian learning (see Equations 3.4 and 3.5), an agent's neural anatomy can vary significantly throughout its lifetime, as synaptic weights are constantly changing in response to neural activity. In keeping with prior work, most of the analyses presented here make use of the neural anatomy recorded at agent death. It is therefore desirable to have agents die naturally rather than having their lives artificially truncated at the end of the simulation. To accomplish this, simulations are run for an additional 1000 timesteps (the default value of `MaxLifeSpan`) beyond the 15,000 timesteps used for analysis. This guarantees that all agents born within the first 15,000 timesteps of a driven simulation will die naturally. Note that no such guarantee can be made for passive simulations: since deaths occur randomly, lifespans are not constrained by `MaxLifeSpan`. Passive agents born within the first 15,000 timesteps may therefore survive beyond the 16,000th timestep,

though in practice this affects less than 0.1% of the population. Moreover, there is no fixed number of additional timesteps that would guarantee natural deaths for such agents, so they are simply excluded from any analysis.

4.2 High-Level Results

The analyses presented in Chapters 5, 6, and 7 are all based on the same set of ten driven/passive pairs of *Legacy* simulations. Before presenting these analyses, I report some high-level results for these simulations, both for introductory purposes and in order to verify the observations noted in prior work. I demonstrate the existence of a significant driven trend toward the evolution of adaptive behavior, accompanied by increases in the complexity of agents' neural dynamics. Further, I corroborate quantitatively the explanation for this trend previously proposed by [Yaeger et al. \(2008\)](#).

Typical driven and passive populations are shown in Figure 4.1 at both the early and late stages of a simulation. The behavioral adaptation exhibited by late-stage driven populations is most obvious from their physical distribution: well-adapted agents locate themselves optimally according to resource availability, congregating in the northern and southern food patches while avoiding the energy-starved central portion of the world. In contrast, late-stage passive agents ignore the presence of food, spreading themselves essentially uniformly throughout the world. Thus, as previously demonstrated by [Griffith and Yaeger \(2006\)](#), driven populations achieve an ideal free distribution ([Fretwell and Lucas, 1969](#)) with respect to food. Such a distribution maximizes the utilization of resources while minimizing competition for those resources. Videos of typical driven and passive *Legacy* simulations are available at <https://youtu.be/FZq2TcyRTTo> and https://youtu.be/Stv0Y_F5-2c.

Further evidence for the driven population's behavioral adaptation comes from its agents' body colors. Since two color components (blue and red) are controlled by two action volitions (Mate and Fight), the population-wide predisposition toward those actions can to some degree be inferred visually. Agents in late-stage driven populations are biased toward mating and against fighting, which manifests as a primarily blue coloration in the typical agent. These biases are adaptive from the perspective of both the individual and the population. Mating is a cooperative action, required to ensure survival of the population via the propagation of adaptive genes. Fighting, however, is an uncooperative action. It not only results in energy *loss* for the target agent, but is also associated with significant energy *usage* for the source agent, as the

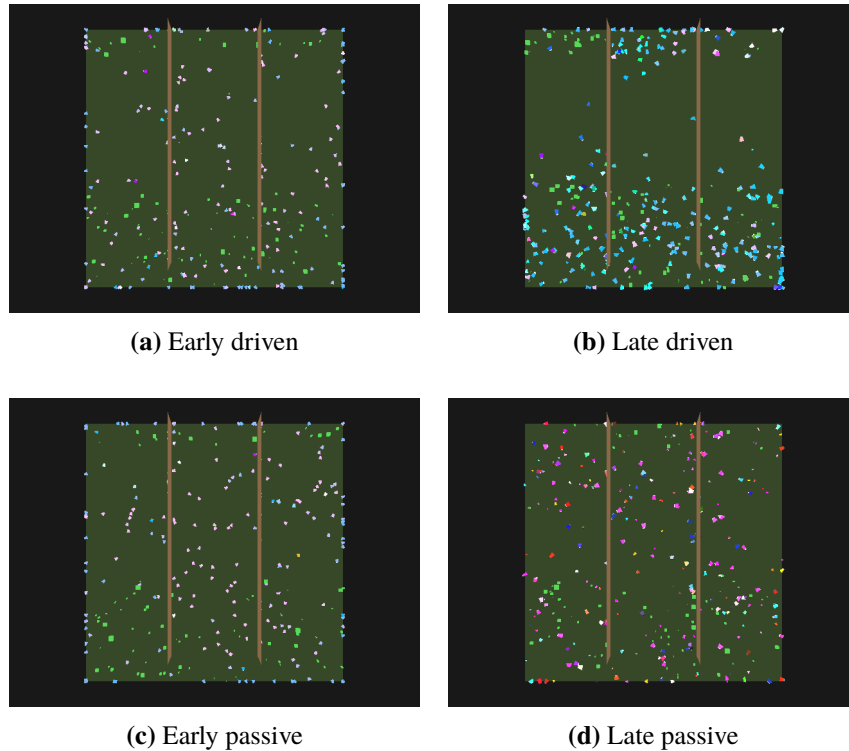


Figure 4.1: Typical [Legacy](#) simulations in both driven (top) and passive (bottom) modes. Early (left) and late (right) stages represent the 100th and 10,000th timesteps, respectively. Contrast has been enhanced.

Fight action is by far the most costly in terms of energy consumption. Furthermore, fighting is difficult for an agent to take advantage of: it requires visual tracking and pursuit of another agent and—in the [Legacy](#) condition—does not result in the creation of a carcass food item, making predation impossible. Effective use of the `Fight` action would thus seem to require a longer time scale to evolve than the 15,000 timesteps analyzed in these simulations, and is in any case unnecessary for resource competition given the abundance of food in the [Legacy](#) environment.

4.2.1 Population

The evolutionary trend in population is shown in [Figure 4.2](#). Population rises sharply in the first 50 or so timesteps, due mainly to peculiarities of the [Legacy](#) condition. Specifically, at the beginning of a simulation, agents are initialized at full energy level (100% of capacity). This “free” initial supply of energy, coupled with the seed genome’s predisposition toward mating, invariably results in a population boom.

This situation is, however, not typical of a simulation’s steady state. The initial energy surplus is quickly passed from parents to offspring, after which the typical agent has less energy than required to reproduce.

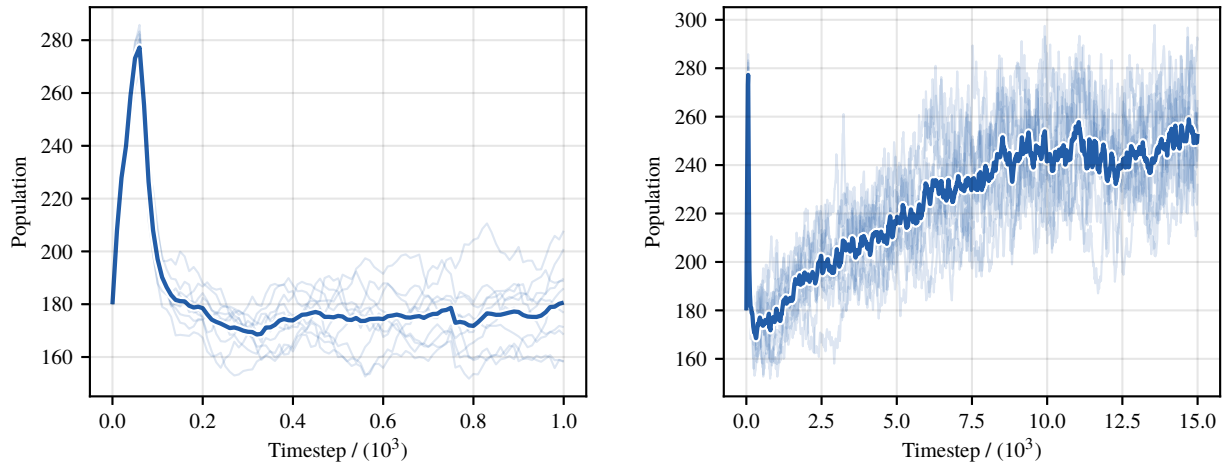


Figure 4.2: Population vs. timestep for ten *Legacy* simulations over the first 1000 timesteps (left) and all 15,000 timesteps (right). Light lines represent individual simulations, while the heavy line represents the mean over all ten simulations.

The population then drops as sharply as it rose, with less fit (or simply less lucky) agents dying off due to inflated energy usage, competition for resources, and the inability of unadapted agents to effectively forage.

After another 50 or so timesteps, this mass extinction slows as the simulation stabilizes in a regime where gross energetics are more typical, based on the balance among the number of agents, their typical energy level, and the availability of food. At this point, the population begins to grow steadily in response to the evolution of adaptive behavior, increasing throughout the first 10,000 timesteps. As a successful survival strategy spreads throughout the population, resource utilization is eventually maximized. The population effectively reaches a saturation point with respect to the amount of energy available in the environment, leveling off around 245 agents.

4.2.2 Adaptive Behavior

Evolutionary trends in adaptive behavioral strategies are quantified in Figure 4.3. Driven and passive trends are reported for foraging and mating, as measured by population-wide rates of food consumption and reproduction. Note that the passive birth rate reported here refers to “virtual” births (i.e., those that would have occurred if evolutionary dynamics had been unconstrained). Recall that, since passive populations are forced to evolve in lockstep with a corresponding driven simulation, birth rates are identical between the two modes. But while natural births are suppressed in passive mode, they are still recorded as virtual births for the purposes of later analysis. Thus a comparison between *actual driven* births and *virtual passive* births

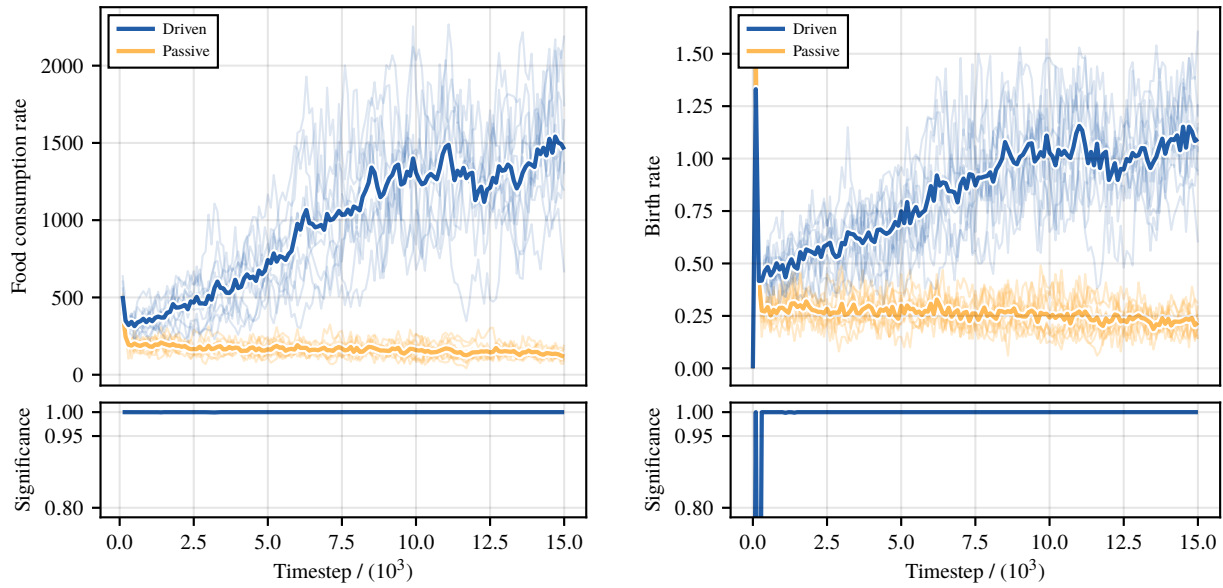


Figure 4.3: Food consumption rate (left) and birth rate (right) vs. timestep for ten driven/passive pairs of Legacy simulations. Passive birth rate refers to “virtual” births that would have occurred if evolutionary dynamics had been unconstrained. Light lines represent population means for individual simulations, while heavy lines represent meta-means over all ten simulations. Significance is calculated as $1 - p$ for a two-tailed, paired Student’s t -test of driven and passive means.

allows quantification of the relative reproductive fitness of driven and passive populations.

Both foraging and mating become more prevalent over evolutionary time, but only in driven simulations, matching the qualitative observations noted in prior work. Driven trends track fairly closely with population (compare to Figure 4.2), while passive trends remain essentially stagnant throughout the course of a simulation. (On close inspection, passive trends are observed to be slightly decreasing, indicating random drift away from the seed genome’s modest predisposition toward seeking food and mates.) Driven trends in food consumption and reproduction thus appear to be mediated both by selection for adaptive behavior—accounting for the differences between driven and passive simulations—and by increases in population toward full resource utilization. Prior work has hinted that coevolution may drive further advances in adaptive trends, with more effective behavioral strategies eventually emerging. However, these arise somewhat sporadically and are typically only observed on longer time scales than those investigated here.

4.2.3 Complexity

As shown in Figure 4.4, driven and passive evolutionary trends in the complexity of agents’ neural dynamics demonstrate strong agreement with prior work (compare to Figure 3.4). Driven simulations tend toward

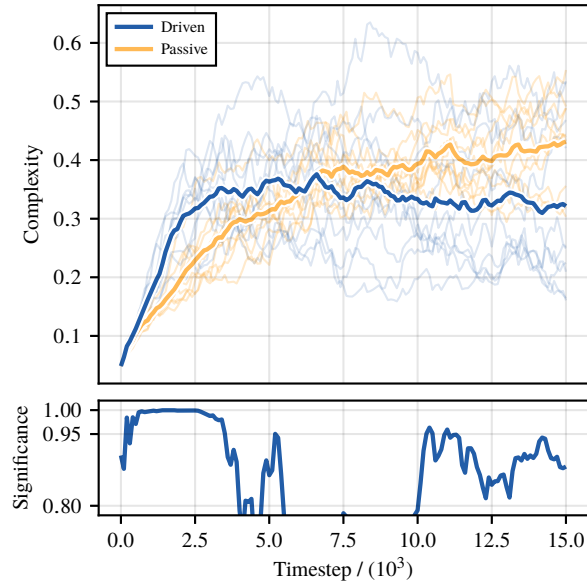


Figure 4.4: Complexity vs. timestep for ten driven/passive pairs of [Legacy](#) simulations. Light lines represent population means for individual simulations, while heavy lines represent meta-means over all ten simulations. Significance is calculated as $1 - p$ for a two-tailed, paired Student’s t -test of driven and passive means.

increased complexity, with this trend significantly faster than the corresponding passive trend during the period of behavioral adaptation (i.e., the first several thousand timesteps).

While [Figure 4.4](#) reports the same complexity metric used in previous studies, there are significant differences in its method of calculation. Prior work has used an *in vivo* technique relying on the time series of neural activations recorded during agents’ lifetimes. To overcome difficulties with this technique, I instead use an *in vitro* technique in this work, which is based on time series generated post hoc by exposing agents’ neural networks to uncorrelated noise. This method will be discussed in more detail in [Section 4.3](#), where comparisons will be made between the two techniques.

4.2.4 Evolutionary Selection

What is the underlying cause of the complexity trend reported in [Figure 4.4](#)? [Yaeger et al. \(2008\)](#) provide a probable explanation—driven selection for the amplification of adaptive properties, followed by selection for the stability of those adaptations—but they do not verify this hypothesis quantitatively. Given that all evolutionary trends are genetically mediated, it seems worthwhile to investigate the population genetics of these simulations in order to confirm such a hypothesis.

The analysis conducted here employs a novel metric inspired by the K_a/K_s ratio. Typically used

in evolutionary biology, K_a/K_s measures “the ratio of the number of nonsynonymous substitutions per nonsynonymous site (K_a) to the number of synonymous substitutions per synonymous site (K_s)” (Hurst, 2002). By comparing putatively non-neutral and neutral mutations, K_a/K_s indicates whether genes are drifting randomly ($K_a/K_s \approx 1$) or are instead experiencing evolutionary selection ($K_a/K_s \neq 1$). In the context of evolutionary biology, synonymous substitutions provide a *proxy* for neutral evolution, since the true rate of neutral mutation cannot be known. In Polyworld, however, a precise model of neutral evolution already exists in the passive mode. Thus, to measure evolutionary selection, it is sufficient to simply compare the diversity of alleles between driven and passive modes, with passive diversity providing a *direct measurement* of neutral evolution.

I quantify the diversity d_i of the i th gene in information-theoretic terms as

$$d_i = \frac{H(A_i^k)}{8 - k}, \quad (4.1)$$

where H is the Shannon entropy, calculated here over the population-wide distribution of alleles A_i^k for the i th gene. The grouping parameter k indicates that raw alleles (each a single byte in $[0,255]$) may be aggregated by ignoring differences in the least significant k bits, resulting in an effective set of 2^{8-k} alleles. E.g., by selecting $k = 4$, alleles are collected into groups of 16, such that alleles in $[0, 15]$ are considered identical, as are alleles in $[16, 31]$, $[32, 47]$, etc. This grouping treats mutations occurring in lower-order bits as effectively neutral. The denominator of Equation 4.1 ensures that, regardless of the value of the grouping parameter, diversity is normalized into $[0, 1]$ (assuming that entropy is calculated in bits). Note that diversity is minimized for genes having a single fixed allele, as in the initial population of a simulation. It is maximized when alleles are distributed uniformly over the entire range of possible values, a state approached by the neutrally evolving passive simulations.

Evolutionary selection S is calculated using the average ratio of driven to passive diversity:

$$S = \frac{1}{n} \sum_{i=1}^n \frac{d_i^D}{d_i^P} - 1, \quad (4.2)$$

where n is the total number of genes, and the superscripts D and P emphasize that diversity is calculated separately over driven and passive populations. The subtraction of unity in Equation 4.2 is made for naturalness of interpretation, such that the sign of S matches the type of selection experienced by the driven

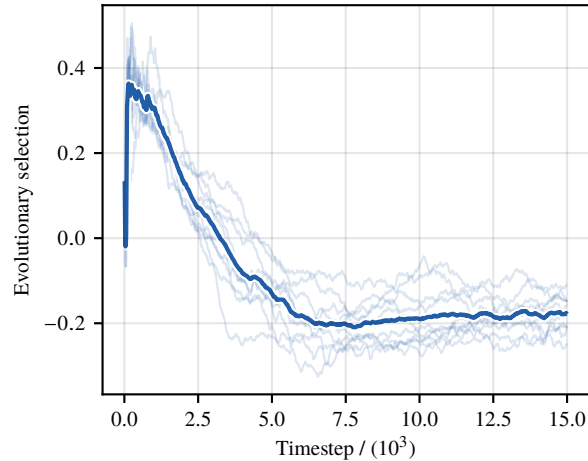


Figure 4.5: Evolutionary selection vs. timestep for ten driven/passive pairs of Legacy simulations. Light lines represent individual simulations, while the heavy line represents the mean over all ten simulations. Diversity is calculated using ungrouped alleles (i.e., $k = 0$ in Equation 4.1).

population. $S > 0$ indicates high genetic diversity in the driven simulation relative to neutral mutation (favoring change and thus indicating positive—typically directional—selection). I.e., there is pressure to *escape* the ancestral state via selection *for* adaptive alleles. $S < 0$ indicates low genetic diversity in the driven simulation relative to neutral mutation (opposing change and thus indicating negative—typically stabilizing—selection). I.e., there is pressure to *conserve* the ancestral state via selection *against* deleterious alleles.

The trend in evolutionary selection is shown in Figure 4.5 for ungrouped alleles (i.e., $k = 0$ in Equation 4.1), though this trend is largely insensitive to the choice of grouping parameter. Driven simulations are dominated by positive selection during the first few thousand timesteps, followed by negative selection throughout their remainder. This result agrees with intuitions from prior work: early in the simulation, a driven population experiences strong selection pressure to escape the relatively unfit seed genome. During this stage, genetic diversity accumulates more quickly than in the corresponding passive population, yielding positive evolutionary selection. Once a “good enough” survival strategy arises, evolution acts to conserve the genes responsible for it, allowing the strategy to spread throughout the population. During this stage, genetic diversity is suppressed, yielding negative evolutionary selection.

Figure 4.5 provides quantitative support for Yaeger et al.’s (2008) explanation of the evolutionary trend in complexity. Specifically, there is early selection for the amplification of adaptive properties, followed by selection for the stability of those adaptations. I leverage this result throughout Chapters 5, 6, and 7: if

Gene	Diversity
Strength	0.578
Bias[Mate]	0.680
MutationRate	0.685
Bias[Speed]	0.699
InternalNeuronGroupCount	0.702
MateEnergyFraction	0.704
Bias[Eat]	0.726
Bias[Yaw]	0.727
Green	0.742
Size	0.750

Table 4.1: Genes by diversity for ten driven [Legacy](#) simulations. Diversity is calculated using groups of 16 alleles (i.e., $k = 4$ in Equation 4.1) over the entire set of genomes observed in each simulation. The ten least diverse genes are reported here based on the average per-simulation diversity.

some neural property of interest is significantly amplified in the early stage of driven simulations (relative to passive simulations), I take this as evidence for the property’s adaptive significance.

A more detailed analysis of population genetics could investigate evolutionary selection for specific alleles or schemata, rather than on a genome-wide basis as shown here. Genetic features that exhibit positive selection could then be correlated with neural properties of interest (e.g., complexity), providing more direct confirmation of those properties’ adaptive significance.

Preliminary work has indicated that the direction and magnitude of selection vary considerably among genes. Table 4.1 shows the least diverse genes for ten driven [Legacy](#) simulations. These genes typically experience the strongest negative selection or, equivalently, the most aggressive stabilization of their alleles. Here I use groups of 16 alleles (i.e., $k = 4$ in Equation 4.1) as a first approximation toward ignoring effectively neutral mutations. Passive simulations are excluded from this analysis since their randomly drifting genes all tend toward roughly the same level of diversity.

The Strength gene is by far the most stable in the [Legacy](#) condition: the increase in diversity from the Strength gene to the second-least-diverse gene (Bias[Mate]) is roughly equivalent to the increase from the second- to the 38th-least-diverse gene. Note from Figure 4.6 that the Strength gene experiences significant directional selection toward smaller values before stabilizing. I.e., in order to solve the energy balance problem posed by the [Legacy](#) environment, evolution drives agents to be weak, pursuing fixation of the Strength gene considerably more aggressively than any other. This is reasonable since, while stronger agents are capable of inflicting more fight damage, they consume more energy for all actions. High strength

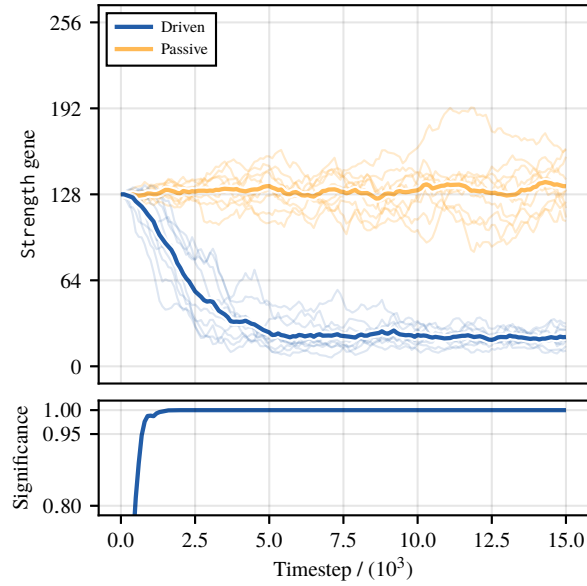


Figure 4.6: Strength gene vs. timestep for ten driven/passive pairs of *Legacy* simulations. Light lines represent population means for individual simulations, while heavy lines represent meta-means over all ten simulations. Significance is calculated as $1 - p$ for a two-tailed, paired Student’s t -test of driven and passive means.

is thus a hindrance to energy conservation, while the advantage it confers is difficult to exploit. Recall that fighting is not emphasized in the survival strategy that develops in these simulations.

Despite a major goal of these experiments being to investigate the evolution of neural networks, most of the ten genes listed in Table 4.1 have no effect on neural wiring: one is meta-genetic, three are physiological, and four are simple neural biases. This observation is revisited in the design of the experiments reported in Chapter 8, where meta-genetic and physiological genes are fully constrained, forcing all adaptation to take place in neural genes.

4.3 *In Vitro* Analysis

Complexity has traditionally been measured *in vivo* by applying Equation 3.9 to the time series of neural activations recorded during an agent’s lifetime. However, this technique is associated with a number of difficulties. First, the time series length is limited by the agent’s lifespan. The amount of available data thus varies from agent to agent and, in the case of short-lived agents, can be insufficient to construct the probability distributions needed to calculate the relevant information-theoretic quantities. Indeed, in the ten driven/passive pairs of *Legacy* simulations reported here, roughly 13% of agents had a lifespan (in timesteps)

that was less than their neuron count. In prior work, such agents would have been excluded from complexity analysis due to potential numerical instability.

Second, an agent’s neural activity is driven primarily by its visual input, which is highly contextual. What an agent sees during its lifetime is affected by aspects of the simulation that are irrelevant to the potential complexity of its neural dynamics, yet they become a significant factor in the complexity calculation when the *in vivo* technique is used. Consider an agent living in a sparsely populated area of the world that offers little in the way of visual input. The neural activity of such an agent will be largely determined by its network’s intrinsic dynamics—those that prevail in the absence of external stimuli—and completely unrepresentative of the dynamics achievable when driven by more substantial input.

Third, due to Hebbian learning, synaptic weights change throughout an agent’s lifetime. *In vivo* complexity is thus somewhat difficult to relate to other analyses, which have typically made use of the neural anatomy recorded at agent death.

To summarize, the *in vivo* technique is highly variable from agent to agent and, more crucially, can be numerically unstable due to insufficient data. To address these difficulties, I instead calculate complexity *in vitro* using fixed-length time series generated post hoc by exposing agents’ neural networks to uncorrelated noise. I.e., after a simulation is completed, an agent’s neural network is reconstituted and analyzed outside of the Polyworld environment. At each timestep, each neural input is set to a random value drawn uniformly from $[0, 1]$, and Equation 3.9 is applied to the resulting time series of neural activations. All agents are analyzed for 1000 timesteps, which has proved by inspection to provide sufficient numerical stability, and in any case is the maximum used in prior work (since `MaxLifeSpan` is set to 1000 in the `Legacy` condition). This technique neatly addresses issues of inter-agent variability: it not only maintains consistency in the quantity of data, but it also ensures that sensory input is statistically similar across all agents. For compatibility with other analyses, the *in vitro* technique makes use of the neural anatomy recorded at agent death. Synaptic weights are held constant, ensuring that the *analyzed* neural anatomy remains the same as the *recorded* neural anatomy throughout the calculation process.

Uncorrelated noise is, of course, completely unlike the well-ordered visual scene that confronts an agent during its lifetime. (An exception to this is the offline gestation period during which neural inputs are similarly randomized.) As a result, dynamics can be driven into areas of the neural activation state space that are not typically visited *in vivo*. Thus, while the *in vitro* technique represents a more mathematically principled approach, care must be taken in the interpretation of its results. It characterizes the neural dynamics that

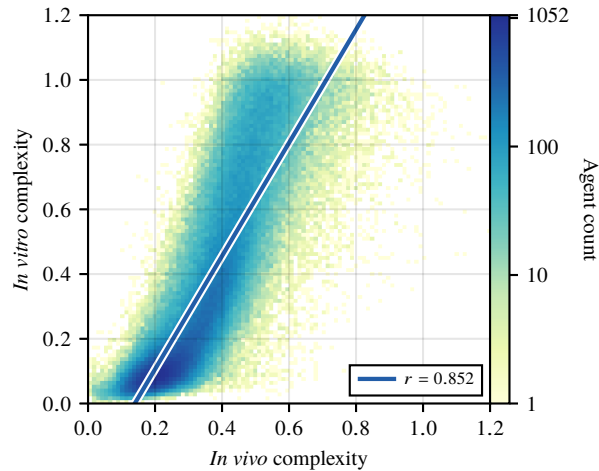


Figure 4.7: *In vitro* complexity vs. *in vivo* complexity for ten driven/passive pairs of Legacy simulations. The Pearson correlation coefficient r is reported for a linear least-squares regression.

agents *are capable of* in general, not those that they *happen to exhibit* during a simulation.

A direct comparison of *in vivo* and *in vitro* complexity is shown in Figure 4.7. Most notable is the increased range demonstrated by the *in vitro* technique: large *in vivo* values tend to become larger *in vitro*—due to a fuller exploration of the neural activation state space—while small *in vivo* values tend to become smaller *in vitro*—thanks to a reduction in spurious correlations. These plague the *in vivo* technique, especially in short-lived agents where insufficient data leads to undersampling.

Despite their differences, there is generally good agreement ($r = 0.852$) between the two techniques. In this work, I prefer the *in vitro* technique for its inter-agent consistency and numerical stability, using it heavily in the analyses reported in Chapters 6 and 7.

4.4 Conclusion

This chapter has introduced the Legacy experimental condition, reported high-level results of ten driven/passive pairs of simulations, and demonstrated the strong agreement of these results with those of prior work. I have also validated the quantitative framework that is used throughout this work to recognize adaptively significant neural properties. Assured of a well-grounded approach, I turn now to the analyses that constitute the major contribution of this dissertation: the investigation of evolutionary trends in neural structure, dynamics, and information processing.

Chapter 5

Analysis of Structure

This chapter presents an analysis of Polyworld’s neural networks using the tools of graph theory. Such an approach emphasizes a network’s structure—the architecture realized by its interconnected neurons and synapses. Similar work was originally reported in [Yaeger et al. \(2010\)](#). For a more comprehensive introduction to graph theory, see [Newman \(2010\)](#).

5.1 Foundations

A graph G is a mathematical structure consisting of a set of vertices V (each representing some entity under consideration) and a set of edges E (each representing some relationship between a pair of entities). As a concrete example, consider the graph in [Figure 5.1a](#) showing a friendship network. Each rectangle is a vertex representing a person, and each line is an edge connecting two people who are friends with one another. As shown in [Figure 5.1b](#), this graph may be represented succinctly as an adjacency matrix. Each entry in this matrix indicates the presence or absence of an edge—as an entry of one or zero, respectively—between the vertices corresponding to the entry’s row and column.

Note that the representations in [Figure 5.1](#) betray some details about how the concept of friendship is formalized here. First, friendship is an *undirected* relationship: since Anthony is friends with Barbara, Barbara must be friends with Anthony. This results in a symmetric adjacency matrix, where each row is equivalent to the corresponding column. Second, friendship is a *binary* relationship: either two people are friends, or they’re not—with no gradations between these extremes. This results in an adjacency matrix whose entries consist entirely of ones and zeros.

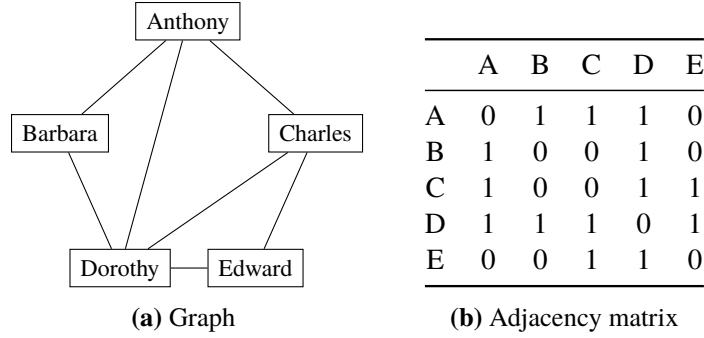


Figure 5.1: Friendship network.

More formally, for an entry a_{ij} on the i th row and j th column of the adjacency matrix, we have

$$\forall i, j : a_{ij} = a_{ji} \in \{0, 1\}. \quad (5.1)$$

Note further that this concept of friendship appears to prohibit *loops* or self-relationships. I.e., a person cannot be friends with himself. This results in an adjacency matrix with zeros on the diagonal:

$$\forall i : a_{ii} = 0. \quad (5.2)$$

Of course, the concept of friendship could have been formalized differently, in which case Equations 5.1 and 5.2 may not have held.

In this work, vertices and edges are used to represent the neurons and synapses of Polyworld’s neural networks. The resulting graphs are directed, weighted, and signed. Synapses are *directed* in that they have a well-defined orientation, projecting from a *presynaptic* neuron to a *postsynaptic* neuron. Individual connections are not reciprocal, though a pair of neurons can be connected in both directions via separate synapses. Each synapse is *weighted* by its efficacy, representing the presynaptic neuron’s degree of influence on the postsynaptic neuron. These graphs may thus be represented more accurately not as adjacency matrices, but as weight matrices:

$$\forall i, j : w_{ij} \in [-w_{\max}, w_{\max}], \quad (5.3)$$

where each entry w_{ij} denotes the weight of the synapse connecting the i th neuron to the j th neuron, with the absolute maximum synaptic weight w_{\max} corresponding to the `MaxSynapseWeight` parameter. *Signed* weights support the notion that the presynaptic neuron’s influence on the postsynaptic neuron may be either

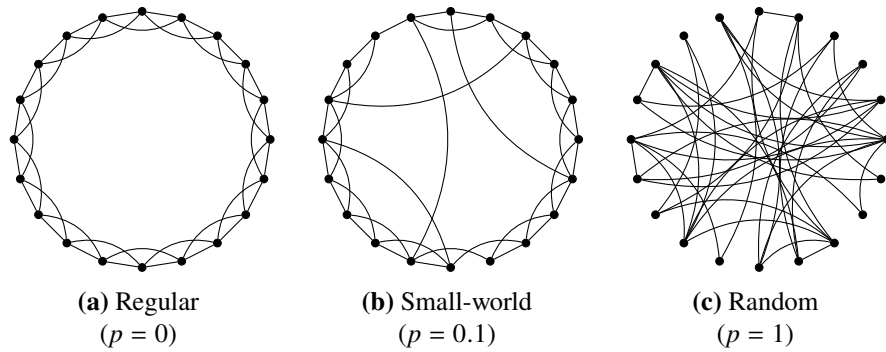


Figure 5.2: Graphs for the Watts–Strogatz model with $n = 20$, $k = 4$, and representative values of $p \in [0, 1]$.

excitatory or inhibitory, corresponding to either positive or negative weight. As in the friendship network of Figure 5.1, Polyworld’s neural model prohibits loops: a neuron cannot synapse onto itself.

Graph theory provides many metrics with which to characterize individual vertices and edges, neighborhoods, or entire graphs. Given that Polyworld’s neural networks admit of a straightforward graph-theoretic formalization, it is natural to apply these metrics to investigate how neural structure changes over evolutionary time. Particularly relevant to this work is the “small-world” property (Watts and Strogatz, 1998). To motivate this concept, first consider the regular lattice shown in Figure 5.2a, topologically a ring structure in which each vertex is connected to its four nearest neighbors. This graph is highly locally clustered, as each vertex’s neighbors are likely to themselves be neighbors of one another. However, paths between arbitrary vertices are relatively long, as the lattice contains no “shortcuts”: traveling between distant vertices requires movement by short hops between adjacent neighborhoods. Now consider a graph with the same number of vertices and edges, but where each edge has been randomly rewired (see Figure 5.2c). In such a graph, the ordered structure that characterizes the regular lattice is completely absent, resulting in poor local clustering. However, paths between arbitrary vertices are minimal due to the presence of many long-range connections.

It might seem that locally well-clustered neighborhoods and globally short path lengths would be mutually exclusive properties. Yet this is not the case, as shown by Watts and Strogatz (1998), who developed a model to smoothly interpolate between the two extremes of regular lattices and random graphs. In this model, we begin with a graph having n vertices, each connected to its k nearest neighbors. We then randomly rewire each edge with probability p . The presence of just a few long-range connections (see Figure 5.2b) drastically reduces the typical path length without significantly destroying the graph’s locally clustered structure. Such a graph is said to exhibit the small-world property.

Small-worldness has been observed in many contexts—neural (Sporns and Zwi, 2004), biological (Wag-

ner and Fell, 2001), social (Newman, 2001), linguistic (Ferrer i Cancho and Solé, 2001), and technological (Albert et al., 1999). It may thus be a general hallmark of complex networks. I therefore hypothesize an evolutionary trend in Polyworld’s neural networks toward small-worldness: this locally clustered yet globally proximal *structure* might provide the basis for the locally segregated yet globally integrated *dynamics* recognized in prior work.

5.2 Methods

Existing work (Yaeger et al., 2010) has investigated small-worldness in Polyworld’s neural networks similarly to Watts and Strogatz (1998). Specifically, the *clustering coefficient* is used to quantify local structure, while path length–based metrics such as the *characteristic path length* are used to measure the typical distance between vertices. For a directed, binary graph G , the clustering coefficient C may be understood as the proportion of a vertex’s neighbors that are themselves neighbors of one another, averaged over all vertices:

$$C = \frac{1}{n} \sum_{i=1}^n \frac{|E_i|}{k_i(k_i - 1)}, \quad (5.4)$$

where E_i is the set of edges in the neighborhood N_i of the i th vertex:

$$N_i = \{v_j : e_{ij} \in E \vee e_{ji} \in E\}, \quad (5.5)$$

$$E_i = \{e_{jk} : v_j, v_k \in N_i \wedge e_{jk} \in E\}. \quad (5.6)$$

The graph’s characteristic path length L is the distance of the shortest path between two vertices, averaged over all pairs of vertices:

$$L = \frac{1}{n(n-1)} \sum_{i=1}^n \sum_{\substack{j=1 \\ j \neq i}}^n d_{ij}, \quad (5.7)$$

where d_{ij} is the distance between the i th and j th vertices. Of course, in the case of a binary graph, the distance of the shortest path is simply the number of edges it contains.

Extensions to these metrics have been made to accommodate their calculation on weighted graphs (Barrat et al., 2004). One of the most important considerations is the treatment of edge weights in calculating path lengths. If weights correspond to a notion of *distance* between the connected vertices (as in, say, a highway network, where weights represent travel distances between cities), then a path’s length is determined by

simple summation of its edges' weights. If, however, weights correspond to a notion of *interaction strength* between the connected vertices (as in a neural network, where weights represent synaptic efficacies), then each weight must be inverted. The intuition is that larger weights (i.e., stronger interactions) should represent shorter distances. Disconnected graphs are still problematic, however, as they result in infinite distances between some pairs of vertices, causing Equation 5.7 to lack a finite solution. Such graphs are prevalent in Polyworld, especially during the early portion of a simulation.

Existing work has dealt with this problem by defining an ad hoc path length metric in which distances are capped at a theoretical maximum. In this work, I instead use *efficiency* as the primary metric for investigating neural structure. Introduced by [Latora and Marchiori \(2001\)](#), efficiency indicates the typical effectiveness of communication between pairs of vertices. It may be calculated either *locally* within a neighborhood or *globally* over the entire graph.

The efficiency E of a graph G is defined as

$$E(G) = \frac{1}{n(n-1)} \sum_{i=1}^n \sum_{\substack{j=1 \\ j \neq i}}^n \frac{1}{d_{ij}}, \quad (5.8)$$

where the distance d_{ij} from the i th vertex to the j th vertex is found here by summing the inverted weights of the edges constituting the shortest path. Note that d_{ij} is itself inverted in Equation 5.8, making efficiency better suited to graphs in which communication occurs in a parallel fashion: it is less sensitive to the existence of a few relatively large distances, and it remains finite when calculated over disconnected graphs. (If there is no path from the i th vertex to the j th vertex, then d_{ij} is infinite, and $1/d_{ij}$ is taken to be zero.) Before calculating efficiency, I normalize edge weights by their absolute maximum, yielding $E \in [0, 1]$.

Local efficiency corresponds directly to clustering coefficient (i.e., $E_L \sim C$) and is calculated as the average efficiency over all neighborhoods in the graph:

$$E_L = \frac{1}{n} \sum_{i=1}^n E(N_i). \quad (5.9)$$

Global efficiency corresponds inversely to characteristic path length (i.e., $E_G \sim 1/L$) and is simply the efficiency of the entire graph—exactly as in Equation 5.8:

$$E_G = E(G). \quad (5.10)$$

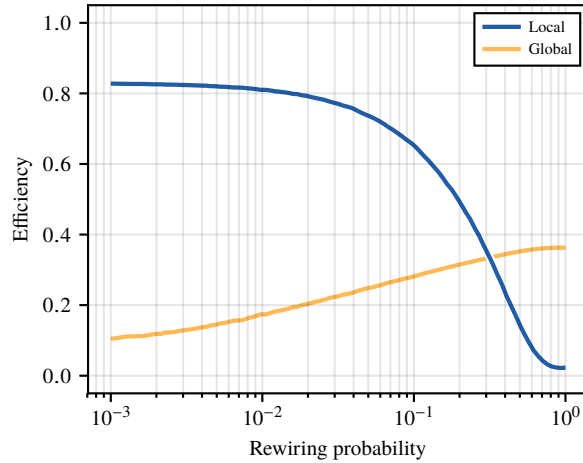


Figure 5.3: Efficiency vs. rewiring probability for the Watts–Strogatz model with $n = 500$ and $k = 10$.

Applied to the Watts–Strogatz model, these metrics behave as expected given their correspondence to clustering coefficient and characteristic path length. As shown in Figure 5.3, a regular lattice ($p = 0$) has maximal local efficiency and minimal global efficiency. A small proportion of random rewirings ($p \approx 0.1$) induces a significant increase in global efficiency relative to its maximum value. Local efficiency, meanwhile, decreases comparatively little until the graph is rewired more aggressively. These trends continue until reaching values typical of a random graph ($p = 1$), which exhibits minimal local efficiency and maximal global efficiency.

Unfortunately, the concept of efficiency does not extend well to signed graphs. As in prior work, I make the common simplification of using the absolute value of synaptic efficacy as an edge’s weight when calculating graph-theoretic metrics.

5.3 Results

Evolutionary trends in the local and global efficiency of Polyworld’s neural networks are shown in Figure 5.4. As expected, these metrics increase over evolutionary time. The driven trends are significantly faster during the period of behavioral adaptation. Both metrics remain at relatively small magnitudes, but this is not unexpected. It merely indicates that the simple environment used for these simulations may be “solved” using relatively inefficient neural structure.

In contrast to the complexity results reported in Figure 4.4, trends in local and global efficiency do not reach a hard plateau within the 15,000 timesteps analyzed here. Instead, modest increases in these metrics

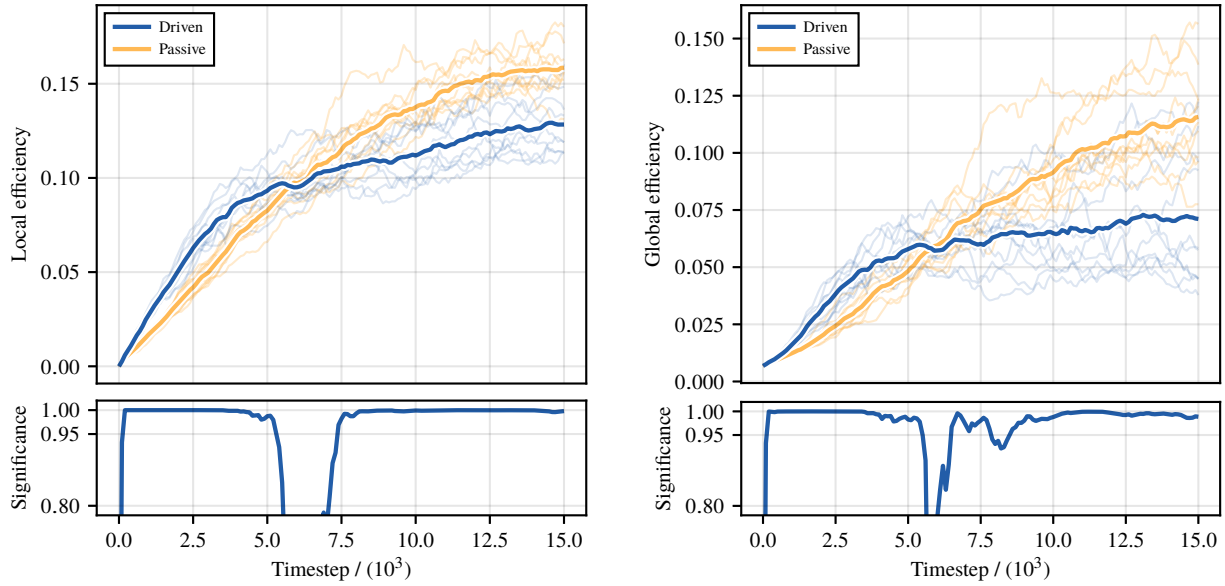


Figure 5.4: Local efficiency (left) and global efficiency (right) vs. timestep for ten driven/passive pairs of *Legacy* simulations. Light lines represent population means for individual simulations, while heavy lines represent meta-means over all ten simulations. Significance is calculated as $1 - p$ for a two-tailed, paired Student’s t -test of driven and passive means.

persist throughout the duration of the simulation, well past the period of behavioral adaptation. Presumably, these increases further facilitate agents’ survival and reproduction. However, this result has not been analyzed extensively.

5.4 Discussion

In the formulations used here, local and global efficiency are both normalized within $[0, 1]$. For convenience, we may take their product as a single combined measure of small-worldness:

$$W = E_L E_G \in [0, 1]. \quad (5.11)$$

Given its straightforward relationship to local and global efficiency, this metric exhibits a predictable evolutionary trend (see Figure 5.5). Its overall shape closely matches that of the trends shown in Figure 5.4. Specifically, there is a significant driven increase during the period of behavioral adaptation.

This trend toward small-worldness appears to be mediated by increasing synaptic weight and density. As shown in Figure 5.6, the initial stage of a driven simulation is typified by sparsely connected neural

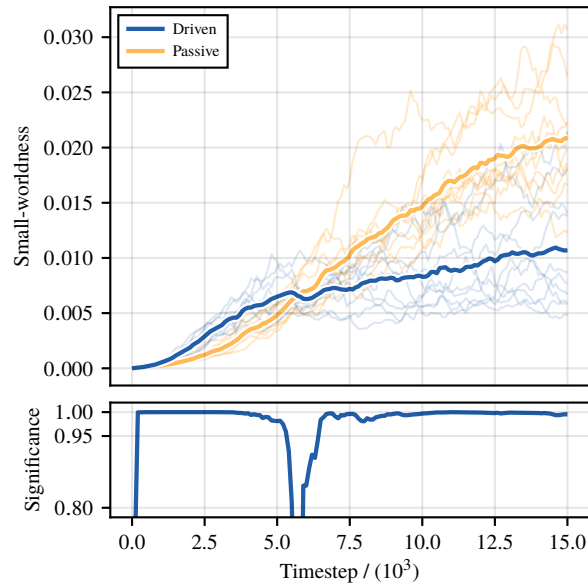


Figure 5.5: Small-worldness vs. timestep for ten driven/passive pairs of *Legacy* simulations. Light lines represent population means for individual simulations, while heavy lines represent meta-means over all ten simulations. Significance is calculated as $1 - p$ for a two-tailed, paired Student’s t -test of driven and passive means.

networks having weak synapses. Over time, synaptic connections become denser and stronger, due in part to evolutionary selection for larger values of the *ConnectionDensity* and *LearningRate* genes.

Figure 5.7 indicates a direct and fairly linear relationship between individual networks’ small-worldness and the average absolute (i.e., unsigned) weight of their synapses. However, the relationship between small-worldness and synaptic density is somewhat more complicated: the largest values of small-worldness are only observed at intermediate synaptic densities. Extremely sparse networks lack the required connectivity for effective communication, while extremely dense networks lack the required organization.

5.5 Conclusion

In *Polyworld*, evolution selects for small-world neural structure. The primary evidence for this claim comes from the evolutionary trends in local and global efficiency. During the period of behavioral adaptation, these trends are significantly faster in driven simulations than in passive simulations, suggesting a correlation between small-world structure and adaptive behavior.

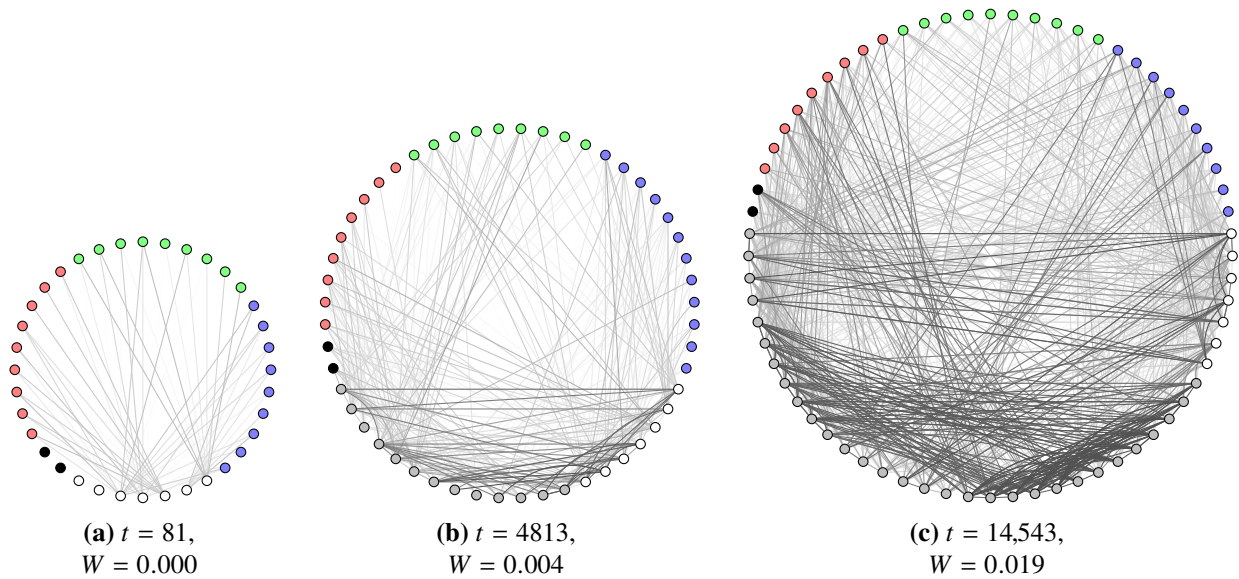


Figure 5.6: Typical neural networks for agents from the initial (left), early (center), and late (right) stages of a *Legacy* simulation. Vertex color indicates neuron type: red, green, or blue for visual input neurons; black for other input neurons; white for output neurons; and gray for internal neurons. Edge color indicates absolute synaptic weight: white for zero, black for the maximum, and shades of gray for intermediate values.

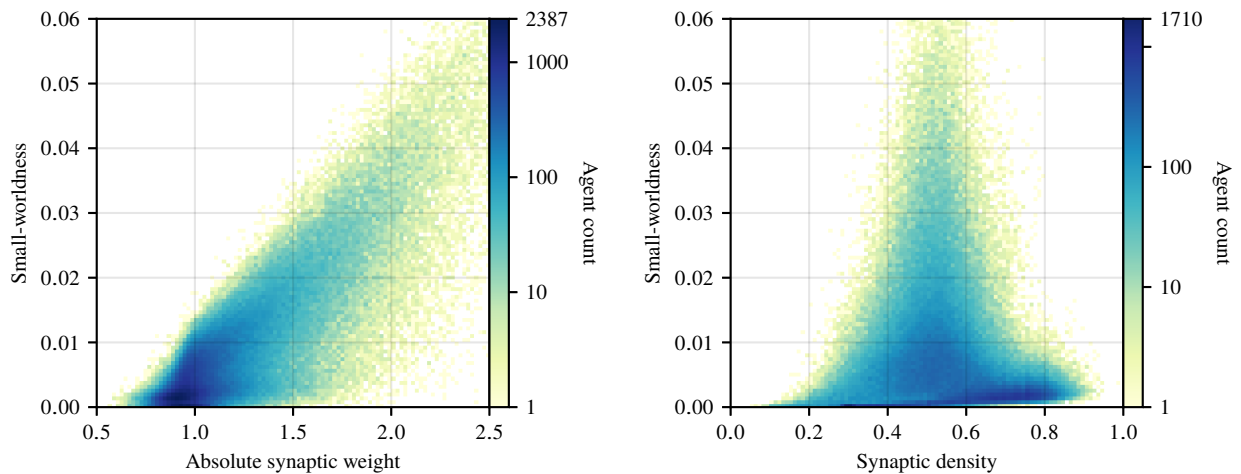


Figure 5.7: Small-worldness vs. absolute synaptic weight (left) and synaptic density (right) for ten driven/passive pairs of *Legacy* simulations.

Chapter 6

Analysis of Dynamics

This chapter presents an analysis of Polyworld’s neural networks using the tools of dynamical systems theory. Such an approach emphasizes a network’s dynamics—its time evolution on the neural activation state space. Similar work was originally reported in [Williams and Yaeger \(2017\)](#). For a more comprehensive introduction to dynamical systems theory, see [Abraham and Shaw \(2000\)](#).

6.1 Foundations

A dynamical system consists of a set of *state variables* and an *evolution operator* that describes how those variables change over time. The evolution operator typically takes the form of a set of differential or difference equations, depending on whether the time variable is taken to be continuous or discrete. At any given time, the system is located at a particular point in its *state space*, which is constituted by the set of all possible values of its state variables. The action of the evolution operator moves the state variables through this space, generating a sequence of states called a *trajectory*. Taken together, all such trajectories constitute the system’s *flow*.

Dynamical systems theory provides an extensive set of tools with which to characterize time-varying systems, from individual trajectories to entire flows. Especially relevant to this work are the *limit sets* of a system—portions of its state space that are invariant with respect to the evolution operator. Upon entering a limit set, a system’s state variables are kept there indefinitely by the action of the evolution operator. We are specifically interested in *stable* limit sets—those that attract nearby trajectories—since they determine the system’s long-term behavior.

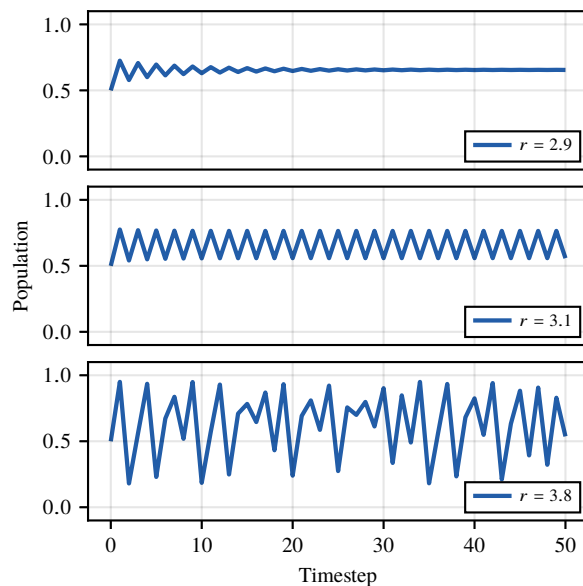


Figure 6.1: Time series for the logistic map with $x(0) = 0.5$ and representative values of r .

As a concrete example, consider the logistic map

$$x(t + 1) = r x(t) [1 - x(t)], \quad (6.1)$$

which models the evolution of a population x (expressed relative to its maximum value) over discrete timesteps t based on a parameter r that captures a combined rate of reproduction and starvation. Note that there is a slight abuse of notation here: Equation 6.1 appears to indicate that population is a function of time, and that the logistic map is therefore a *nonautonomous* dynamical system. However, this is not the case: the population in the next timestep depends only on the current population and the fixed reproduction/starvation parameter. I could more correctly represent $x(t)$ as x_t , but I retain the former notation for consistency with Chapter 7, where readability is impacted by placing the time variable in subscripts.

Figure 6.1 shows representative time series for the logistic map. The type of limit set exhibited by the system depends on the value of the reproduction/starvation parameter. With $r = 2.9$, the system settles on a single *fixed point*. With $r = 3.1$, the system enters a *limit cycle*, perpetually oscillating between two states. With $r = 3.8$, the system exhibits *chaos*, never entering a regularly recurring pattern.

Two hallmarks of chaos are aperiodicity and sensitive dependence on initial conditions. The former indicates that chaotic systems appear random; it is difficult to find patterns in their behavior even if they are completely deterministic. The latter indicates that arbitrarily similar initial states eventually yield wildly

different outcomes; long-term prediction of a chaotic system's behavior is thus impossible. However, chaos does not necessarily imply instability. Despite their local divergence, nearby trajectories may be globally confined to a subset of the system's state space—a chaotic attractor.

Many systems are capable of exhibiting both ordered and chaotic activity. The behavior of such systems is often governed by parameters that can cause chaos to appear or disappear. The “edge of chaos” refers to this transition between the ordered and chaotic regimes. E.g., as demonstrated in Figure 6.1, the reproduction/starvation parameter r may be used to tune the logistic map between order ($r = 2.9$) and chaos ($r = 3.8$). At a particular point in the system's parameter space ($r \approx 3.570$), a *bifurcation* occurs, qualitatively altering the system's long-term behavior. When poised at a bifurcation point, a system becomes extremely sensitive to perturbations in its parameters: even infinitesimal changes to their values can drastically alter the system's limit sets.

One of the first descriptions of the edge-of-chaos concept was provided by Langton (1990), whose analysis of cellular automata suggested that information storage, transmission, and modification are most effective in systems of intermediate chaoticity. A system that is too ordered cannot transmit or modify information: the necessary dynamics are prevented by the system's rigidity. A system that is too chaotic cannot store information: any coherent structure is quickly destroyed by the system's intrinsic disorder. Systems located near a critical point—somewhere “just this side of chaos”—are rigid enough to store information, yet flexible enough to transmit and modify it.

Critical dynamics are associated with a variety of computational systems, both biological (Skarda and Freeman, 1987; Goldberger et al., 1990; Schiff et al., 1994) and artificial (Packard, 1988; Kauffman and Johnsen, 1991; Bertschinger and Natschläger, 2004). However, it is important to note that critical dynamics do not guarantee high performance on arbitrary tasks (Mitchell et al., 1993). Instead, the optimal level of chaos for a particular task is dependent on the nature of the task itself. Yet it is evident that the dynamics of many systems may be separated into ordered and chaotic regimes, and that most interesting behaviors are found between these extremes.

Polyworld's neural networks may be viewed as sets of difference equations (see Equation 3.1), optionally parameterized by the maximum synaptic weight as in Equation 3.2. Given this amenability to dynamical analysis, I apply the tools of dynamical systems theory to investigate how neural dynamics change over evolutionary time. I hypothesize an evolutionary trend in Polyworld's neural networks toward critical dynamics: criticality might foster the balance of integration and segregation required to produce the complex

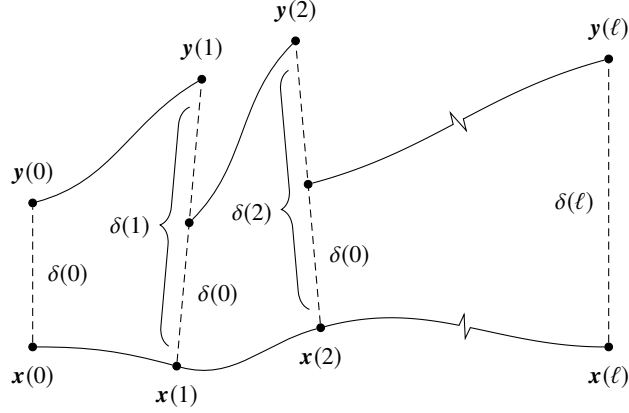


Figure 6.2: Numerical calculation of the maximal Lyapunov exponent. At each timestep t , the separation $\delta(t)$ between nearby trajectories $\mathbf{x}(t)$ and $\mathbf{y}(t)$ is rescaled to its initial value $\delta(0)$.

neural dynamics recognized in prior work.

6.2 Methods

One metric commonly used to characterize a dynamical system is the *maximal Lyapunov exponent* (MLE or λ), which measures the typical rate of divergence of two initially close trajectories:

$$\lambda = \lim_{t \rightarrow \infty} \lim_{\delta(0) \rightarrow 0} \frac{1}{t} \ln \frac{\delta(t)}{\delta(0)}. \quad (6.2)$$

The sign of the MLE is often taken as an indication of the system's overall dynamics—ordered ($\lambda < 0$), chaotic ($\lambda > 0$), or critical ($\lambda \approx 0$). Following the method described by [Sprott \(2003\)](#), the MLE of a discrete-time dynamical system may be calculated numerically by taking the logarithm of the separation δ between nearby trajectories $\mathbf{x}(t)$ and $\mathbf{y}(t)$:

$$\lambda = \frac{1}{\ell} \sum_{t=1}^{\ell} \ln \frac{\delta(t)}{\delta(0)}. \quad (6.3)$$

During the calculation process, it is important to ensure that the trajectories remain close to one another. If they diverge, one trajectory must be returned to the vicinity of the other along their line of separation. Conservatively, this may be done after every timestep as shown in [Figure 6.2](#).

To calculate the MLE of a Polyworld agent's neural network, I use a variation of the *in vitro* approach described in [Section 4.3](#) that facilitates the necessary rescaling of the separation between trajectories.

Synaptic weights are based on the neural anatomy recorded at agent death, and they are held constant during analysis. I begin by initializing the network randomly, setting all neural activations to random values drawn uniformly from $[0, 1]$. Next, I iterate the network for 100 timesteps using random inputs (uncorrelated sensory noise), driving the system into some arbitrary area of its state space. Finally, I iterate the network for 10,000 timesteps using quiescent inputs (sensory silence), allowing the system to reach an attractor.

At this point ($t = 0$), I create a copy of the network, slightly perturb the copied network’s neural activations, and measure the effect of this perturbation over time. To accomplish this, I take the original network’s activations as the vector $\mathbf{x}(0)$. I ignore the input neurons, since their activations are completely determined by environmental and physiological factors. I copy $\mathbf{x}(0)$ into $\mathbf{y}(0)$ and apply a random perturbation having magnitude $\delta(0) = 10^{-6}$. This results in two networks with slightly different activations—conceptually, two trajectories separated by $\delta(0)$. I then iterate both networks for a single timestep using quiescent inputs, yielding activation vectors $\mathbf{x}(1)$ and $\mathbf{y}(1)$ separated by $\delta(1) = \|\mathbf{y}(1) - \mathbf{x}(1)\|$.

The network’s response to the initial perturbation may be determined by comparing the two trajectories’ separation before and after the timestep. If $\delta(1) < \delta(0)$, the trajectories *converged*, representing a *negative* contribution to the MLE. If $\delta(1) > \delta(0)$, the trajectories *diverged*, representing a *positive* contribution to the MLE. Prior to performing the next iteration, the trajectories’ separation is rescaled to match the magnitude of the original perturbation by moving $\mathbf{y}(1)$ along the line between it and $\mathbf{x}(1)$. After $\ell = 100$ iterations, I calculate the MLE as in Equation 6.3. I repeat this entire process ten times using different initial states, and I report the agent’s MLE as the average of these ten values. It should be noted that—by using quiescent inputs during the calculation phase—this analysis characterizes the *intrinsic* dynamics of a network, not its dynamics *as driven by sensory input*.

Note from Equation 6.3 that, as the separation between trajectories shrinks to zero—a common occurrence when analyzing agents from the early portion of a simulation—the MLE diverges to negative infinity. To avoid this, I define a related metric called *state space expansion* (SSE or χ) which simply omits the logarithm, moving the critical value to unity:

$$\chi = \frac{1}{\ell} \sum_{t=1}^{\ell} \frac{\delta(t)}{\delta(0)}. \quad (6.4)$$

While the MLE has a theoretical range of $(-\infty, \infty)$, the SSE is only semi-infinite: $\chi \in [0, \infty)$. The SSE is thus better suited for the population-level analysis presented here.

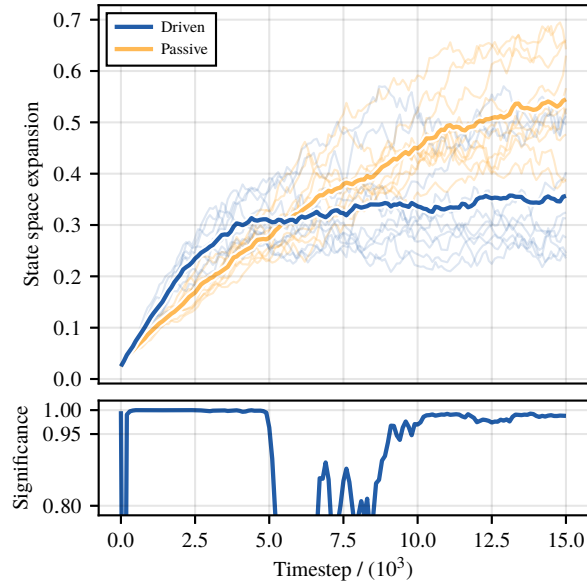


Figure 6.3: State space expansion vs. timestep for ten driven/passive pairs of [Legacy](#) simulations. Light lines represent population means for individual simulations, while heavy lines represent meta-means over all ten simulations. Significance is calculated as $1 - p$ for a two-tailed, paired Student’s t -test of driven and passive means.

6.3 Results

The evolutionary trend in the SSE of Polyworld’s neural networks is shown in [Figure 6.3](#). As expected, this metric increases over evolutionary time. The driven trend is significantly faster during the period of behavioral adaptation, though it plateaus relatively far from criticality, indicating that a successful survival strategy may be implemented using relatively ordered neural dynamics.

6.4 Discussion

To investigate neural dynamics in more detail, I use *bifurcation diagrams*, which depict the long-term behavior of a system in response to a varying parameter. In ordered regimes, a bifurcation diagram collapses to a single point (representing a stable fixed point) or a set of n points (representing a stable limit cycle of period n). Chaotic regimes appear as widely spread points, indicating the system’s long-term unpredictability. In keeping with similar work, I use the maximum synaptic weight $w_{\max} \in [0, 100]$ as the parameter in these diagrams. Recall that the actual value of w_{\max} is determined by the [MaxSynapseWeight](#) worldfile parameter and fixed globally at 8 for [Legacy](#) simulations.

To construct a bifurcation diagram for a given Polyworld agent’s neural network, I proceed in a manner similar to the SSE analysis described in Section 6.2. Synaptic weights are again held constant during analysis. However, before each calculation, weights are rescaled by the current value of w_{\max} , which is varied in increments of 0.01 from 0 to 100. Starting from a random initial state, I iterate the network for 100 timesteps with random inputs, then iterate for 10,000 timesteps with quiescent inputs. I plot the final activation of the neuron controlling the Turn action against the current value of w_{\max} . I repeat this process 100 times for each value of w_{\max} , yielding a total of roughly one million points. Additionally, I plot the SSE over the same range of w_{\max} values.

Typical bifurcation diagrams from early in a simulation (see Figure 6.4) indicate extremely ordered neural dynamics. Indeed, an agent from the initial population exhibits virtually no change in dynamics for varying w_{\max} . As the population evolves from this maximally rigid initial state, neural dynamics become more flexible. However, the long-term behavior of the system is still generally characterized by a single stable fixed point, with the SSE remaining relatively far from criticality.

Later in the same simulation (see Figure 6.5), bifurcation diagrams begin to show evidence of richer neural dynamics. A common feature is a Hopf bifurcation (accompanied by critical SSE) where the system’s single fixed point loses stability and is replaced by a period-2 limit cycle. Initially appearing at relatively large values of w_{\max} , these bifurcations tend to move leftward on the diagram over evolutionary time toward the actual value of $w_{\max} = 8$.

Toward the end of the simulation (see Figure 6.6), agents exhibit increasingly complicated neural dynamics, with bifurcation diagrams showing evidence of period-doubling bifurcations, quasiperiodicity, and chaos. The SSE regularly increases into the chaotic range ($\chi > 1$), with such regimes again tending to move leftward on the diagram over evolutionary time.

Note that these 12 diagrams represent a tiny fraction of the 12,419 agents that comprise the corresponding simulation’s entire historical population. They represent an even smaller proportion of the 137,740 agents observed across all ten driven simulations. Despite this, I have included these results—without comparison to the corresponding passive simulation—for two reasons. First, they demonstrate in detail the typical kinds of dynamics produced by agents’ neural networks. Second, they help to motivate the metric defined below, which provides a more complete characterization of neural dynamics.

An overarching theme in Figures 6.4, 6.5, and 6.6 is that rich neural dynamics initially appear at relatively large values of w_{\max} and subsequently move leftward on the diagram over evolutionary time. Thus, the trend

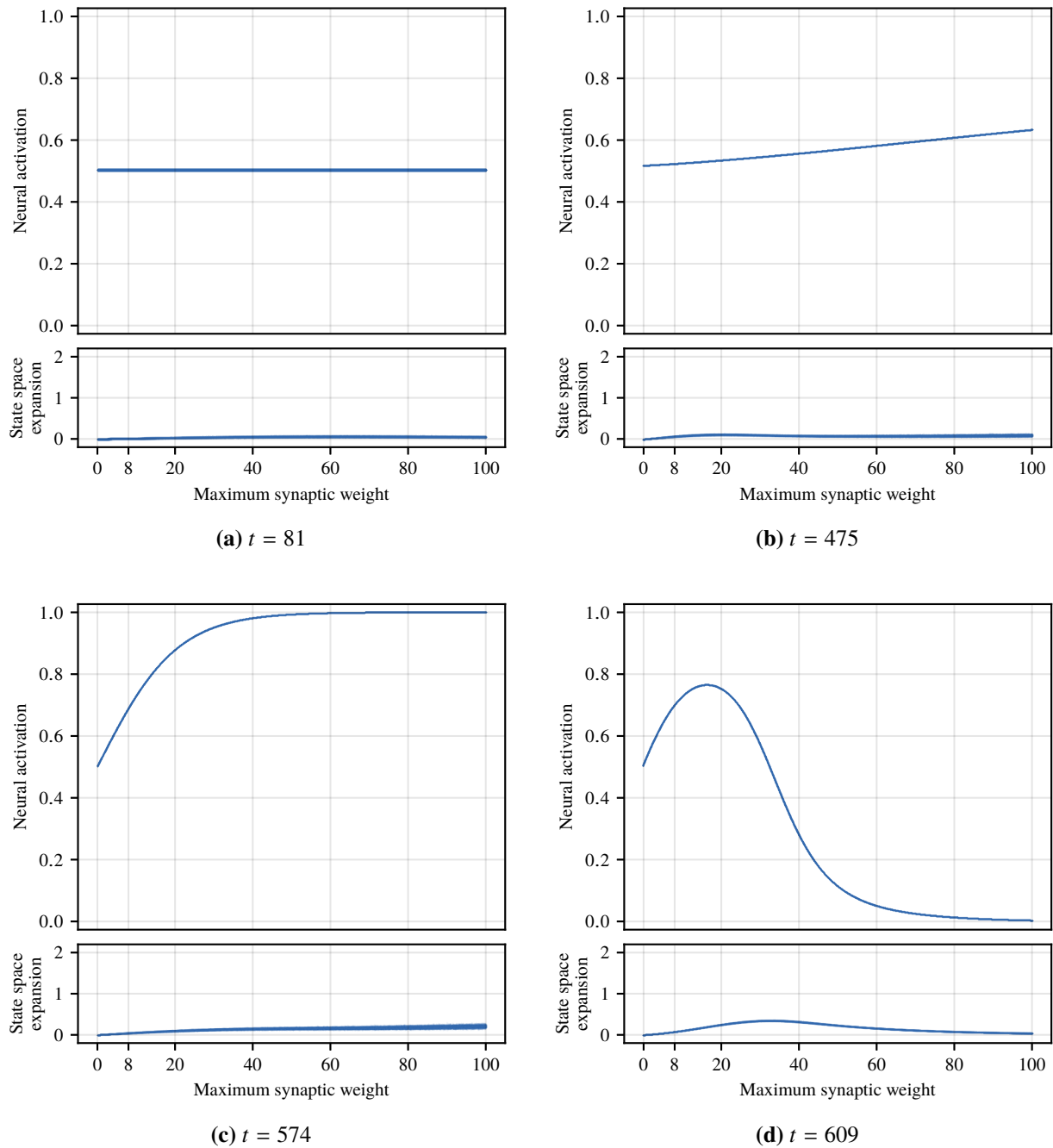


Figure 6.4: Typical bifurcation diagrams for agents from the early stage of a driven *Legacy* simulation, arranged in chronological order of the timestep t in which the agent died. Each diagram shows the activation of the neuron controlling the Turn action (top) and state space expansion (bottom) vs. maximum synaptic weight w_{\max} , with roughly one million points plotted for $w_{\max} \in [0, 100]$. During the simulation, the actual value of w_{\max} is fixed globally at 8.

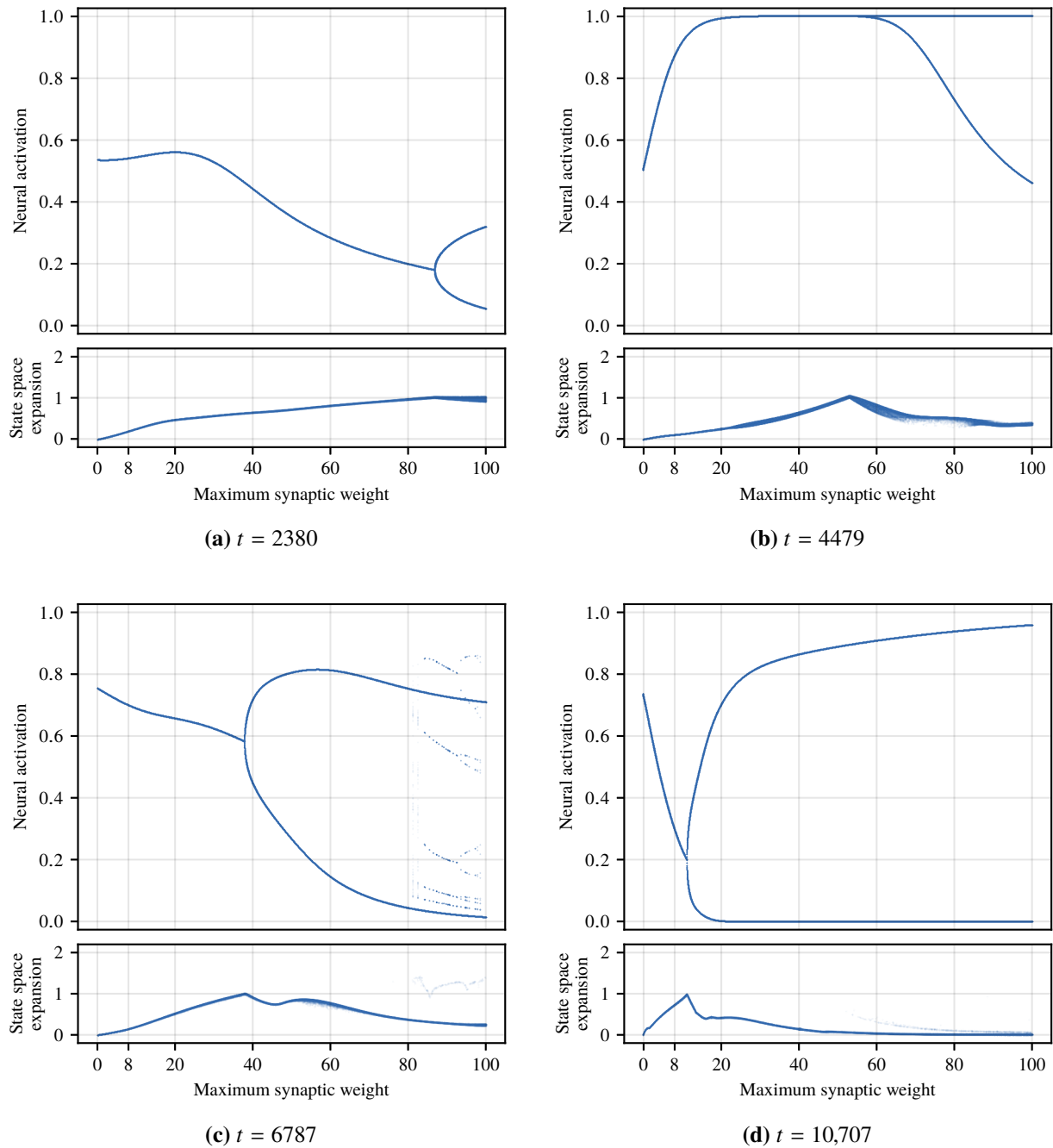


Figure 6.5: Typical bifurcation diagrams for agents from the middle stage of a driven *Legacy* simulation, arranged in chronological order of the timestep t in which the agent died. Each diagram shows the activation of the neuron controlling the Turn action (top) and state space expansion (bottom) vs. maximum synaptic weight w_{\max} , with roughly one million points plotted for $w_{\max} \in [0, 100]$. During the simulation, the actual value of w_{\max} is fixed globally at 8.

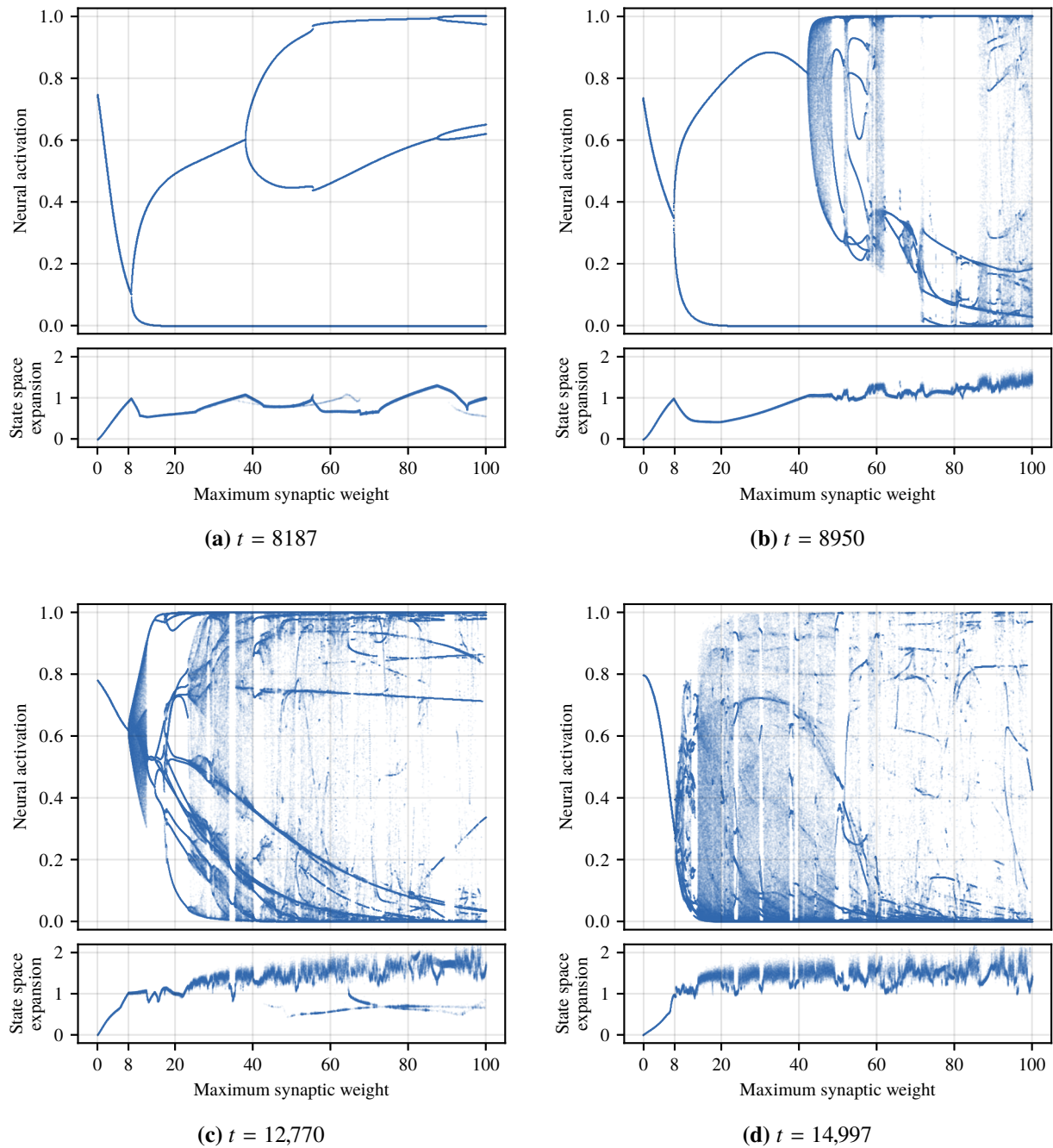


Figure 6.6: Typical bifurcation diagrams for agents from the late stage of a driven *Legacy* simulation, arranged in chronological order of the timestep t in which the agent died. Each diagram shows the activation of the neuron controlling the Turn action (top) and state space expansion (bottom) vs. maximum synaptic weight w_{\max} , with roughly one million points plotted for $w_{\max} \in [0, 100]$. During the simulation, the actual value of w_{\max} is fixed globally at 8.

toward critical dynamics identified in Figure 6.3 appears to be mediated by evolutionary tuning such that the onset of the chaotic regime occurs near the actual value of $w_{\max} = 8$.

I quantify this observation by measuring the *onset of criticality* (OOC or ω) as the least value of w_{\max} for which the SSE exceeds some threshold value χ^* :

$$\omega = \min\{w_{\max} : \chi(w_{\max}) \geq \chi^*\}, \quad (6.5)$$

where $\chi(w_{\max})$ denotes a parameterized version of the SSE in which synaptic weights are rescaled by a particular value of w_{\max} . The OOC thus provides a way to quantify the apparent leftward movement of rich dynamics on the bifurcation diagrams over evolutionary time.

Like the SSE, the OOC provides a high-level characterization of a Polyworld agent's neural network. However, the OOC takes into account much more of the neural dynamics that are theoretically possible, since it investigates a range of w_{\max} values. Intuitively, if dynamics are being tuned toward criticality, the OOC should tend toward the actual value of $w_{\max} = 8$ over evolutionary time.

Using the method described in Section 6.2, I calculate the SSE for each agent while varying w_{\max} from 0 to 1,000,000, stopping when the SSE exceeds the threshold value. To make this analysis computationally feasible, I consider only integral values of w_{\max} , using increasingly coarse-grained resolution as larger values of w_{\max} are considered. The fine-grained analysis demonstrated in Figures 6.4, 6.5, and 6.6—varying w_{\max} in increments of 0.01—becomes prohibitively time-consuming when applied to ten pairs of simulations, each consisting of tens of thousands of agents.

The threshold value χ^* must be chosen with care. Even at arbitrarily large values of w_{\max} , some agents' neural dynamics remain highly ordered. If χ^* is chosen too close to unity, these agents' SSEs will never exceed it, and the OOC must be taken as infinite. However, if χ^* is chosen too far below unity (in an attempt to avoid this problem), the OOC will fail to capture its intended meaning. Despite this difficulty, the evolutionary trend in the OOC appears stable over a range of thresholds. I report $\chi^* = 0.5$ here—well within the ordered regime, but larger than the typical SSE observed in driven simulations.

The evolutionary trend in the OOC of Polyworld's neural networks is shown in Figure 6.7. Since the OOC may be infinite, this figure is generated slightly differently from similar figures elsewhere in this work: values are aggregated using the median rather than the mean, and significance is calculated using a Wilcoxon signed-rank test rather than a paired Student's t -test. As expected, the OOC decreases over evolutionary

time, mirroring the leftward movement of rich dynamics on the bifurcation diagrams. The driven trend is significantly faster during the period of behavioral adaptation. Like previously reported metrics, it plateaus relatively far from its critical value.

As shown in Figure 6.8, the relationship between the SSE and the OOC makes clear that the full range of neural dynamics—from ordered to critical to chaotic—is only possible when the OOC is less than the actual $w_{\max} = 8$. Adaptive dynamics thus appear to be achieved via evolutionary tuning such that networks are appropriately sensitive to perturbation.

6.5 Conclusion

In Polyworld, evolution selects for critical neural dynamics. The primary evidence for this claim comes from the evolutionary trend in state space expansion. During the period of behavioral adaptation, this trend is significantly faster in driven simulations than in passive simulations, suggesting a correlation between critical dynamics and adaptive behavior.

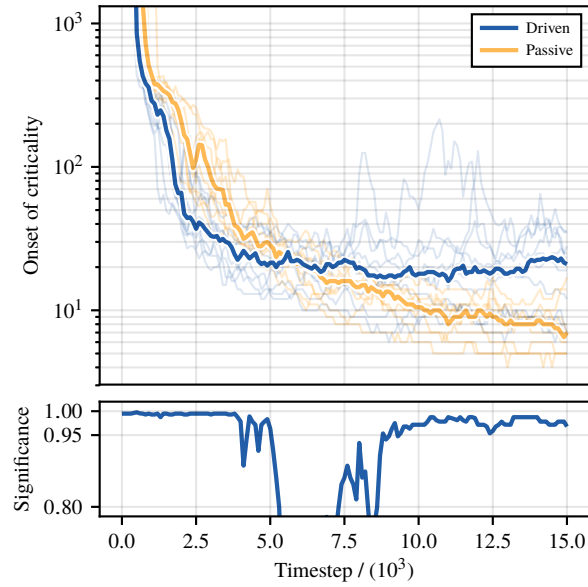


Figure 6.7: Onset of criticality vs. timestep for ten driven/passive pairs of *Legacy* simulations. Light lines represent population medians for individual simulations, while heavy lines represent meta-medians over all ten simulations. Significance is calculated as $1 - p$ for a two-tailed Wilcoxon signed-rank test of driven and passive medians.

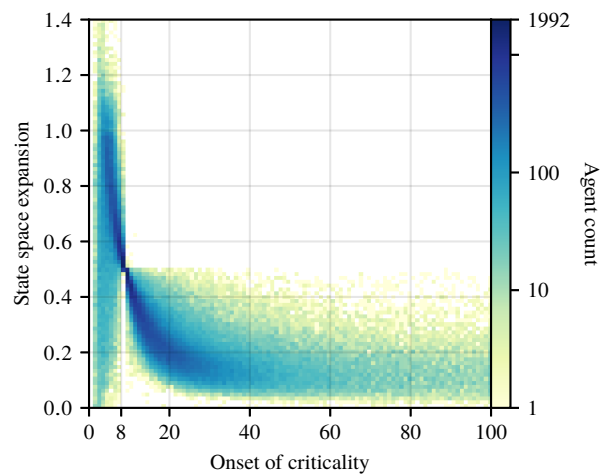


Figure 6.8: State space expansion vs. onset of criticality for ten driven/passive pairs of *Legacy* simulations.

Chapter 7

Analysis of Information Processing

This chapter presents an analysis of Polyworld’s neural networks using the tools of information theory. Such an approach emphasizes a network’s information-processing capabilities—its capacity to store, transmit, and modify information. For a more comprehensive introduction to information theory, see [Cover and Thomas \(2006\)](#).

7.1 Foundations

In his seminal work, [Shannon \(1948\)](#) introduced information theory as a framework for investigating the theoretical limits of data compression and transmission in a communication channel. Since then, the tools of information theory have been applied well beyond their original field to disciplines such as statistics, physics, computer science, and economics, among many others.

The central measure of information theory is the Shannon entropy H . To motivate its definition, consider the information content h of an event E that occurs with probability $\Pr(E)$. Upon its observation, how much information does this event provide? We would like $h(E)$ to satisfy the following criteria:

1. $h(E) \geq 0$: observing an event never reduces the amount of information we have.
2. $h(E) = 0$ for $\Pr(E) = 1$: an event that occurs with absolute certainty is not informative.
3. $h(E)$ is monotonically decreasing over $\Pr(E)$: less likely events are more informative. (For this reason, the information content of an event is also referred to as its *surprisal*.)
4. $h(E, F) = h(E) + h(F)$ for independent events, which carry no information about each other.

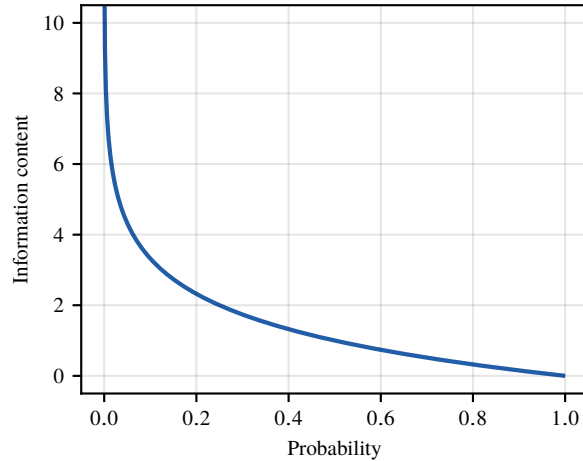


Figure 7.1: Information content vs. probability. Information content is reported in bits (i.e., using the base-2 logarithm in Equation 7.1).

Given these criteria, information content must be of the form

$$h(E) = -\log \Pr(E). \quad (7.1)$$

The base of the logarithm determines the units of this quantity, typically *bits* for the base-2 logarithm or *nats* for the natural logarithm. Note from Figure 7.1 that this formulation of information content satisfies the first three criteria listed above.

Joint information content (i.e., the information provided by two events E and F occurring together) is defined by simple extension of Equation 7.1:

$$h(E, F) = -\log \Pr(E, F). \quad (7.2)$$

It is instructive to decompose this quantity into its *marginal* and *conditional* components using the chain rule of conditional probability $\Pr(E, F) = \Pr(E) \Pr(F | E)$:

$$\begin{aligned} h(E, F) &= -\log \Pr(E, F) \\ &= -\log \Pr(E) \Pr(F | E) \\ &= -\log \Pr(E) - \log \Pr(F | E) \\ &= h(E) + h(F | E), \end{aligned} \quad (7.3)$$

	W	\bar{W}	Marginal
S	0.09	0.01	0.10
\bar{S}	0.16	0.74	0.90
Marginal	0.25	0.75	1.00

Table 7.1: Joint probability distribution for events S (it is snowing) and W (it is winter).

which emphasizes that the joint information content of E and F may be decomposed into

- $h(E)$, the information provided by E occurring alone, and
- $h(F | E)$, the additional information provided by F occurring when we already know that E holds.

If E and F are independent, the conditional probability simplifies to $\Pr(F | E) = \Pr(F)$. This yields

$$\begin{aligned}
 h(E, F) &= -\log \Pr(E, F) \\
 &= -\log \Pr(E) \Pr(F | E) \\
 &= -\log \Pr(E) \Pr(F) \\
 &= -\log \Pr(E) - \log \Pr(F) \\
 &= h(E) + h(F),
 \end{aligned} \tag{7.4}$$

which satisfies the fourth criterion above. Note that, by symmetry,

$$\begin{aligned}
 h(E, F) &= h(F, E) \\
 &= h(F) + h(E | F).
 \end{aligned} \tag{7.5}$$

As a concrete example, consider events S (it is snowing) and W (it is winter), with the joint probability distribution shown in Table 7.1. Taken in isolation, these events provide $h(S) = 3.322$ b and $h(W) = 2$ b of information. Taken together, they do not provide $h(S) + h(W) = 5.322$ b, but rather $h(S, W) = 3.474$ b of information, since they are dependent. I.e., if we know that it is snowing, we are fairly confident that it is also winter, and so W only provides an additional $h(W | S) = 0.152$ b of information, as opposed to the $h(W) = 2$ b it provides in isolation.

Thus, to arrive at the joint information content, the sum of two events' marginal information contents

must be reduced by the amount of information $i(S; W)$ that is shared between them:

$$h(S, W) = h(S) + h(W) - i(S; W). \quad (7.6)$$

Solving for $i(S; W)$,

$$\begin{aligned} i(S; W) &= h(S) + h(W) - h(S, W) \\ &= 3.322 \text{ b} + 2 \text{ b} - 3.474 \text{ b} \\ &= 1.848 \text{ b}. \end{aligned} \quad (7.7)$$

This quantity captures the *shared information content* between the two events (generally referred to as their *pointwise mutual information*). Recalling that $h(S, W) = h(S) + h(W | S)$,

$$\begin{aligned} i(S; W) &= h(S) + h(W) - h(S, W) \\ &= h(S) + h(W) - [h(S) + h(W | S)] \\ &= h(W) - h(W | S). \end{aligned} \quad (7.8)$$

This formulation emphasizes that there is a reduction in the information content of W given our knowledge of S (or, to restate from above, that snow usually implies winter). Note that, by symmetry, we could have substituted $h(S, W) = h(W) + h(S | W)$:

$$\begin{aligned} i(S; W) &= h(S) + h(W) - h(S, W) \\ &= h(S) + h(W) - [h(W) + h(S | W)] \\ &= h(S) - h(S | W) \\ &= i(W; S). \end{aligned} \quad (7.9)$$

Pointwise mutual information is thus a symmetric measure of co-occurrence between two events. Stating

its definition more fundamentally,

$$\begin{aligned}
 i(E; F) &= h(E) + h(F) - h(E, F) \\
 &= -\log \Pr(E) - \log \Pr(F) + \log \Pr(E, F) \\
 &= \log \frac{\Pr(E, F)}{\Pr(E) \Pr(F)}. \tag{7.10}
 \end{aligned}$$

If E and F are informative about one another (i.e., if they are expected to co-occur), then the pointwise mutual information is positive, as we saw in the previous example. If E and F are independent, we have $\Pr(E, F) = \Pr(E) \Pr(F)$, and Equation 7.10 yields a pointwise mutual information of zero. As emphasized by Equation 7.4, this independence also means that the joint information content simplifies to a sum of marginal information contents.

It is also possible for pointwise mutual information to become negative. This occurs in cases where E and F are misinformative about one another (i.e., if they are not expected to co-occur). E.g., for events S (it is snowing) and \overline{W} (it is not winter),

$$\begin{aligned}
 i(S; \overline{W}) &= h(S) + h(\overline{W}) - h(S, \overline{W}) \\
 &= 3.322 \text{ b} + 0.415 \text{ b} - 6.644 \text{ b} \\
 &= -2.907 \text{ b}. \tag{7.11}
 \end{aligned}$$

In this case, there is an increase in the information content of \overline{W} given our knowledge of S . I.e., the conditional information content $h(\overline{W} | S)$ exceeds the marginal information content $h(\overline{W})$. Stated more simply, we are surprised when it snows in spring, summer, or fall.

Now instead of individual events, consider a discrete random variable x . Its entropy H is defined as the expected information content over all possible outcomes E , yielding a measure of the variable's overall unpredictability:

$$H(x) = - \sum_{E \in x} \Pr(E) \log \Pr(E). \tag{7.12}$$

We may similarly extend the other definitions above to arrive at the *joint entropy*, *conditional entropy*, and

mutual information of two discrete random variables x and y :

$$H(x, y) = - \sum_{E \in x} \sum_{F \in y} \Pr(E, F) \log \Pr(E, F), \quad (7.13)$$

$$H(x | y) = - \sum_{E \in x} \sum_{F \in y} \Pr(E, F) \log \Pr(E | F), \quad (7.14)$$

$$I(x; y) = \sum_{E \in x} \sum_{F \in y} \Pr(E, F) \log \frac{\Pr(E, F)}{\Pr(E) \Pr(F)}. \quad (7.15)$$

Joint entropy may also be expressed as

$$\begin{aligned} H(x, y) &= - \sum_{E \in x} \sum_{F \in y} \Pr(E, F) \log \Pr(E, F) \\ &= - \sum_{E \in x} \sum_{F \in y} \Pr(E, F) \log \Pr(E) \Pr(F | E) \\ &= - \sum_{E \in x} \sum_{F \in y} \Pr(E, F) [\log \Pr(E) + \log \Pr(F | E)] \\ &= - \sum_{E \in x} \sum_{F \in y} \Pr(E, F) \log \Pr(E) - \sum_{E \in x} \sum_{F \in y} \Pr(E, F) \log \Pr(F | E) \\ &= - \sum_{E \in x} \Pr(E) \log \Pr(E) - \sum_{E \in x} \sum_{F \in y} \Pr(E, F) \log \Pr(F | E) \\ &= H(x) + H(y | x). \end{aligned} \quad (7.16)$$

By symmetry,

$$\begin{aligned} H(x, y) &= H(y, x) \\ &= H(y) + H(x | y). \end{aligned} \quad (7.17)$$

Likewise, mutual information may be expressed as

$$\begin{aligned} I(x; y) &= \sum_{E \in x} \sum_{F \in y} \Pr(E, F) \log \frac{\Pr(E, F)}{\Pr(E) \Pr(F)} \\ &= \sum_{E \in x} \sum_{F \in y} \Pr(E, F) [\log \Pr(E, F) - \log \Pr(E) - \log \Pr(F)] \\ &= H(x) + H(y) - H(x, y). \end{aligned} \quad (7.18)$$

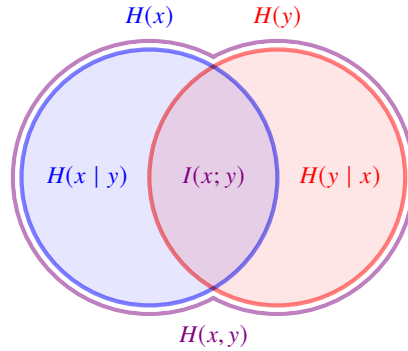


Figure 7.2: Relationships among information-theoretic metrics. Marginal, joint, and conditional entropy and mutual information are shown for dependent random variables x and y .

These relationships are summarized in Figure 7.2. Unlike the information content–based quantities defined previously, these entropy-based quantities are never negative. Intuitively, while *specific* (per-event) observations of a random variable may be misinformative, these observations can never be misinformative *on average*. At worst, a random variable has zero entropy (as in the case of a two-headed coin toss, whose result provides no information). If x and y are independent, there is no overlap between $H(x)$ and $H(y)$ in Figure 7.2, and so we have

$$H(x | y) = H(x), \quad (7.19)$$

$$H(y | x) = H(y), \quad (7.20)$$

$$I(x; y) = 0, \quad (7.21)$$

$$H(x, y) = H(x) + H(y). \quad (7.22)$$

How might these information-theoretic tools be applied to the analysis of Polyworld’s neural networks? In Chapter 6, I demonstrated that neural dynamics are tuned toward criticality over evolutionary time. Related studies of other computational systems have suggested that this “edge of chaos” is an optimal dynamical regime for supporting information storage, transmission, and modification. To investigate this more directly, I quantify the information-processing capabilities of Polyworld’s neural networks using the entropy-based metrics defined below. I hypothesize an evolutionary trend toward increased information storage, transmission, and modification. Such a trend would agree with the intuition that intelligence—recognized here as the production of adaptive behavior—requires effective information processing. Indeed, this intuition is supported by a wide variety of data—neural (Vicente et al., 2011), physiological (Faes

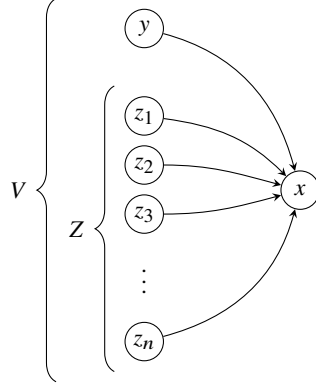


Figure 7.3: Causal relationships in a neural network in the neighborhood of a particular postsynaptic neuron x . The set of presynaptic neurons V may be separated into a particular presynaptic neuron y and all other presynaptic neurons $Z = V \setminus \{y\} = \{z_1, z_2, z_3, \dots, z_n\}$.

and Porta, 2014), sensorimotor (Lungarella and Sporns, 2006), social (Oka and Ikegami, 2013), financial (Sandoval, 2014), and computational (Lizier et al., 2010).

7.2 Methods

To investigate information processing in Polyworld’s neural networks, I use the framework developed by Lizier et al. (2014) and realized in the Java Information Dynamics Toolkit (JIDT), a software package for analyzing the information dynamics of distributed computational systems (Lizier, 2014). Neural states (i.e., vectors of neural activations) are considered as multivariate random variables. Each neuron is analyzed in the context of its own “causal neighborhood”—consisting of itself and all of its presynaptic neurons—as indicated in Figure 7.3.

In Lizier et al.’s (2014) framework, information storage is quantified by the *active information storage* (see Figure 7.4), defined as the mutual information between the current state of a neuron x and its recent history (based on some embedding length k):

$$A_x = I\left(\mathbf{x}^{(k)}(t-1); x(t)\right). \quad (7.23)$$

Information transmission is quantified by various formulations of the transfer entropy (Schreiber, 2000). The simplest formulation is the *apparent transfer entropy* (see Figure 7.5a) between a particular presynaptic

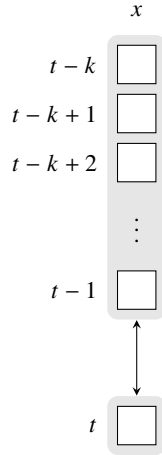


Figure 7.4: Active information storage: the mutual information between the last k states of neuron x and its current state. See Figure 7.3.

neuron y and postsynaptic neuron x :

$$T_{y \rightarrow x} = I\left(y(t-1); x(t) \mid \mathbf{x}^{(k)}(t-1)\right). \quad (7.24)$$

The *complete transfer entropy* (see Figure 7.5b) conditions on x 's other presynaptic neurons Z :

$$T_{y \rightarrow x|Z} = I\left(y(t-1); x(t) \mid \mathbf{x}^{(k)}(t-1), \mathbf{Z}(t-1)\right). \quad (7.25)$$

In general, x may be influenced by many of its presynaptic neurons synergistically. The complete transfer entropy is capable of detecting y 's contribution to such synergies, whereas the apparent transfer entropy is not. Meanwhile, the apparent transfer entropy is capable of detecting information transmitted from y to x that is redundantly carried in Z , whereas the complete transfer entropy is not. It should be noted that the calculation of complete transfer entropy can be quite computationally intensive; in the results reported here, I limit this calculation to a maximum of 250 randomly selected synapses per agent.

The *collective transfer entropy* (see Figure 7.6) quantifies information transmission due to all of x 's presynaptic neurons V considered jointly:

$$T_{V \rightarrow x} = I\left(\mathbf{V}(t-1); x(t) \mid \mathbf{x}^{(k)}(t-1)\right). \quad (7.26)$$

Note that the apparent and complete transfer entropy are measured at an *individual* synapse. In contrast,

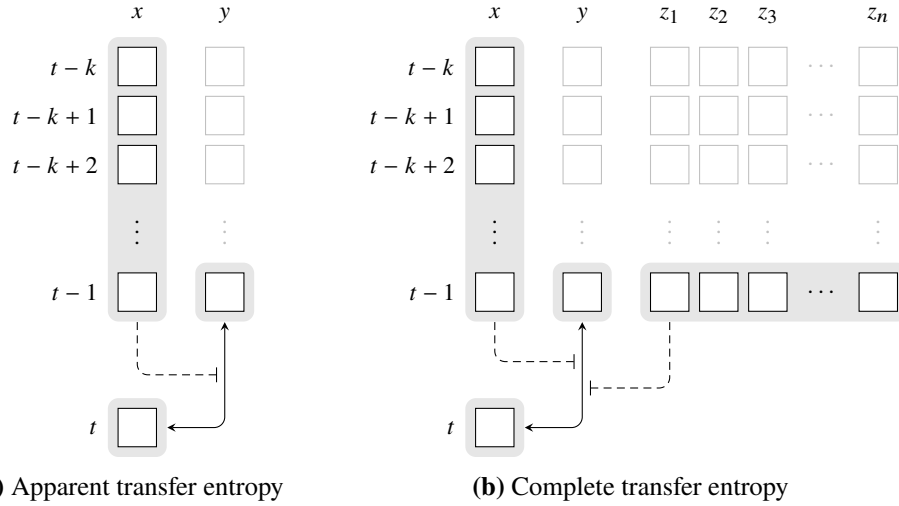


Figure 7.5: Apparent transfer entropy (left): the mutual information between the last state of presynaptic neuron y and the current state of postsynaptic neuron x , conditioned on the last k states of x . Complete transfer entropy (right): the mutual information between the last state of presynaptic neuron y and the current state of postsynaptic neuron x , conditioned on the last k states of x and on the last state of all other presynaptic neurons $Z = \{z_1, z_2, z_3, \dots, z_n\}$. See Figure 7.3.

the collective transfer entropy is measured over *all* synapses for a particular postsynaptic neuron, making it more consistent with the other metrics identified here. Furthermore, it detects both synergistic and redundant information transmission, overcoming the previously noted deficiencies of the apparent and complete transfer entropy. For these reasons, I use collective transfer entropy as the preferred formulation for reporting information transmission.

In much the same way that the entropy of a random variable represents the average information content of *specific* events, the metrics introduced above may be reformulated to quantify *local* information storage and transmission. These values are specific to the neural states exhibited at a particular point in time. Local versions of the active information storage and apparent transfer entropy are defined as

$$a_x(t) = i(\mathbf{x}^{(k)}(t-1); x(t)), \quad (7.27)$$

$$\tau_{y \rightarrow x}(t) = i(y(t-1); x(t) | \mathbf{x}^{(k)}(t-1)), \quad (7.28)$$

where i is the pointwise mutual information, applied here to specific states of x and y at a given time.

Unlike their corresponding nonlocal formulations, these quantities may be either positive or negative, indicating whether particular states are informative or misinformative about one another. When reporting

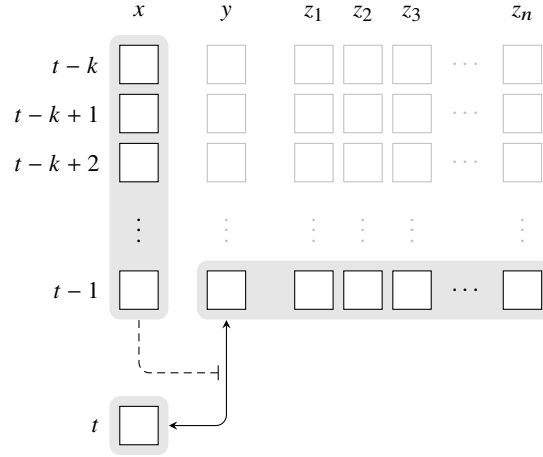


Figure 7.6: Collective transfer entropy: the mutual information between the last state of all presynaptic neurons $V = \{y, z_1, z_2, z_3, \dots, z_n\}$ and the current state of postsynaptic neuron x , conditioned on the last k states of x . See Figure 7.3.

population-level results, I first average these local values, collapsing over time and “space” (i.e., neurons or synapses) in order to arrive at a single value for each agent. I then average these per-agent values, yielding a single value that characterizes the entire population at a particular point in evolutionary time.

Information modification is quantified by the *separable information*, defined locally as the information gained from separate observation of each of x ’s causal information sources (i.e., each neuron in x ’s causal neighborhood, including x itself). This is simply the sum of x ’s local active information storage and all local apparent transfer entropies for its presynaptic neurons V :

$$s_x(t) = a_x(t) + \sum_{v_i \in V} \tau_{v_i \rightarrow x}(t). \quad (7.29)$$

This formulation is known to double-count portions of the information carried in the system. While it is thus not strictly an information-theoretic measure, it has proved to be a powerful heuristic for detecting information modification events.

Local separable information is *positive* when x ’s causal information sources (considered in isolation) are overall informative regarding x ’s next state, indicating a *trivial* information modification event. It is *negative* when these sources are overall misinformative, indicating a *nontrivial* information modification event. I

consider these two cases separately when collapsing local values:

$$S_x^+ = \frac{1}{\ell} \sum_{t=1}^{\ell} \max\{s(x, t), 0\}, \quad (7.30)$$

$$S_x^- = -\frac{1}{\ell} \sum_{t=1}^{\ell} \min\{s(x, t), 0\}. \quad (7.31)$$

All of these calculations require estimation of joint probability distributions in a potentially quite high-dimensional space. A significant amount of data—in the form of neural activation time series—must be provided in order to construct these distributions. As discussed in Chapter 4, time series collected *in vivo* are insufficient for this purpose. I instead use the previously validated *in vitro* approach of generating time series by exposing agents’ neural networks to uncorrelated noise while holding synaptic weights constant at the values recorded at agent death. Enough data is generated to provide stable results given the selection of a generous but practicable embedding length $k = 10$.

For each agent, I generate ten separate time series, each starting from a random initial state and then iterated for 1020 timesteps. I discard an initial transient of ten timesteps. An additional ten timesteps are consumed by embedding, leaving 1000 timesteps per time series (10,000 timesteps per agent) of usable data for analysis. Mutual information is estimated using a best-of-breed solution for continuous data based on nearest-neighbor search (Kraskov et al., 2004). In JIDT, this estimator makes use of the natural logarithm; all information-theoretic quantities are thus reported in nats.

7.3 Results

Evolutionary trends in the active information storage; apparent, complete, and collective transfer entropy; and positive and negative separable information of Polyworld’s neural networks are shown in Figures 7.7, 7.8, 7.9, and 7.10. As expected, active information storage and separable information (both positive and negative) increase over evolutionary time. The driven trends are significantly faster during the period of behavioral adaptation. Typical values of active information storage have very small magnitudes, indicating that little “memory” is required to produce adaptive behavior. In these simulations, it is sufficient for agents’ behavior to be governed almost completely by reaction to environmental stimuli. Meanwhile, the trends in positive and negative separable information suggest evolutionary selection for increased trivial and nontrivial information modification during the period of behavioral adaptation.

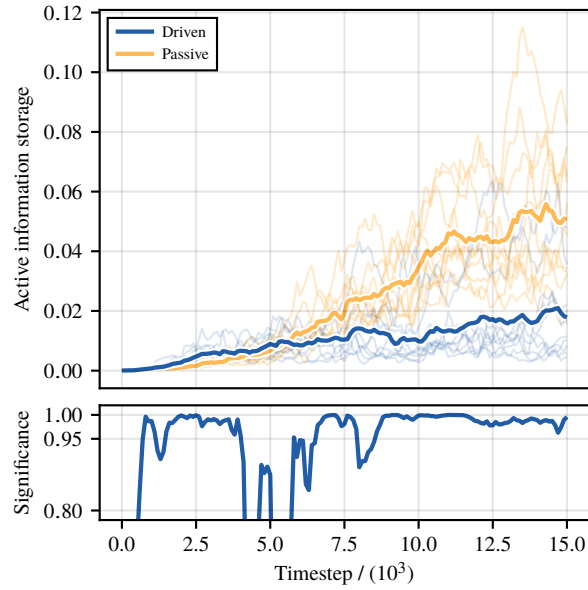


Figure 7.7: Active information storage vs. timestep for ten driven/passive pairs of *Legacy* simulations. Light lines represent population means for individual simulations, while heavy lines represent meta-means over all ten simulations. Significance is calculated as $1 - p$ for a two-tailed, paired Student's t -test of driven and passive means.

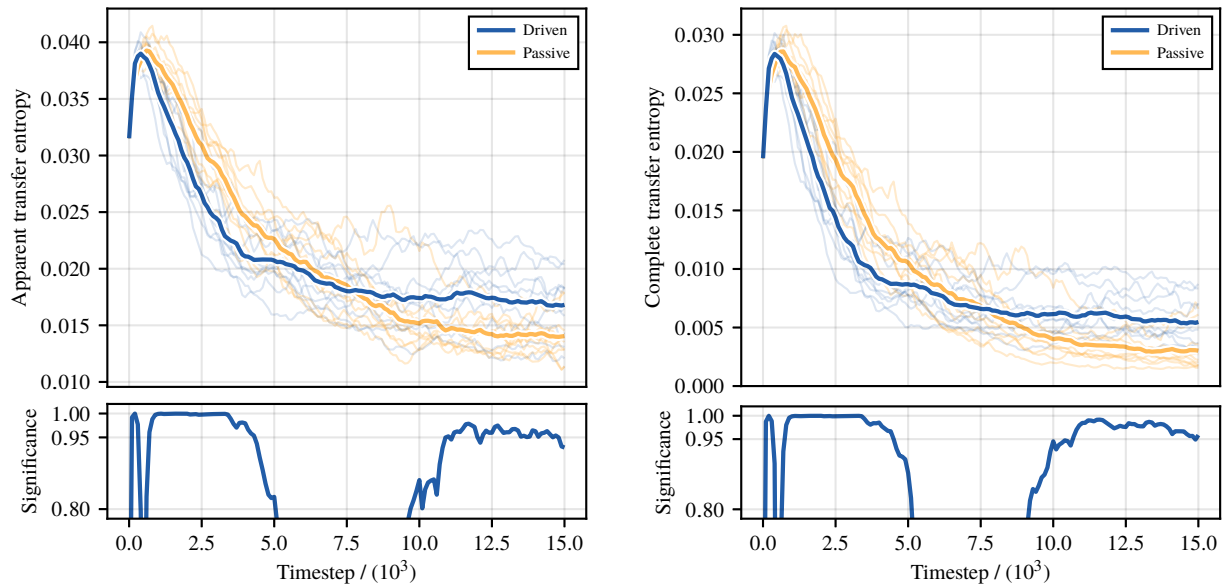


Figure 7.8: Apparent transfer entropy (left) and complete transfer entropy (right) vs. timestep for ten driven/passive pairs of *Legacy* simulations. Light lines represent population means for individual simulations, while heavy lines represent meta-means over all ten simulations. Significance is calculated as $1 - p$ for a two-tailed, paired Student's t -test of driven and passive means.

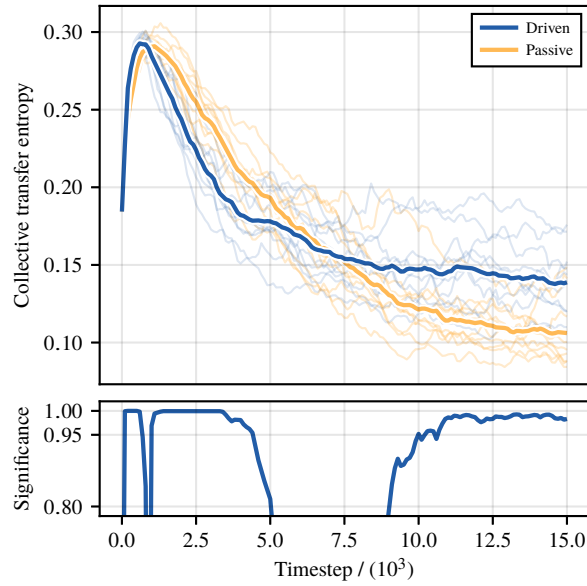


Figure 7.9: Collective transfer entropy vs. timestep for ten driven/passive pairs of *Legacy* simulations. Light lines represent population means for individual simulations, while heavy lines represent meta-means over all ten simulations. Significance is calculated as $1 - p$ for a two-tailed, paired Student's t -test of driven and passive means.

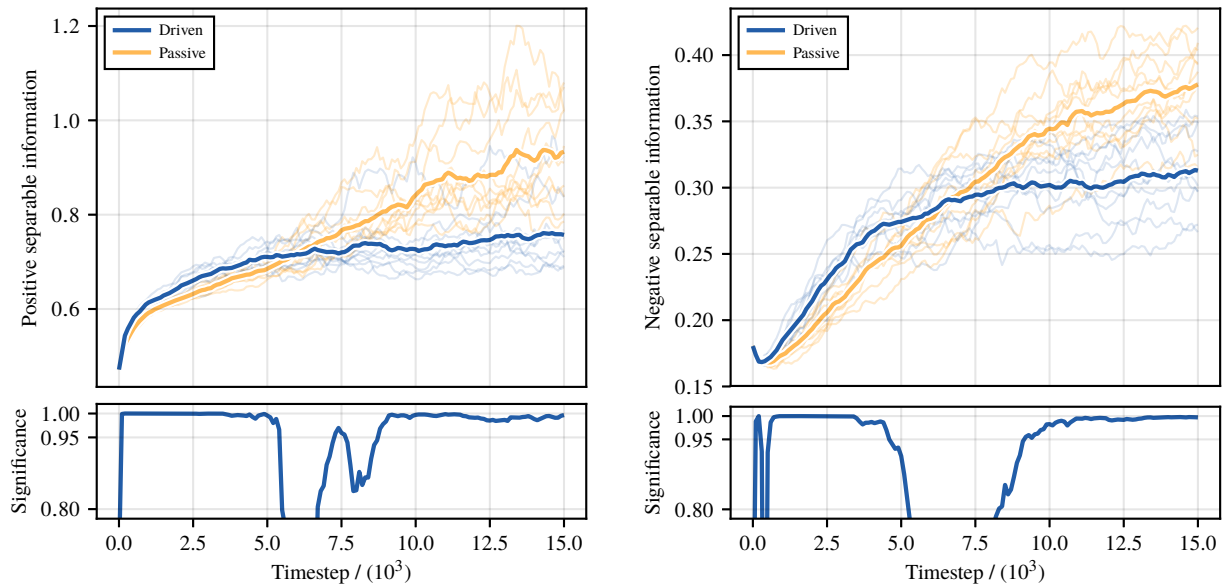


Figure 7.10: Positive separable information (left) and negative separable information (right) vs. timestep for ten driven/passive pairs of *Legacy* simulations. Light lines represent population means for individual simulations, while heavy lines represent meta-means over all ten simulations. Significance is calculated as $1 - p$ for a two-tailed, paired Student's t -test of driven and passive means.

In its synapse-specific formulations (see Figure 7.8), transfer entropy initially increases sharply during the first thousand or so timesteps, then decreases throughout the remainder of the simulation. This is not necessarily unexpected: it could simply indicate that information transmission becomes more synergistic and more redundant over evolutionary time, and is thus captured by neither the apparent nor the complete transfer entropy. However, collective transfer entropy exhibits a similar trend (see Figure 7.9), indicating that—contrary to expectations—there is indeed a significant driven evolutionary trend toward decreased information transmission.

7.4 Discussion

To help explain this unexpected evolutionary trend in collective transfer entropy, I investigate its relationship with a more fundamental information-theoretic metric. Specifically, I examine the *differential entropy*, a somewhat naive extension of the Shannon entropy that may be applied to continuous data:

$$H(x) = - \int f(x) \log f(x) dx. \quad (7.32)$$

Here a probability density function f is used, rather than discrete probability masses as in Equation 7.12. In JIDT, this quantity is estimated as in [Kozachenko and Leonenko \(1987\)](#) and reported in nats.

When applied to Polyworld’s neural activation time series (see Figure 7.11), differential entropy exhibits an evolutionary trend similar to that of collective transfer entropy—peaking early in a simulation, then decreasing during its remainder. Furthermore, Figure 7.12a suggests that, on an individual level, there is a direct relationship between these two metrics: the amount of information *transferred* to a typical postsynaptic neuron is limited by the amount of information *available* in a typical presynaptic neuron.

In turn, the trend toward decreased differential entropy appears to be mediated by evolutionary tuning toward segregation of network function. Previously reported trends in complexity and small-worldness (see Figures 4.4 and 5.5) have indicated that, over evolutionary time, the information contained in a typical network tends to “spread out,” with different portions of the network becoming specialized for different aspects of information processing. Networks that are more clustered (i.e., more locally efficient) tend to carry less information in each neuron, yielding the inverse relationship depicted in Figure 7.12b. It is unclear as yet why information storage and modification are not similarly limited by the evolutionary trend toward decreased differential entropy. This question should be addressed in future work.

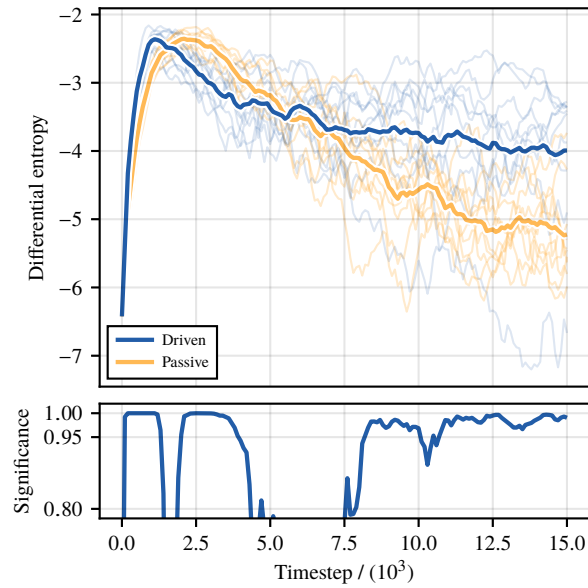
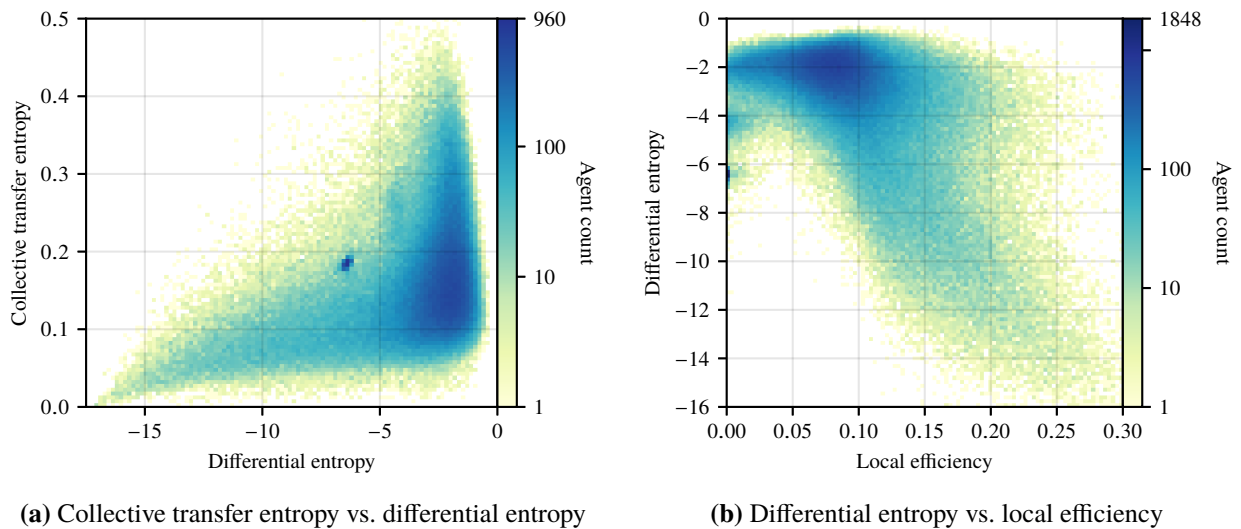


Figure 7.11: Differential entropy vs. timestep for ten driven/passive pairs of Legacy simulations. Light lines represent population means for individual simulations, while heavy lines represent meta-means over all ten simulations. Significance is calculated as $1 - p$ for a two-tailed, paired Student's t -test of driven and passive means.



(a) Collective transfer entropy vs. differential entropy

(b) Differential entropy vs. local efficiency

Figure 7.12: Relationships between differential entropy and other metrics for ten driven/passive pairs of Legacy simulations.

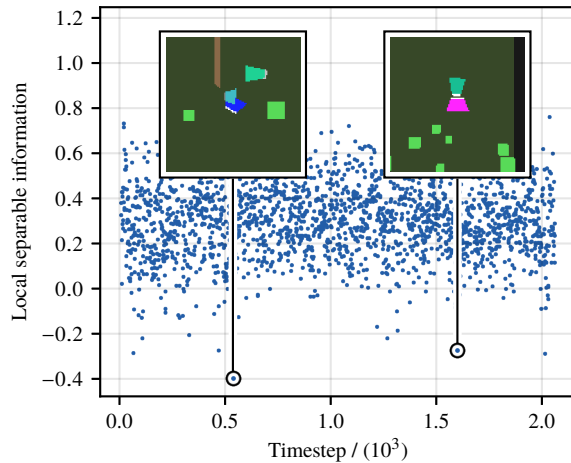


Figure 7.13: Local separable information vs. timestep for an agent from the late stage of a driven *Legacy* simulation. Inset images show an overhead view of the agent at the timestep corresponding to a particular point. Contrast has been enhanced.

A more detailed investigation of the local information dynamics of behaving agents could be helpful in this regard. As in related work, preliminary studies have demonstrated the utility of the local separable information in recognizing information modification events. Specifically, negative separable information (i.e., nontrivial information modification) appears to be correlated with agent-agent interaction, as shown in Figure 7.13.

7.5 Conclusion

In Polyworld, evolution selects for effective neural information processing. The primary evidence for this claim comes from the evolutionary trends in active information storage and separable information (both positive and negative). During the period of behavioral adaptation, these trends are significantly faster in driven simulations than in passive simulations, suggesting a correlation between effective information processing and adaptive behavior.

Chapter 8

Synthesis

This chapter presents a synthesis of the structural, dynamical, and information-processing analyses presented in the previous three chapters.

8.1 Introduction

I have demonstrated that Polyworld’s neural networks experience selection for small-world structure, critical dynamics, and effective information processing over evolutionary time. These trends are statistically significant relative to neutral evolution, suggesting correlations with adaptive behavior. To synthesize these results—and to address the main goal of this dissertation—I attempt to characterize the relationship between neural properties and adaptive behavior more directly.

How might such an investigation be conducted? Given some individual-level measure of evolutionary fitness, we could calculate its value for each agent, then look for correlations with neural properties of interest. Recall, however, that Polyworld uses natural selection—its evolutionary algorithm incorporates no well-defined fitness function that could be used in such a study. We could invent a measure of fitness by appealing to intuition (perhaps incorporating features such as lifespan, food consumption, fecundity, etc.), but there is no reason why such intuitions must necessarily hold. In general, we do not know what features are being selected for by evolution, therefore we cannot claim that any set of such features reliably represents evolutionary fitness. Indeed, preliminary studies have indicated the impracticability of such an approach: relationships between neural properties and putative measures of fitness have proved to be nontrivial and difficult to interpret.

Since it seems infeasible to *infer* individual-level fitness post hoc, I instead *impose* population-level fitness in the experimental design. By varying simulation parameters, I define a set of three experimental conditions with increasing environmental complexity, thereby requiring the evolution of increasingly complex adaptations. Gross energetics remain identical across all simulations, allowing their results to be directly compared. Note that this represents a departure from the *Legacy* simulation parameters introduced by [Yaeger et al. \(2008\)](#) and used throughout the experiments discussed in previous chapters.

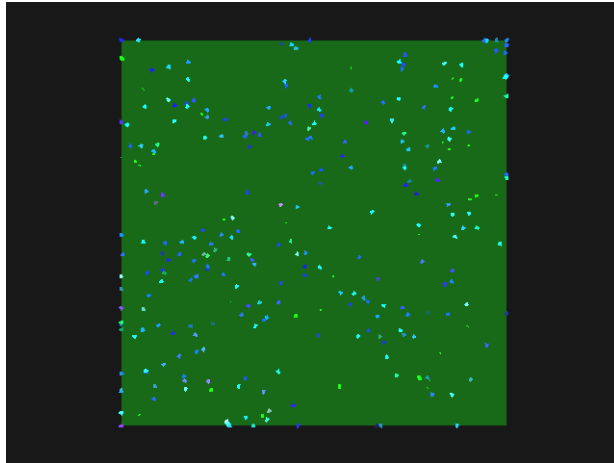
With this experimental design in place, I investigate trends in neural properties across conditions. Properties that are correlated with adaptive behavior should vary directly with environmental complexity. I therefore hypothesize significant cross-condition differences in structural small-worldness, dynamical criticality, and effective information processing. Increases in environmental complexity should be accompanied by increases in each relevant metric. I exclude measures of information transmission from this investigation: the information-processing analysis conducted in this chapter omits all formulations of the transfer entropy. As discussed in Chapter 7, these metrics exhibit counterintuitive evolutionary trends that are difficult to interpret without further analysis.

8.2 Methods

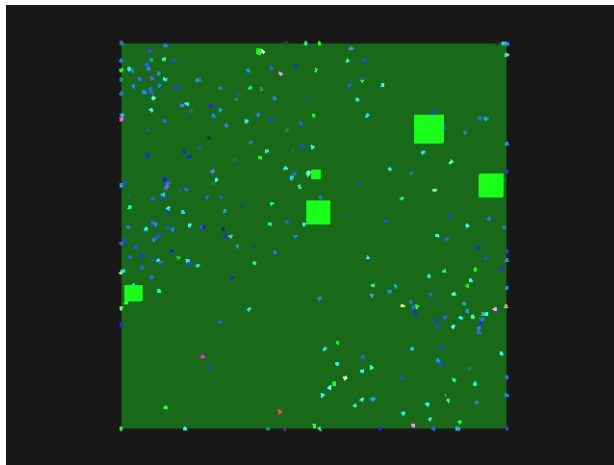
The experimental conditions used in this synthesis are shown in Figure 8.1. Videos of typical simulations are available at https://youtu.be/U2w1_q73Wb8, <https://youtu.be/-U8EvRlz27U>, and <https://youtu.be/Gk3tBPuqmPY>.

In the *Simple* condition (see Figure 8.1a), food items of nominal size grow randomly but consistently throughout the environment. Since food items are small and evenly spread, opportunities for food consumption are mostly a matter of luck: whenever a food item is created, it is quickly consumed by an agent that, by chance, happens to be nearby. The *Simple* condition thus provides very little for natural selection to take advantage of. Indeed, over evolutionary time, there are few qualitative changes in agent behavior beyond increases in Mate and Move biases, accompanied by a decrease in Fight bias. Agents adopt an essentially uniform distribution throughout the world and exhibit a simple survival strategy, typically turning in constant circles, canvassing the local environment for food.

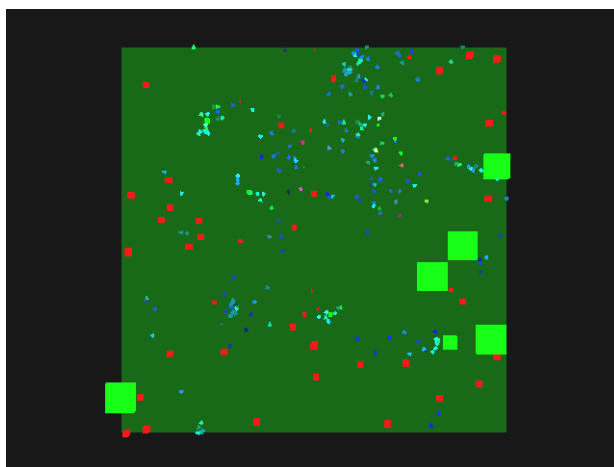
In the *Forage* condition (see Figure 8.1b), individual food items become larger (i.e., they contain more energy when created) over the first 10,000 timesteps. This increased size is offset by less frequent growth,



(a) Simple



(b) Forage



(c) Poison

Figure 8.1: Typical *Simple* (top), *Forage* (middle), and *Poison* (bottom) simulations at the 10,000th timestep. Contrast has been enhanced.

keeping the overall rate of energy influx identical to the [Simple](#) condition. By the 10,000th timestep, food size has increased tenfold relative to its initial value, and it correspondingly grows at a tenth of the initial rate. Effective foraging becomes necessary for individual survival: food grows too patchily for an agent to rely on luck in finding it. Agents must visually recognize and seek out food; those that do so more effectively than their contemporaries gain a considerable advantage. As a result, the evolved agent distribution is patchier, mediated by a simple but efficient foraging strategy: agents turn in circles until a food item appears nearby, at which point they travel toward it, circling it until it is consumed. A modest flocking strategy also evolves: even in the absence of food, groups of agents may be observed forming trails or circling one another. This behavior is also adaptive, as food is often only visible to the “leaders” of a trail or to the agents on the fringes of a flock. But agents who follow these leaders are typically rewarded with an opportunity for food consumption when they reach their destination.

In the [Poison](#) condition (see [Figure 8.1c](#)), food growth follows the same schedule as in the [Forage](#) condition, becoming progressively patchier throughout the first 10,000 timesteps. Additional complexity is introduced by the presence of “poisonous” (energy-reducing) food items, which grow consistently throughout the environment, increasing in potency until the 10,000th timestep. These items are colored red, allowing agents to visually distinguish between them and the normal, energy-increasing food items, which are green. At full potency, a single poison item can be deadly even to an agent at full health energy level. Therefore, a successful survival strategy requires not only that agents effectively forage for food, but also that they avoid consumption of poison. Typical strategies include physical avoidance (via the [Move](#) and [Turn](#) actions) and selective suppression of the [Eat](#) action in response to visual input. The evolved flocking behavior initially observed in the [Forage](#) condition becomes more exaggerated in the [Poison](#) condition. The presence of poison appears to induce more “deliberate” movements: the exploratory drive supporting an agent’s foraging behavior must be tempered by its need for self-preservation. As a result, agents congregate in tighter flocks, exhibiting almost no forward movement until presented with a nearby food item that is sufficiently unobstructed by poison.

Full detail of the parameters used in the [Simple](#), [Forage](#), and [Poison](#) conditions may be found in [Appendix A](#). In addition to the details highlighted above—explaining the ways in which these three conditions differ among themselves—many changes have been made which distinguish these “modern” conditions from the [Legacy](#) condition. Some of these changes facilitate the experimental setup described in this chapter, while others allow these experiments to make use of relatively recent enhancements to Polyworld. Major

differences in parameter settings with respect to the [Legacy](#) condition are described below.

MaxSteps

Legacy Simulations are run for 16,000 timesteps, providing an analysis window of 15,000 timesteps. Driven agents born within this window are guaranteed to die naturally rather than having their lives artificially truncated at the end of the simulation. This facilitates the analyses presented in previous chapters, which make use of the neural anatomy recorded at agent death.

Modern Simulations are run for 20,000 timesteps, which has proved to be ample for the evolution of a successful survival strategy even in the [Poison](#) condition. It is not necessary to run simulations for additional 1000 timesteps: as described below, Hebbian learning is only applied during gestation. Since synaptic weights do not vary throughout an agent's lifetime, the analysis presented in this chapter may make use of the neural anatomy recorded at agent birth.

Initialization

Legacy Agents in the seed population are initialized at maximum energy and are permitted to mate immediately. Early in a simulation, this can lead to population boom-and-bust cycles caused by a significant proportion of the population reproducing in a synchronized fashion.

Modern The states of seed agents are randomized, suppressing population boom-and-bust cycles. Since food items have a finite lifespan in these simulations, their states are similarly randomized: food items created in the initial timestep are assigned random ages. These modifications bring the initial state of a simulation into closer alignment with its steady state.

Barriers and Domains

Legacy The world is separated into three equal-area regions by two barriers. Food grows in two patches at opposite ends of the world.

Modern The gross structure of the environment is completely uniform: there are no barriers, and a single food patch covers the entire world.

FoodColor

Legacy Food items are approximately “forest green” (#339933 ■).

Modern Food items are pure green (#00FF00 ■) and poison items are pure red (#FF0000 ■), making them easier to visually identify.

FoodMaxLifeSpan

Legacy Food items have an infinite lifespan. If not consumed, they remain in place indefinitely.

Modern Food items have a lifespan of 250 timesteps. They are removed from the simulation if not consumed before this time elapses.

MinFood

Legacy A minimum of 90 food items is enforced at all times, making the [Legacy](#) environment relatively benign. An agent that finds itself in one of the world’s food patches rarely needs to move very far in order to find food or mates.

Modern There is no minimum number of food items, forcing the agent population to adapt to the world’s food growth rate.

Population Control

Legacy Energy usage is increased/decreased *linearly* when the population is above/below its nominal level of 180 agents.

Modern Energy usage is increased/decreased *quadratically* when the population is above/below its nominal level of 200 agents.

MinAgents and MaxAgents

Legacy The population is permitted to range from 90 to 300 agents.

Modern The population is permitted to range from 50 to 1000 agents. This upper limit is effectively infinite, as it is far above the carrying capacity of the environment, and would in any case be associated with prohibitively high energy usage.

AgentsAreFood

Legacy No carcass food items are created upon agent death. Predation is therefore impossible, but the Fight action is still potentially useful as a way of competing for resources.

Modern Carcass food items are created upon agent death, reintroducing the possibility of predation. Despite this, fighting is still de-emphasized in the evolved survival strategy, perhaps due to its relatively high energy usage.

EnergyUseNeurons and EnergyUseSynapses

Legacy Neural energy usage is disabled to avoid imposing physiological constraints on network topology.

Modern Neural energy usage is reinstated, though for networks of typical size, it is negligible compared to activity-based energy usage.

BodyGreenChannel

Legacy The green component of an agent's body color is proportional to the value of its ID gene and does not change throughout the agent's life.

Modern The green component of an agent's body color is proportional to its food energy level. Since food items are also green, this aids in foraging: the presence of bright green in an agent's visual field reliably indicates food—either a food item itself or an agent who has recently eaten.

EnableMateWaitFeedback and EnableSpeedFeedback

Legacy Agents are provided with sensors for vision, health energy level, and a source of randomness.

Modern Two additional sensors provide agents with feedback on their own fertility and their rate of forward motion.

NoseColor

Legacy The color components of the front surface of an agent are constrained to be equal. I.e., an agent’s “nose” must be black, white, or some shade of gray. The actual brightness is controlled by the agent’s Light volition.

Modern The Light action is disabled, making an agent’s color uniform over its entire body. This simplification allows more reliable visual parsing of the environment.

ProbabilisticMating

Legacy Mating is deterministic: as long as an agent’s Mate volition is above a threshold, it mates at the first available opportunity.

Modern Mating is probabilistic: provided it exceeds a threshold, an agent’s Mate volition represents its likelihood of attempting to mate. This is more consistent with other thresholded actions such as Eat and Fight.

Neural Architecture

Legacy An agent’s neural network has between one and 16 visual input neurons per color, as specified by the Red, Green, and Blue genes. Up to five internal neural groups are possible, each with a maximum of 16 excitatory and 16 inhibitory neurons per group, yielding an overall maximum of 160 internal neurons.

Modern An agent’s neural network has exactly seven visual input neurons per color; the Red, Green, and Blue genes are disabled. Up to ten internal neural groups are possible, each with a maximum of seven excitatory and seven inhibitory neurons per group, yielding an overall maximum of 140 internal neurons. Total network size is thus comparable to the Legacy condition. However, there is a greater degree of evolutionary control over neural wiring, since neural genes are specified per pair of groups.

MirroredTopologicalDistortion

Legacy For each pair of neural groups, the TopologicalDistortion gene interpolates between sequential and random connections between neurons.

Modern The TopologicalDistortion gene interpolates between sequential, random, and antisequential connections. This allows the genome to specify, e.g., selective wiring of either the left or right portions of a visual input group.

OrderedInternalNeuralGroups

Legacy An internal neural group's physical position in the genome controls its logical priority during development. I.e., internal neural groups are grown in the order their genes are laid out. Groups which appear later in the genome are not grown unless all preceding groups are grown.

Modern Each internal neural group is assigned a gene indicating its logical priority during development. This allows evolution to operate more effectively over genes controlling neural wiring. E.g., a single neural group may be grown regardless of its physical position in the genome.

Learning

Legacy Synaptic weights are initialized randomly at the time of an agent's inception. During gestation, the agent's neural network is exposed to uncorrelated noise while synaptic weights are updated via Hebbian learning. This learning process continues throughout the agent's lifetime, with synaptic weights potentially increasing to a maximum of 8.

Modern Synaptic weights are set to a globally constant value at inception, and Hebbian learning occurs only during gestation. This period is greatly increased in length, ultimately allowing comparable synaptic weights to be achieved. This removes some randomness from the neural development process and alleviates difficulties in analysis caused by an agent's neural anatomy changing throughout its lifetime. The maximum synaptic weight in these conditions is 5.

YawEncoding

Legacy An agent's Turn action is controlled by a single output neuron. When nominally activated, this neuron produces no rotation. When saturated on or off, it produces maximal right or left rotation, respectively.

Modern Turning is controlled by a pair of opposing output neurons, each dedicated to either right or

left rotation. The degree of rotation is proportional to the relative difference in activation between these two neurons.

Evolvable Genes

Legacy An agent's genome controls aspects of its genetics, physiology, and neural wiring.

Modern All genetic adaptation is constrained to occur in neural genes. Meta-genetic and physiological genes are fixed at globally constant values across the entire agent population.

AgeEnergyMultiplier and DieAtMaxAge

Legacy Maximum lifespan is genetically specified. An agent whose age reaches this value dies, regardless of its health energy level.

Modern Replacing the LifeSpan gene, an aging mechanism is introduced whereby agents become progressively inefficient throughout their lifetimes. I.e., an agent expends more and more energy on the same actions as it ages.

GeneticOperatorResolution

Legacy While an agent's genome is interpreted at byte-level resolution, genetic operators are implemented at bit-level resolution. Crossover may occur in the middle of a gene's bits. Mutation occurs randomly and independently for each bit; the magnitude of a single mutation may be as large as $2^7 = 128$ or as small as $2^0 = 1$, depending on where it occurs in the gene.

Modern Genetic operators are implemented at byte-level resolution. Crossover may only occur at gene/byte boundaries. All genes undergo mutation during reproduction, with the magnitude of each mutation drawn from a normal distribution having a mean of zero and a globally constant standard deviation.

GenomeLayout and MaxCrossoverPoints

Legacy The genome is laid out such that genes coding for the same property (e.g., ConnectionDensity, TopologicalDistortion, etc.) are contiguous. Genes coding for a particular pair of neural groups are

spread throughout the genome. Between two and eight crossover points are used during reproduction.

Modern The genome is laid out such that genes coding for the same pair of neural groups are contiguous. Genes coding for a particular property are spread throughout the genome. Single-point crossover is used during reproduction.

SeedType

Legacy Intelligent defaults are designed into the genomes of the seed population. These include neural wiring parameters that encourage seeking food and mates and avoiding fights.

Modern Designed-in neural wiring has been removed, though nominal biases toward eating, mating, moving, and turning remain in order to ensure simulation viability.

8.3 Results

Figures 8.2, 8.3, and 8.4 show population-wide distributions of relevant metrics for agents born in the last 5000 timesteps of ten *Simple*, *Forage*, and *Poison* simulations. I report the p -value for a two-tailed t -test of adjacent conditions, indicating whether the corresponding population means are equal. Since simulations are not paired (an in the earlier driven/passive analyses), and since population sizes and variances are therefore likely to be unequal, I use a Welch's t -test rather than a paired Student's t -test.

As expected, increases in environmental complexity (from *Simple* to *Forage* and from *Forage* to *Poison*) are accompanied by increases in small-worldness, state space expansion, complexity, active information storage, and separable information (both positive and negative). As noted previously, little “memory” is required to produce adaptive behavior in these simulations. This results in highly skewed distributions for active information storage.

8.4 Conclusion

In Polyworld, environmental complexity is correlated with the neural properties analyzed in previous chapters. Small-world structure, critical dynamics, and effective information processing are all amplified in response to challenges imposed by the experimental condition. These properties support the increasingly complex adaptations required to survive in increasingly complex environments. It appears, then, that the previously

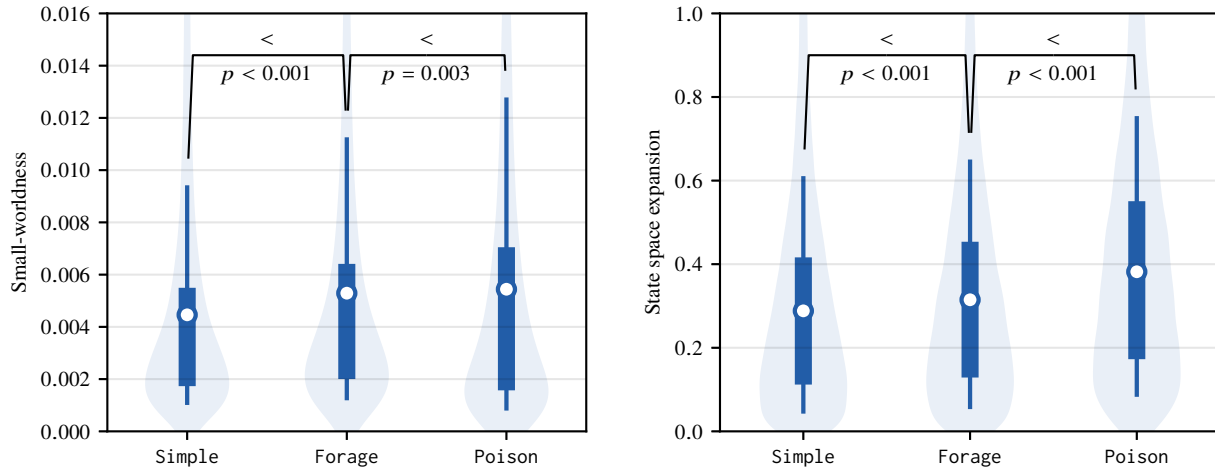


Figure 8.2: Small-worldness (left) and state space expansion (right) for agents born in the last 5000 timesteps of ten *Simple*, *Forage*, and *Poison* simulations. For each condition, the box indicates the interquartile range, whiskers extend to the top and bottom decile, and the circle represents the mean. The p -value is reported for a two-tailed Welch’s t -test of adjacent conditions.

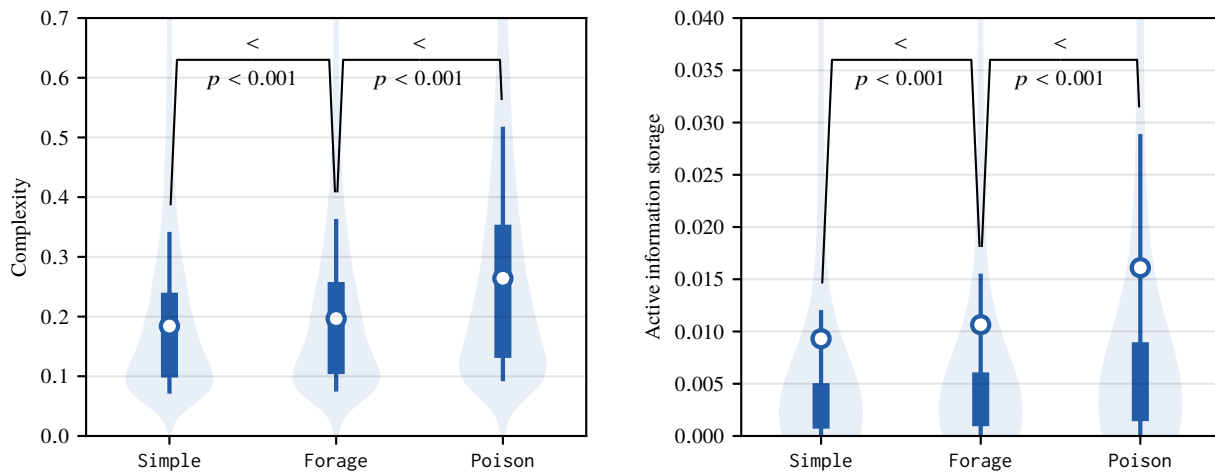


Figure 8.3: Complexity (left) and active information storage (right) for agents born in the last 5000 timesteps of ten *Simple*, *Forage*, and *Poison* simulations. For each condition, the box indicates the interquartile range, whiskers extend to the top and bottom decile, and the circle represents the mean. The p -value is reported for a two-tailed Welch’s t -test of adjacent conditions.

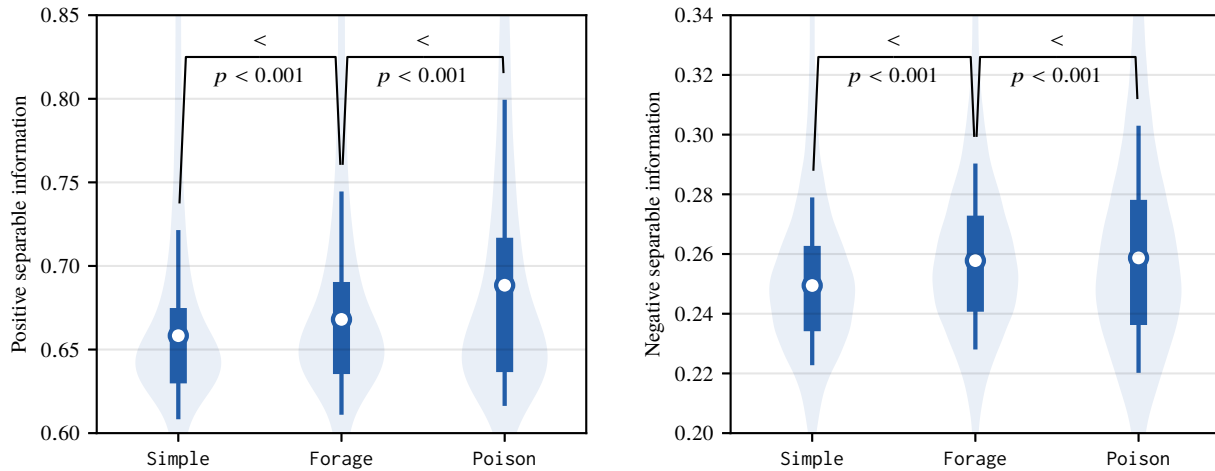


Figure 8.4: Positive separable information (left) and negative separable information (right) for agents born in the last 5000 timesteps of ten *Simple*, *Forage*, and *Poison* simulations. For each condition, the box indicates the interquartile range, whiskers extend to the top and bottom decile, and the circle represents the mean. The p -value is reported for a two-tailed Welch's t -test of adjacent conditions.

recognized evolutionary trends toward the amplification of these properties are indeed driven by direct links between them and adaptive behavior.

Chapter 9

Conclusion

Returning to the motivating question of this work, what makes a brain smart? Based on the results presented here, I conclude that neural networks with *small-world structure* support *critical dynamics* which facilitate *effective information processing*. Together these properties foster the production of adaptive behavior.

I have provided two main pieces of evidence for this claim. First, these properties are amplified during a period of positive evolutionary selection. This suggests correlations with adaptive behavior which are capitalized upon by natural selection, resulting in driven evolutionary trends. Second, these properties are directly correlated with environmental complexity. In other words, harder problems require “more” small-world structure, critical dynamics, and effective information processing in order to implement successful survival strategies.

Directions for possible extension of this work have also been noted.

First, neural properties of interest could be correlated with specific genetic features—individual alleles or more complex schemata—known to experience positive evolutionary selection. Such studies could more directly confirm those properties’ adaptive significance.

Second, the local information dynamics of behaving agents could be studied in more detail. Such studies could help explain how agents employ information storage, transmission, and modification in the production of adaptive behavior. They could also shed more light on the unexpected evolutionary trend toward decreased information transmission.

Third, relationships among the metrics discussed here could be explored. Such studies could verify the intuitions that drove earlier hypotheses (e.g., that complexity is fostered by small-world structure) and demonstrate whether neural properties facilitate one another (e.g., whether effective information processing

requires critical dynamics). Preliminary studies have indicated that many of these relationships are nontrivial; much work must be done to determine whether such results highlight peculiarities of Polyworld itself, or whether they indicate deeper connections between neural properties.

Appendix A

Worldfiles

This appendix describes the worldfile parameters used to customize Polyworld simulations. The worldfiles used in this dissertation are also provided. Note that all parameters are optional: those not explicitly specified in a worldfile are given default values.

A.1 Parameters

In the descriptions below, *genetic algorithm* (GA) refers specifically to the steady-state (constant-population) GA, which uses artificial selection to optimize complexity and/or an ad hoc fitness function. It may be invoked explicitly by certain parameters or in response to the population dropping below a threshold.

A.1.1 Simulation

Parameter	Description
AdaptivityMode	Whether to run the simulation in adaptivity mode, a post hoc analysis mode in which all births are prohibited.
CheckPointFrequency	Interval (i.e., number of timesteps) at which checkpoints are made. Intended to allow restarting a simulation from a specified point in time, this functionality is not currently implemented.
InitSeed	Random number generator seed applied prior to initializing the simulation.

Parameter	Description
LegacyMode	Whether to use legacy default values for parameters that are not explicitly specified.
MaxSteps	Number of timesteps at which the simulation is terminated.
ParallelBrains	Whether all agents' neural networks are updated concurrently. Requires StaticTimestepGeometry to be true.
ParallelCreateAgents	Whether agents introduced by the GA are created concurrently.
ParallelInitAgents	Whether agents in the initial population are created concurrently.
ParallelInteract	Whether agents interact with the environment (and each other) concurrently.
PassiveLockstep	Whether to run the simulation in passive mode, a post hoc analysis mode in which natural births and deaths are prohibited. See Chapter 3 for more information.
PositionSeed	Random number generator seed applied prior to assigning initial positions to objects. This parameter is not currently used.
SeedGenomeFromRun	Whether to initialize genomes based on a previous simulation.
SeedPositionFromRun	Whether to initialize agent positions based on a previous simulation.
SeedSynapsesFromRun	Whether to initialize synaptic weights based on a previous simulation.
SimulationSeed	Random number generator seed applied prior to the first timestep. If zero, specifies that the generator is not reseeded.
StaticTimestep- Geometry	Whether all agents' visual inputs are updated simultaneously at the beginning of each timestep rather than sequentially.
Variables	Arbitrary variables that may be used to help specify other parameters.

A.1.2 Environment

Parameter	Description
AgentHeight	Height of agents.
BarrierColor	Color of barriers.
BarrierHeight	Height of barriers.

Parameter	Description
Barriers	Barrier positions, expressed either in raw coordinates or as proportions of WorldSize . See RatioBarrierPositions .
BrickColor	Color of bricks.
BrickHeight	Height of bricks.
Domains	Nonoverlapping portions of the world allowing localized management of agents, food items, and bricks.
Edges	Type of boundary imposed at world edges: wraparound (toroidal), blocked (agents cannot move past edges but can slide along them), sticky (agents cannot move past or slide along edges), or tabletop (agents that move past edges immediately die).
ExpFogDensity	Fog density if FogFunction is exponential.
FogFunction	Type of fog applied during visual rendering: off, linear, or exponential.
FoodColor	Color of food items.
FoodHeight	Height of food items.
GroundClearance	Separation between the ground plane and the bottom surfaces of objects.
GroundColor	Color of the ground plane.
LinearFogEnd	Maximum visibility (distance) if FogFunction is linear.
RatioBarrierPositions	Whether Barriers are expressed as proportions of WorldSize .
SolidAgents	Whether agents are prevented from passing through one another.
SolidBricks	Whether agents are prevented from passing through bricks.
SolidFood	Whether agents are prevented from passing through food items.
StickyBarriers	Whether agents are prevented from sliding along barriers.
WorldSize	Width and depth of the world.

A.1.3 Food

Parameter	Description
FoodEnergySizeScale	Scales the amount of energy a food item contains to determine its size (width and depth): $(\text{size}) = \frac{3}{4} \times (\text{energy}) \div (\text{FoodEnergySizeScale}).$
FoodGrowthModel	Role of the fractional part of <code>FoodGrowthRate</code> : independent (raw probability of growing a food item) or relative (multiplied by the difference between <code>MaxFoodGrown</code> and the number of existing food items to determine food item growth probability).
FoodGrowthRate	Number of food items grown per timestep.
FoodMaxLifeSpan	Number of timesteps after which an uneaten food item is removed from the simulation.
FoodRemoveEnergy	Minimum amount of energy remaining in a food item after being partially eaten. Food items that are eaten to an energy level below this threshold are removed from the simulation.
FoodRemoveFirstEat	Whether food items are removed from the simulation when first eaten, regardless of the amount of energy remaining.
FoodTypes	Types of food. Includes subparameters that control energy polarity (used in conjunction with <code>AgentMetabolisms</code>) and color.
InitFood	Number of food items grown when the simulation is initialized.
Max/MinFood	Maximum/minimum number of food items. At each timestep, food items are forcibly grown (if necessary) until <code>MinFood</code> is reached. Above <code>MaxFood</code> , all food creation is prohibited.
Max/MinFoodEnergy	Maximum/minimum amount of energy contained by newly grown food items. The actual energy level is drawn from a uniform distribution over this range.
MaxFoodGrown	Maximum number of food items. Above this threshold, natural food growth is prohibited, but carcass food items may still be created.
NumEnergyTypes	Number of nutrients contained in each food item.

Parameter	Description
ProbabilisticFood-Patches	Whether food growth is performed probabilistically across patches rather than independently for each patch.
RandomInitFoodAge	Whether initial food items have random ages drawn from a triangular distribution with a maximum of <code>FoodMaxLifeSpan</code> .

A.1.4 Population

Parameter	Description
AllowBirths	Whether births are permitted. Used in conjunction with <code>AdaptivityMode</code> .
AllowMinDeaths	Whether the population is permitted to drop to <code>MinAgents</code> . Deaths resulting in this condition may normally be prevented by other forms of population control. If this parameter is true, such deaths are stochastically permitted based on the number of agents created by the GA as a proportion of the total number of agents ever observed.
ApplyLowPopulation-Advantage	Whether to scale energy usage and fight damage when the population drops below <code>InitAgents</code> : $(\text{scale factor}) = 1 - \frac{(\text{InitAgents}) - (\text{population})}{(\text{InitAgents}) - (\text{MinAgents})}$
ComplexityFitness-Weight	Relative weight given to complexity in the GA's fitness function. If nonzero, specifies that the GA should run for the entire simulation.
ComplexityType	Type of complexity used by the GA's fitness function. Specifies how neural activation time series are filtered and which subset sizes k are used in applying Equation 3.6.
EliteFrequency	Interval (i.e., number of creations) at which the GA recreates the single fittest agent ever observed.
EndOnPopulationCrash	Whether to prematurely terminate the simulation if the population drops to <code>MinAgents</code> .

Parameter	Description
EnergyBased- PopulationControl	Whether to scale energy usage and fight damage based on population. The scale factor is nominal (i.e., its value is unity) in the range defined by PopControlMinFixedRange and PopControlMaxFixedRange . Outside this range, the scale factor decreases linearly to PopControlMinScaleFactor at MinAgents and increases quartically to PopControlMaxScaleFactor at MaxAgents .
FitnessWeightEating	Ad hoc fitness contribution based on food consumption.
FitnessWeightEnergy- AtDeath	Ad hoc fitness contribution based on health energy level at death.
FitnessWeight- Longevity	Ad hoc fitness contribution based on lifespan.
FitnessWeightMating	Ad hoc fitness contribution based on offspring count.
FitnessWeightMoving	Ad hoc fitness contribution based on distance moved.
HeuristicFitness- Weight	Relative weight given to ad hoc fitness in the GA's fitness function. If nonzero, specifies that the GA should run for the entire simulation.
InitAgents	Number of agents in the initial population.
Max/MinAgents	Maximum/minimum number of agents.
NumberFittest	Number of agents to maintain in the list of fittest agents ever observed.
NumberRecentFittest	Number of agents to maintain in the list of fittest agents recently observed (at intervals of EpochFrequency timesteps).
NumDepletionSteps	If nonzero, imposes an energy penalty when the population rises above InitAgents : $(\text{penalty}) = \frac{(\text{energy capacity})}{(\text{NumDepletionSteps})} \times \frac{(\text{population}) - (\text{InitAgents})}{(\text{MaxAgents}) - (\text{InitAgents})}$
PairFrequency	Interval (i.e., number of creations) at which the GA creates agents by mating two parents selected from the list of fittest agents ever observed, unless preempted by EliteFrequency . Other agents created by the GA are given randomized genomes.
PopControlDomains	Whether to apply EnergyBasedPopulationControl per domain.

Parameter	Description
PopControlGlobal	Whether to apply EnergyBasedPopulationControl globally.
PopControlMax/Min-FixedRange	Maximum/minimum population at which the scale factor due to EnergyBasedPopulationControl is nominal, expressed as a proportion of the range defined by MinAgents and MaxAgents .
PopControlMax/Min-ScaleFactor	Maximum/minimum value of the scale factor due to EnergyBasedPopulationControl .
SmiteAgeFrac	Minimum age at which an agent is eligible for smiting, expressed as a proportion of its maximum lifespan.
SmiteFrac	Number of agents to maintain in the list of least fit agents (i.e., the list of agents eligible for smiting) if SmiteMode is least-fit.
SmiteMode	Type of smiting, a mechanism whereby existing agents are killed when mating occurs while the population is MaxAgents : off (no smiting occurs, and mating in such a situation is denied), random (an agent selected at random is smitten), or least-fit (an agent selected from the list of least fit agents is smitten).
TournamentSize	Tournament size used by the GA when selecting each parent for mating. If zero, specifies that successive pairs of parents are selected by iterating through the list of fittest agents ever observed.

A.1.5 Energetics

Parameter	Description
AgeEnergyMultiplier	Scales energy usage based on an agent's age: $(\text{scale factor}) = 1 + (\text{AgeEnergyMultiplier}) \times (\text{age}).$
AgentHealingRate	Rate at which an agent can convert food energy into health energy. This functionality is not currently implemented.
AgentMetabolisms	Types of metabolisms. Includes subparameters that control energy polarity (used in conjunction with FoodTypes), carcass food item generation, and the minimum age at which an agent may eat.

Parameter	Description
AgentMetabolism- SelectionMode	Method of assigning metabolisms to agents: genetic (via an additional MetabolismIndex gene) or random. Only applicable if multiple AgentMetabolisms are defined.
AgentsAreFood	Whether a dying agent's remaining food energy is converted to a food item.
DamageRate	Scales an agent's fight power to determine the damage taken by a collocated agent: $(\text{damage}) = (\text{DamageRate}) \times (\text{FightMultiplier})$ $\times (\text{Fight volition}) \times (\text{normalized health energy}) \times (\text{Strength})$ $\times [1 + (\text{normalized Size}) \times [(\text{MaxSizeFightAdvantage}) - 1]].$
EnergyUseCarryAgent	Base energy usage due to carrying an agent.
EnergyUseCarryAgent- Size	Additional energy usage due to a carrying an agent (scaled by the carried agent's Size).
EnergyUseCarryBrick	Energy usage due to carrying a brick.
EnergyUseCarryFood	Energy usage due to carrying a food item (scaled by its size).
EnergyUseDrop	Energy usage due to the Drop action (scaled by volition). This parameter is not currently used.
EnergyUseEat	Energy usage due to the Eat action (scaled by volition).
EnergyUseFight	Energy usage due to the Fight action (scaled by volition).
EnergyUseFixed	Fixed energy usage imposed irrespective of activity.
EnergyUseFocus	Energy usage due to the Focus action (scaled by volition).
EnergyUseGive	Energy usage due to the Give action (scaled by volition).
EnergyUseLight	Energy usage due to the Light action (scaled by volition).
EnergyUseMate	Energy usage due to the Mate action (scaled by volition).
EnergyUseMove	Energy usage due to the Move action (scaled by volition).
EnergyUseMultiplier	Global energy usage scale factor.
EnergyUseNeurons	Energy usage due to neurons (scaled by normalized count).
EnergyUsePickup	Energy usage due to the Pickup action (scaled by volition). This parameter is not currently used.

Parameter	Description
EnergyUseSynapses	Energy usage due to synapses (scaled by normalized count).
EnergyUseTurn	Energy usage due to the Turn action (scaled by volition).
Max/MinAgentMaxEnergy	Maximum/minimum energy capacity. The actual capacity is interpolated using the agent's Size.
MaxSeedEnergy	Maximum energy level of an agent in the initial population, expressed as a proportion of its energy capacity.
Max/MinSizeEnergy-Penalty	Scales energy usage due to the Move and Turn actions based on an agent's Size: $(\text{scale factor}) = (\text{MaxSizeEnergyPenalty}) \times (\text{MaxSpeed})$ $\times [(\text{MinSizeEnergyPenalty}) + (\text{Size}) - (\text{MinAgentSize})]$ $\div [(\text{MinSizeEnergyPenalty}) + (\text{MaxAgentSize}) - (\text{MinAgentSize})].$
MinFoodEnergyAtDeath	Minimum energy contained in a newly created carcass food item. Takes precedence over the actual food energy level of a dying agent.
MinMateEnergyFraction	Minimum health energy level at which an agent may successfully mate, expressed as a proportion of its energy capacity.
RandomSeedEnergy	Whether agents in the initial population have random energy levels drawn from a triangular distribution with a maximum determined by <code>MaxSeedEnergy</code> .
StarvationEnergy-Fraction	Food energy level below which an agent starves, expressed as a proportion of its energy capacity.
StarvationWait	Minimum age at which an agent may starve.

A.1.6 Physiology

Parameter	Description
BodyBlueChannel	Method of setting the blue color component of an agent's body: proportional to Mate volition or inverted health energy level.

Parameter	Description
BodyGreenChannel	Method of setting the green color component of an agent's body: proportional to the ID gene, Light volition, Eat volition, or food energy level.
BodyRedChannel	Method of setting the red color component of an agent's body: proportional to Fight volition or Give volition.
CarryAgents	Whether agents may be carried.
CarryBricks	Whether bricks may be carried.
CarryFood	Whether food items may be carried.
CarryPreventsEat	Probability that carrying an object prevents the Eat action.
CarryPreventsFight	Probability that carrying an object prevents the Fight action.
CarryPreventsGive	Probability that carrying an object prevents the Give action.
CarryPreventsMate	Probability that carrying an object prevents the Mate action.
DropThreshold	Minimum volition at which the Drop action is effective.
EatMateMinDistance	Minimum distance an agent must move after eating in order to successfully mate.
EatMateWait	Minimum number of timesteps an agent must wait after eating in order to successfully mate.
EatThreshold	Minimum volition at which the Eat action is effective.
EatWait	Minimum number of timesteps an agent must wait after eating in order to successfully eat again.
EnableCarry	Whether an agent may carry objects. If true, provides additional PickUp and Drop actions.
EnableGive	Whether an agent may transfer health energy to a collocated agent. If true, provides an additional Give action.
EnableMateWait-Feedback	Whether an agent can sense its fertility. If true, provides an additional input neuron whose activation is proportional to the number of timesteps remaining before the agent may successfully mate.
EnableSpeedFeedback	Whether an agent can sense movement speed. If true, provides an additional input neuron whose activation is proportional to the agent's speed.

Parameter	Description
EnableVisionPitch	Whether an agent may control the vertical orientation of its visual field. If true, provides an additional <code>VisionPitch</code> action.
EnableVisionYaw	Whether an agent may control the horizontal orientation of its visual field. If true, provides an additional <code>VisionYaw</code> action.
EyeHeight	Height of an agent's eye, expressed as a proportion of <code>AgentHeight</code> .
FightMode	Type of fighting: normal or null (causes no health energy loss).
FightMultiplier	Scales an agent's fight power. See <code>DamageRate</code> .
FightThreshold	Minimum volition at which the <code>Fight</code> action is effective.
FoodConsumptionRate	Scales an agent's <code>Eat</code> volition to determine the amount of energy consumed from a collocated food item.
GiveFraction	Scales an agent's <code>Give</code> volition to determine the proportion of its energy transferred to a collocated agent.
GiveThreshold	Minimum volition at which the <code>Give</code> action is effective.
InitMateWait	Additional <code>MateWait</code> applied to newly born or created agents.
InvertFocus	Whether to invert the effect of the <code>Focus</code> action. If true, a larger neural activation corresponds to a smaller horizontal field of view.
InvertMateWait-Feedback	Whether to invert the effect of the <code>MateWait</code> sensor. If true, a larger neural activation corresponds to a shorter wait time.
MateThreshold	Minimum volition at which the <code>Mate</code> action is effective.
MateWait	Minimum number of timesteps an agent must wait after mating in order to successfully mate again.
Max/MinAgentMaxSpeed	Phenotypic range of the <code>MaxSpeed</code> gene.
Max/MinAgentSize	Phenotypic range of the <code>Size</code> gene.
Max/MinAgentStrength	Phenotypic range of the <code>Strength</code> gene.
MaxCarries	Maximum number of objects an agent may carry.
Max/MinEatVelocity	Maximum/minimum movement speed at which the <code>Eat</code> action is effective, expressed as a proportion of <code>MaxVelocity</code> .

Parameter	Description
MaxEatYaw	Maximum turning angle at which the Eat action is effective, expressed as a proportion of YawRate .
Max/MinEnergy-FractionToOffspring	Phenotypic range of the MateEnergyFraction gene.
Max/MinHorizontal-FieldOfView	Maximum/minimum horizontal field of view of an agent's vision. The actual field of view is interpolated using Focus volition.
MaxMateVelocity	Maximum movement speed at which the Mate action is effective, expressed as a proportion of MaxVelocity .
MaxSizeFightAdvantage	Scales the effect an agent's Size has on its fight power. See DamageRate .
MaxVelocity	Maximum movement speed.
Max/MinVisionPitch	Maximum/minimum angle of vertical visual field orientation. The actual angle is interpolated using VisionPitch volition.
Max/MinVisionYaw	Maximum/minimum angle of horizontal visual field orientation. The actual angle is interpolated using VisionYaw volition.
MotionRate	Scales an agent's Move volition to determine movement speed.
NoseColor	Method of setting the color of an agent's nose: proportional to Light volition or identical to body color.
PickupThreshold	Minimum volition at which the Pickup action is effective.
ProbabilisticMating	Whether Mate volition represents an agent's probability of attempting to mate. If false, specifies that any Mate volition exceeding MateThreshold represents an attempt to mate.
RandomBirthLocation	Whether newly born agents are placed at a random position in the world rather than the average position of their parents.
RandomBirthLocation-Radius	Maximum distance a newly born agent may be placed from its parents, expressed as a proportion of WorldSize . Requires RandomBirthLocation to be true.
RandomSeedMateWait	Whether agents in the initial population have random MateWait values drawn from a uniform distribution with a maximum of MateWait .

Parameter	Description
RetinaHeight	Height in pixels of the buffer used to render an agent's visual field.
RetinaWidth	Width in pixels of the buffer used to render an agent's visual field.
ShieldAgents	Whether an agent is protected from fight damage by a carried agent.
ShieldBricks	Whether an agent is protected from fight damage by a carried brick.
ShieldFood	Whether an agent is protected from fight damage by a carried food item.
VerticalFieldOfView	Vertical field of view of an agent's vision.
Vision	Whether visual input is provided to agents.
YawRate	Scales an agent's Turn volition to determine turning angle.

A.1.7 Neural Model

Parameter	Description
BrainArchitecture	Method of growing neural topologies: groups (connectivity is specified at the level of neural groups that have no spatial arrangement) or sheets (synapses grow between successive neural sheets located in a virtual space).
EnableInitWeightRngSeed	Whether to provide an additional <code>InitWeightRngSeed</code> gene for each pair of neural groups whose phenotypic value seeds a local random number generator used in initializing synaptic weights.
EnableSpikingGenes	If <code>NeuronModel</code> is spiking, whether to provide additional genes for each neural group that control parameters in the neural activation equation.
EnableTopologicalDistortionRngSeed	Whether to provide an additional <code>TopologicalDistortionRngSeed</code> gene for each pair of neural groups whose phenotypic value seeds a local random number generator used in applying topological distortion.
FixedInitSynapseWeight	Whether initial synaptic weights are fixed at <code>MaxSynapseWeightInitial</code> rather than being drawn from a uniform distribution.
FreezeSeededSynapses	Whether synaptic weights initialized via <code>SeedSynapsesFromRun</code> are frozen (i.e., prohibited from changing via Hebbian learning), possibly overriding <code>LearningMode</code> . Used in conjunction with <code>AdaptivityMode</code> .

Parameter	Description
GainMax/Min	Phenotypic range of Gain genes.
GainSeed	Normalized value of Gain genes for seed agents.
GaussianInitSynapse- Weight	Whether to draw the magnitudes of initial synaptic weights from a half-normal distribution whose standard deviation is controlled by an additional WeightStdev gene for each pair of neural groups.
GaussianInitSynapse- WeightMaxStdev	Phenotypic maximum of WeightStdev genes.
LearningMode	Life stages during which Hebbian learning is applied: none, prebirth (offline gestation period only), or all.
LogisticSlope	Logistic slope α used in Equation 3.3.
MaxBiasWeight	Phenotypic maximum of Bias genes.
Max/MinConnection- Density	Phenotypic range of ConnectionDensity genes.
Max/MinExcitatory- NeuronsPerGroup	Phenotypic range of ExcitatoryNeuronCount genes.
Max/MinInhibitory- NeuronsPerGroup	Phenotypic range of InhibitoryNeuronCount genes.
Max/MinInitWeightRng- Seed	Phenotypic range of InitWeightRngSeed genes.
Max/MinInternal- NeuralGroups	Phenotypic range of the InternalNeuronGroupCount gene.
Max/MinLearningRate	Phenotypic range of LearningRate genes.
MaxSynapseWeight	Maximum synaptic weight.
MaxSynapseWeight- Initial	Maximum initial synaptic weight.
Max/MinTopological- Distortion	Phenotypic range of TopologicalDistortion genes.

Parameter	Description
Max/MinTopological-DistortionRngSeed	Phenotypic range of TopologicalDistortionRngSeed genes.
Max/MinVisionNeurons-PerGroup	Phenotypic range of Red, Blue, and Green visual input neuron count genes.
MirroredTopological-Distortion	Whether topological distortion interpolates from sequential to antisequential connectivity (with random connectivity at the midpoint) rather than from sequential to random connectivity.
NeuronModel	Method of updating neural activations: rate-coded (as in Equation 3.1), tau/gain (as in Equation 3.1 with the addition of a time constant and gain controlled by Tau and Gain genes for each neural group), or spiking (as in Izhikevich's (2003) Equations 1, 2, and 3).
OrderedInternal-NeuralGroups	Whether to provide an additional Order gene for each internal neural group that controls the order in which groups are grown. This only has an effect when the number of internal neural groups is less than the maximum.
OutputSynapseLearning	Whether Hebbian learning is applied to synapses grown from output neurons.
PreBirthCycles	Length (i.e., number of timesteps) of the offline gestation period.
SeedDropBias	Normalized Drop action bias for seed agents if SeedType is legacy.
SeedDropExcitation	Degree to which visual input encourages the Drop action for seed agents if SeedType is legacy.
SeedFightBias	Normalized Fight action bias for seed agents if SeedType is legacy.
SeedFightExcitation	Degree to which red visual input encourages the Fight action for seed agents if SeedType is legacy. Also controls the degree to which the Eat and Mate actions suppress the Fight action.
SeedGiveBias	Normalized Give action bias for seed agents if SeedType is legacy.
SeedPickupBias	Normalized Pickup action bias for seed agents if SeedType is legacy.
SeedPickupExcitation	Degree to which visual input encourages the Pickup action for seed agents if SeedType is legacy. Also controls the degree to which the Pickup action suppresses the Drop action.

Parameter	Description
SeedVisionNeurons	Number of visual input neurons per color component for seed agents, expressed as a proportion of the range defined by MinVisionNeuronsPerGroup and MaxVisionNeuronsPerGroup .
Sheets	Parameters that control the growth of neural topologies if BrainArchitecture is sheets.
SimpleSeedConnection-Density	Connection density between all pairs of neural groups for seed agents if SeedType is simple.
SimpleSeedIO-ConnectionDensity	Connection density between input-to-output pairs of neural groups for seed agents if SeedType is simple. Overrides SimpleSeedConnectionDensity .
SimpleSeedYawBias-Delta	Normalized deviation in Turn action bias from nominal (i.e., no turning) for seed agents if SeedType is simple. The direction of turning (left or right) is chosen randomly.
SpikingAMax/Min	Phenotypic range of genes that control Izhikevich's (2003) parameter <i>a</i> .
SpikingBMax/Min	Phenotypic range of genes that control Izhikevich's (2003) parameter <i>b</i> .
SpikingCMax/Min	Phenotypic range of genes that control Izhikevich's (2003) parameter <i>c</i> .
SpikingDMax/Min	Phenotypic range of genes that control Izhikevich's (2003) parameter <i>d</i> .
SynapseFromInputTo-OutputNeurons	Whether synapses may grow directly from input neurons to output neurons.
SynapseFromOutput-Neurons	Whether synapses may grow from output neurons.
SynapseWeightDecay-Rate	Decay rate <i>r</i> used in Equation 3.5 .
TauMax/Min	Phenotypic range of Tau genes.
TauSeed	Normalized value of Tau genes for seed agents.
YawEncoding	Method of determining Turn volition: squash (proportional to the activation of a single neuron) or oppose (proportional to the difference in activation between two neurons).

A.1.8 Genetic Model

Parameter	Description
DieAtMaxAge	Whether an agent dies at a genetically specified maximum lifespan. Also used in conjunction with PassiveLockstep and AdaptivityMode .
EnableEvolution	Whether crossover and mutation are performed during reproduction. If false, specifies that offspring genomes are copied from a single parent chosen at random.
GeneInterpolation- Power	Exponents to apply when interpolating raw genetic values into their corresponding phenotypic ranges.
GeneticOperator- Resolution	Resolution at which crossover and mutation operate over genomes: bit or byte.
GenomeLayout	Method of ordering genes: groups (genes coding for a particular pair of neural groups are contiguous) or none (genes coding for a particular neural property are contiguous).
GrayCoding	Whether genes are Gray-coded such that successive values differ by one bit.
Max/MinCrossover- Points	Phenotypic range of the CrossoverPointCount gene.
Max/MinInitialBitProb	Maximum/minimum probability of turning on each bit in a randomized genome. Only applicable if GeneticOperatorResolution is bit.
Max/MinLifeSpan	Phenotypic range of the LifeSpan gene.
Max/MinMutationRate	Phenotypic range of the MutationRate gene.
Max/MinMutationStdev- Power	Phenotypic range of the MutationStdevPower gene. When GeneticOperatorResolution is byte, mutations are drawn from a normal distribution whose standard deviation is controlled by this gene.
MiscegenationDelay	Number of consecutive natural births that must occur without the GA being invoked before a miscegenation function is enabled. This function stochastically permits mating with a probability based on the parents' genetic dissimilarity (normalized between zero and one):

Parameter	Description
	$(\text{probability}) = \frac{1}{2} \times [1 + \text{sgn}(\text{biased dissimilarity}) \times (\text{biased dissimilarity}) ^{(\text{MiscegenationFunctionInverseSlope})}],$ $(\text{biased dissimilarity}) = \cos[\pi \times (\text{dissimilarity})^{(\text{MiscegenationFunctionBias})}].$ <p>If this parameter is negative, the miscegenation function is ignored, and genetic dissimilarity has no effect on mating.</p>
Miscegenation-FunctionBias	Used in the miscegenation function. Larger values result in more permissive miscegenation (i.e., higher probability of mating between genetically dissimilar parents). See MiscegenationDelay .
Miscegenation-FunctionInverseSlope	Used in the miscegenation function. Larger values extend the range of genetic dissimilarity over which the probability of successfully mating is 50%. See MiscegenationDelay .
RawSeedMutationRate	Mutation rate applied to agents in the initial population subsequent to any mutations due to SeedMutationProbability . This parameter is expressed as a raw probability and has no effect on the <code>MutationRate</code> gene.
SeedAgents	Number of agents in the initial population to seed according to SeedType . Other agents are given randomized genomes.
SeedMutation-Probability	Probability of mutating the genome of a seed agent.
SeedMutationRate	Normalized value of the <code>MutationRate</code> gene for agents in the initial population.
SeedType	Method of seeding genomes for agents in the initial population: legacy (neural wiring fosters adaptive behavior such as foraging, seeking mates for reproduction, and fleeing attack), simple (neural wiring is random or completely suppressed), or random (all genes—including those specifying neural wiring—are set randomly).

A.1.9 Recording

Parameter	Description
AdamiComplexity-RecordFrequency	Interval (i.e., number of timesteps) at which Adami et al.'s (2000) complexity is calculated.
CompressFiles	Whether to compress recorded data files.
EpochFrequency	Interval (i.e., number of timesteps) at which intermittent recording tasks are performed.
GenomeSubsetLog	Specific genes to record for each agent independently of any recording due to RecordGenomes .
RecordAdamiComplexity	Whether to record the complexity of the population at regular intervals. This parameter refers to the formulation of complexity due to Adami et al.'s (2000) Equation 4, an entropy-based measure of the amount of information stored in genomes population-wide.
RecordAgentEnergy	Whether to record the health and food energy levels of each agent at each timestep.
RecordAll	Default value of parameters that control whether data is recorded.
RecordBarrierPosition	Whether to record the positions of barriers at each timestep. This parameter is not currently used.
RecordBirthsDeaths	Whether to record the birth and death timestep of each agent.
RecordBrainAnatomy	Whether to record the weight matrix of each agent's neural network at each life stage (inception, birth, and death).
RecordBrainBestRecent	Whether to aggregate neural network data files at intervals of EpochFrequency for the list of fittest agents recently observed.
RecordBrainBestSoFar	Whether to aggregate neural network data files at intervals of EpochFrequency for the list of fittest agents ever observed.
RecordBrainFunction	Whether to record the neural activation time series of each agent.
RecordBrainRecent	Whether to aggregate neural network data files at intervals of EpochFrequency .

Parameter	Description
RecordBrain	Default value of parameters that control whether neural network data is recorded.
RecordCarry	Whether to record PickUp and Drop actions.
RecordCollisions	Whether to record collisions between agents and objects (including other agents).
RecordComplexity	Whether to record the complexity of each agent.
RecordContacts	Whether to record contacts (i.e., instances of collocation) between agents.
RecordEnergy	Whether to record energy modifications other than usage (e.g., eating, fighting, and giving).
RecordFoodConsumption	Whether to record food consumption.
RecordFoodEnergy	Whether to record the amount of food energy available worldwide at each timestep.
RecordFrequency	This parameter is not currently used.
RecordGeneStats	Whether to record the population-wide mean and standard deviation of each gene at each timestep.
RecordGenomes	Whether to record the genome of each agent.
RecordGitRevision	Whether to record the current Git revision.
RecordNeural-ComplexityFiles	Default value of parameters that control whether data required to calculate complexity is recorded.
RecordPopulation	Whether to record the population at each timestep.
RecordPosition	Type of position to record for each agent at each timestep: none, precise (three-dimensional position at full precision), or approximate (two-dimensional position rounded to two decimal places).
RecordSeparations	Pairs of agents for which to record genetic dissimilarity: none, contact (agents who are collocated at any point in their lifetimes), or all (agents who are alive at the same time, regardless of duration).
RecordSynapses	Whether to record synaptic weights and learning rates of each agent's neural network at each life stage.

A.2 Legacy

Experiments in the Legacy condition use Git revision [bdc7060](#) with the following values for `InitSeed`: 1526042077, 1526045390, 1526052688, 1526056391, 1526063118, 1526066882, 1526074523, 1526081-535, 1526088596, and 1526092287. Videos of typical driven and passive Legacy simulations are available at <https://youtu.be/FZq2TcyRTTo> and https://youtu.be/Stv0Y_F5-2c.

```
@version 2
# -----
# Simulation
# -----
LegacyMode True
MaxSteps 16000
StaticTimestepGeometry True

# -----
# Environment
# -----
Barriers [
  {
    X1 0.333; Z1 -1.0
    X2 0.333; Z2 -0.1
  },
  {
    X1 0.667; Z1 -1.0
    X2 0.667; Z2 -0.1
  }
]
Domains [
  {
    CenterX 0.5; CenterZ 0.5
    SizeX 1.0; SizeZ 1.0
    FoodPatches [
      {
        CenterX 0.5; CenterZ 0.05
        FoodFraction 0.2
        MaxFoodFraction 1.0
        MaxFoodGrownFraction 1.0
        SizeX 1.0; SizeZ 0.1
      },
      {
        CenterX 0.5; CenterZ 0.8
        FoodFraction 0.8
        MaxFoodFraction 1.0
        MaxFoodGrownFraction 1.0
        SizeX 1.0; SizeZ 0.4
      }
    ]
  }
]
RatioBarrierPositions True

# ----
# Food
# ----
FoodGrowthRate 0.1
InitFood MinFood

MinFood 90; MaxFood 120

# -----
# Population
# -----
ApplyLowPopulationAdvantage True
EndOnPopulationCrash True
InitAgents 180
MinAgents 90; MaxAgents 300
NumDepletionSteps 125

# -----
# Energetics
# -----
AgentsAreFood False
EnergyUseNeurons 0.0
EnergyUseSynapses 0.0
MinMateEnergyFraction 0.5

# -----
# Neural Model
# -----
MinExcitatoryNeuronsPerGroup 1
MaxExcitatoryNeuronsPerGroup 16
MinInhibitoryNeuronsPerGroup 1
MaxInhibitoryNeuronsPerGroup 16
MinInternalNeuralGroups 0
MaxInternalNeuralGroups 5
MaxSynapseWeight 8.0
MaxSynapseWeightInitial 2.0
MinVisionNeuronsPerGroup 1
MaxVisionNeuronsPerGroup 16

# -----
# Genetic Model
# -----
SeedAgents InitAgents

# -----
# Recording
# -----
RecordAgentEnergy True
RecordBirthsDeaths True
RecordBrainAnatomy True
RecordBrainFunction True
RecordFoodConsumption True
RecordFoodEnergy True
RecordGenomes True
RecordGitRevision True
RecordPopulation True
RecordSynapses True
```

A.3 Simple

Experiments in the Simple condition use Git revision [9057d6c](#) with the following values for `InitSeed`: 1527242168, 1527246773, 1527261565, 1527266522, 1527271502, 1527276865, 1527281377, 1527285783, 1527290133, and 1527298881. Video of a typical Simple simulation is available at https://youtu.be/U2wl_q73Wb8.

```
@version 2

# -----
# Simulation
# -----
MaxSteps 20000

# -----
# Environment
# -----
Barriers []
Domains [
  {
    CenterX 0.5; CenterZ 0.5
    SizeX 1.0; SizeZ 1.0
    FoodPatches [
      {
        FoodTypeName "Food"
        FoodFraction 1.0
        MaxFoodFraction 1.0
        MaxFoodGrownFraction 1.0
        FoodRate 0.5
      }
    ]
  }
]
FoodColor {
  R 0.0
  G 1.0
  B 0.0
}
GroundColor {
  R 0.0
  G 0.25
  B 0.0
}

# ----
# Food
# ----
FoodEnergySizeScale 500.0
FoodGrowthRate 0.0
FoodMaxLifeSpan 250
FoodTypes [
  {
    Name "Food"
    EnergyPolarity [ 1 ]
  }
]
InitFood int(50000.0 / MinFoodEnergy)
MinFood 0; MaxFood 10000
MinFoodEnergy 500.0
MaxFoodEnergy MinFoodEnergy

ProbabilisticFoodPatches False
RandomInitFoodAge True

# -----
# Population
# -----
EndOnPopulationCrash True
EnergyBasedPopulationControl False
InitAgents 200
MinAgents 50; MaxAgents 1000

# -----
# Energetics
# -----
AgeEnergyMultiplier 0.002
DamageRate 5.0
EnergyUseEat 0.01
EnergyUseFight (0.1 * DamageRate)
EnergyUseFixed 0.1
EnergyUseFocus 0.001
EnergyUseMate 0.1
EnergyUseMove 0.1
EnergyUseMultiplier dyn(1.0) {
  return pow((double)(AgentCount - MinAgents)
    ↪ / (InitAgents - MinAgents), 2.0);
}
EnergyUseNeurons 0.1
EnergyUseSynapses 0.1
EnergyUseTurn 0.1
MinAgentMaxEnergy 500.0
MaxAgentMaxEnergy MinAgentMaxEnergy
MinMateEnergyFraction 0.5
MinSizeEnergyPenalty 1.0
MaxSizeEnergyPenalty (MaxAgentSize /
  ↪ MinAgentSize)
RandomSeedEnergy True

# -----
# Physiology
# -----
BodyGreenChannel F
EnableMateWaitFeedback True
EnableSpeedFeedback True
InvertFocus True
InvertMateWaitFeedback True
MateWait 25
MinAgentMaxSpeed 1.0
MaxAgentMaxSpeed MinAgentMaxSpeed
MinAgentSize 1.0
MaxAgentSize MinAgentSize
MinAgentStrength 1.0
MaxAgentStrength MinAgentStrength
MinEnergyFractionToOffspring 0.5
```

```

MaxEnergyFractionToOffspring
↳ MinEnergyFractionToOffspring
MinHorizontalFieldOfView (2.0 *
↳ MaxVisionNeuronsPerGroup)
MaxHorizontalFieldOfView (20.0 *
↳ MaxVisionNeuronsPerGroup)
MaxSizeFightAdvantage MaxSizeEnergyPenalty
NoseColor B
ProbabilisticMating True
RandomSeedMateWait True
RetinaWidth (2 * MaxVisionNeuronsPerGroup)
YawRate MinHorizontalFieldOfView

# -----
# Neural Model
# -----
FixedInitSynapseWeight True
LearningMode Prebirth
MinExcitatoryNeuronsPerGroup 0
MaxExcitatoryNeuronsPerGroup
↳ MaxVisionNeuronsPerGroup
MinInhibitoryNeuronsPerGroup 0
MaxInhibitoryNeuronsPerGroup
↳ MaxVisionNeuronsPerGroup
MinInternalNeuralGroups 0
MaxInternalNeuralGroups 10
MinLearningRate 0.004
MaxLearningRate MinLearningRate
MaxSynapseWeight 5.0
MaxSynapseWeightInitial 0.5
MinVisionNeuronsPerGroup 7
MaxVisionNeuronsPerGroup
↳ MinVisionNeuronsPerGroup
MirroredTopologicalDistortion True
OrderedInternalNeuralGroups True

PreBirthCycles int((MaxSynapseWeight -
↳ MaxSynapseWeightInitial) /
↳ (MaxLearningRate * 0.25))
SimpleSeedYawBiasDelta (0.5 / MaxBiasWeight)
SynapseWeightDecayRate (1.0 - MaxLearningRate
↳ * 0.25 / MaxSynapseWeight)

# -----
# Genetic Model
# -----
DieAtMaxAge False
GeneticOperatorResolution Byte
MinCrossoverPoints 1
MaxCrossoverPoints MinCrossoverPoints
MinLifeSpan MaxSteps
MaxLifeSpan MinLifeSpan
MinMutationRate 1.0
MaxMutationRate MinMutationRate
MinMutationStdevPower 4.0
MaxMutationStdevPower MinMutationStdevPower
SeedType Simple

# -----
# Recording
# -----
RecordAgentEnergy True
RecordBirthsDeaths True
RecordBrainAnatomy True
RecordBrainFunction True
RecordFoodConsumption True
RecordFoodEnergy True
RecordGenomes True
RecordGitRevision True
RecordPopulation True
RecordSynapses True

```

A.4 Forage

Experiments in the Forage condition use Git revision [9057d6c](#) with the following values for `InitSeed`: 1527306702, 1527311032, 1527319589, 1527328103, 1527332361, 1527336705, 1527341091, 1527345610, 1527354670, and 1527359074. Parameters are identical to the `Simple` worldfile with the exceptions listed below. Video of a typical Forage simulation is available at <https://youtu.be/-U8EvRlz27U>.

```
# -----
# Environment
# -----
Domains [
  {
    CenterX 0.5; CenterZ 0.5
    SizeX 1.0; SizeZ 1.0
    FoodPatches [
      {
        FoodTypeName "Food"
        FoodFraction 1.0
        MaxFoodFraction 1.0
        MaxFoodGrownFraction 1.0
        FoodRate dyn(0.5) {
          return 250.0 / MinFoodEnergy;
        }
      }
    ]
  }
]

# ----
# Food
# ----
MinFoodEnergy dyn(500.0) {
  return Step < 10000 ? 500.0 + 4500.0 * Step
  ↪ / 10000.0 : 5000.0;
}
MaxFoodEnergy dyn(500.0) {
  return MinFoodEnergy;
}
```

A.5 Poison

Experiments in the Poison condition use Git revision [9057d6c](#) with the following values for `InitSeed`: 1527366504, 1527370385, 1527374321, 1527383672, 1527387405, 1527393989, 1527397289, 1527400496, 1527403970, and 1527407490. Parameters are identical to the `Forage` worldfile with the exceptions listed below. Video of a typical Poison simulation is available at <https://youtu.be/Gk3tBPuqmPY>.

```
# -----
# Environment
# -----
Domains [
  {
    CenterX 0.5; CenterZ 0.5
    SizeX 1.0; SizeZ 1.0
    FoodPatches [
      {
        FoodTypeName "Food"
        FoodFraction 0.5
        InitFoodFraction 1.0
        MaxFoodFraction 1.0
        MaxFoodGrownFraction 1.0
        FoodRate dyn(0.5) {
          return 250.0 / MinFoodEnergy;
        }
      },
      {
        FoodTypeName "Poison"
        FoodFraction 0.5
        InitFoodFraction 0.5
        MaxFoodFraction 1.0
        MaxFoodGrownFraction 1.0
        FoodRate 0.25
        FoodEnergy 1000.0
      }
    ]
  }
]

}
]
# ----
# Food
# ----
FoodTypes [
  {
    Name "Food"
    EnergyPolarity [ 1 ]
  },
  {
    Name "Poison"
    EnergyPolarity [ -1 ]
    EatMultiplier [
      dyn(-0.1) {
        return Step < 10000 ? -0.1 - 0.9 *
          ↪ Step / 10000.0 : -1.0;
      }
    ]
  }
]
FoodColor {
  R 1.0
  G 0.0
  B 0.0
}
```

Appendix B

Proof of Equation 3.9

Neural complexity C_N is defined as

$$C_N(X) = \sum_{k=1}^n \left[\frac{k}{n} M(X) - \langle M(X_i^k) \rangle_i \right], \quad (\text{B.1})$$

where multi-information M may be expressed in terms of the Shannon entropy H as

$$M(X) = \sum_{x_i \in X} H(x_i) - H(X). \quad (\text{B.2})$$

The penultimate ($k = n - 1$) term of neural complexity is given by

$$C_N^{n-1}(X) = \frac{n-1}{n} M(X) - \langle M(X_i^{n-1}) \rangle_i, \quad (\text{B.3})$$

where X_i^{n-1} denotes a subset of the system having size $n - 1$. There exist n such subsets, each comprising the entire system with the exception of a single element:

$$X_i^{n-1} = X \setminus \{x_i\}, \quad (\text{B.4})$$

allowing Equation B.3 to be rewritten as

$$C_N^{n-1}(X) = \frac{n-1}{n} M(X) - \langle M(X \setminus \{x_i\}) \rangle_i. \quad (\text{B.5})$$

By definition, the multi-information of subset X_i^{n-1} is

$$\begin{aligned}
M(X \setminus \{x_i\}) &= \sum_{\substack{x_j \in X \\ j \neq i}} H(x_j) - H(X \setminus \{x_i\}) \\
&= \sum_{x_j \in X} H(x_j) - H(x_i) - H(X \setminus \{x_i\}).
\end{aligned} \tag{B.6}$$

The average multi-information across all such subsets is

$$\begin{aligned}
\langle M(X \setminus \{x_i\}) \rangle_i &= \frac{1}{n} \sum_{x_i \in X} M(X \setminus \{x_i\}) \\
&= \frac{1}{n} \sum_{x_i \in X} \left[\sum_{x_j \in X} H(x_j) - H(x_i) - H(X \setminus \{x_i\}) \right] \\
&= \frac{1}{n} \left[\sum_{x_i \in X} \sum_{x_j \in X} H(x_j) - \sum_{x_i \in X} H(x_i) - \sum_{x_i \in X} H(X \setminus \{x_i\}) \right] \\
&= \frac{1}{n} \left[n \sum_{x_i \in X} H(x_i) - \sum_{x_i \in X} H(x_i) - \sum_{x_i \in X} H(X \setminus \{x_i\}) \right] \\
&= \frac{1}{n} \left[(n-1) \sum_{x_i \in X} H(x_i) - \sum_{x_i \in X} H(X \setminus \{x_i\}) \right].
\end{aligned} \tag{B.7}$$

Substituting Equations B.2 and B.7 into Equation B.5 yields

$$\begin{aligned}
C_N^{n-1}(X) &= \frac{n-1}{n} M(X) - \langle M(X \setminus \{x_i\}) \rangle_i \\
&= \frac{n-1}{n} \left[\sum_{x_i \in X} H(x_i) - H(X) \right] - \frac{1}{n} \left[(n-1) \sum_{x_i \in X} H(x_i) - \sum_{x_i \in X} H(X \setminus \{x_i\}) \right] \\
&= \frac{1}{n} \left[(n-1) \sum_{x_i \in X} H(x_i) - (n-1) H(X) \right] - \frac{1}{n} \left[(n-1) \sum_{x_i \in X} H(x_i) - \sum_{x_i \in X} H(X \setminus \{x_i\}) \right] \\
&= \frac{1}{n} \left[\cancel{(n-1) \sum_{x_i \in X} H(x_i)} - (n-1) H(X) - \cancel{(n-1) \sum_{x_i \in X} H(x_i)} + \sum_{x_i \in X} H(X \setminus \{x_i\}) \right] \\
&= \frac{1}{n} \left[\sum_{x_i \in X} H(X \setminus \{x_i\}) - (n-1) H(X) \right].
\end{aligned} \tag{B.8}$$

Now consider the interaction complexity C_I , defined as

$$C_I(X) = H(X) - \sum_{x_i \in X} H(x_i | X \setminus \{x_i\}). \quad (\text{B.9})$$

By the chain rule of conditional entropy,

$$H(x_i | X \setminus \{x_i\}) = H(X) - H(X \setminus \{x_i\}), \quad (\text{B.10})$$

allowing Equation B.9 to be rewritten as

$$\begin{aligned} C_I(X) &= H(X) - \sum_{x_i \in X} [H(X) - H(X \setminus \{x_i\})] \\ &= H(X) - \sum_{x_i \in X} H(X) + \sum_{x_i \in X} H(X \setminus \{x_i\}) \\ &= H(X) - n H(X) + \sum_{x_i \in X} H(X \setminus \{x_i\}) \\ &= (1 - n) H(X) + \sum_{x_i \in X} H(X \setminus \{x_i\}) \\ &= \sum_{x_i \in X} H(X \setminus \{x_i\}) - (n - 1) H(X). \end{aligned} \quad (\text{B.11})$$

This is identical to the bracketed term in Equation B.8:

$$\begin{aligned} C_N^{n-1}(X) &= \frac{1}{n} \left[\sum_{x_i \in X} H(X \setminus \{x_i\}) - (n - 1) H(X) \right] \\ &= \frac{1}{n} C_I(X), \end{aligned} \quad (\text{B.12})$$

thus proving Equation 3.9.

Bibliography

- Abbott, L. F. and Nelson, S. B. (2000). Synaptic plasticity: taming the beast. *Nature Neuroscience*, 3(11s):1178–1183.
- Abraham, R. H. and Shaw, C. D. (2000). *Dynamics: The Geometry of Behavior*. Aerial Press, fourth edition.
- Ackley, D. and Littman, M. (1991). Interactions between learning and evolution. In Langton, C. G. et al., editors, *Artificial Life II*, pages 487–509.
- Adami, C., Ofria, C., and Collier, T. C. (2000). Evolution of biological complexity. *Proceedings of the National Academy of Sciences*, 97(9):4463–4468.
- Albert, R., Jeong, H., and Barabási, A.-L. (1999). Diameter of the World-Wide Web. *Nature*, 401(6749):130–131.
- Barrat, A., Barthélemy, M., Pastor-Satorras, R., and Vespignani, A. (2004). The architecture of complex weighted networks. *Proceedings of the National Academy of Sciences*, 101(11):3747–3752.
- Barsalou, L. W. (1999). Perceptual symbol systems. *Behavioral and Brain Sciences*, 22(4):577–660.
- Bedau, M. A. (2007). Artificial life. In Matthen, M. and Stephens, C., editors, *Philosophy of Biology*, pages 585–604. Elsevier.
- Bedau, M. A., Snyder, E., and Packard, N. H. (1998). A classification of long-term evolutionary dynamics. In Adami, C. et al., editors, *Artificial Life VI*, pages 228–237.
- Beer, R. D. (1990). *Intelligence as Adaptive Behavior: An Experiment in Computational Neuroethology*. Academic Press.

- Beer, R. D. (1995). A dynamical systems perspective on agent-environment interaction. *Artificial Intelligence*, 72(1–2):173–215.
- Beer, R. D. (1997). The dynamics of adaptive behavior: A research program. *Robotics and Autonomous Systems*, 20(2–4):257–289.
- Bertschinger, N. and Natschläger, T. (2004). Real-time computation at the edge of chaos in recurrent neural networks. *Neural Computation*, 16(7):1413–1436.
- Braitenberg, V. (1984). *Vehicles: Experiments in Synthetic Psychology*. MIT Press.
- Brooks, R. A. (1985). A robust layered control system for a mobile robot. AI Memo 864, Massachusetts Institute of Technology.
- Brooks, R. A. (1986). Achieving artificial intelligence through building robots. AI Memo 899, Massachusetts Institute of Technology.
- Brooks, R. A. (1991). Intelligence without representation. *Artificial Intelligence*, 47(1–3):139–159.
- Chalmers, D. J. (2006). Strong and weak emergence. In Clayton, P. and Davies, P., editors, *The Re-Emergence of Emergence: The Emergentist Hypothesis from Science to Religion*, chapter 11, pages 244–256. Oxford University Press.
- Chomsky, N. (1965). *Aspects of the Theory of Syntax*. MIT Press.
- Clark, A. (1996). *Being There: Putting Brain, Body, and World Together Again*. MIT Press.
- Clark, A. (1997). The dynamical challenge. *Cognitive Science*, 21(4):461–481.
- Clark, A. and Chalmers, D. (1998). The extended mind. *Analysis*, 58(1):7–19.
- Cliff, D. (2003). Biologically-inspired computing approaches to cognitive systems: a partial tour of the literature. Technical Report HPL-2003-11, Hewlett-Packard Company.
- Conrad, M. and Pattee, H. H. (1970). Evolution experiments with an artificial ecosystem. *Journal of Theoretical Biology*, 28(3):393–409.
- Cover, T. M. and Thomas, J. A. (2006). *Elements of Information Theory*. Wiley, second edition.

- Dawkins, R. and Krebs, J. R. (1979). Arms races between and within species. *Proceedings of the Royal Society B: Biological Sciences*, 205(1161):489–511.
- Dennett, D. C. (1996). *Kinds of Minds: Toward an Understanding of Consciousness*. Basic Books.
- Downing, K. L. (2006). Combining evolutionary algorithms and neural networks. Retrieved from <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.100.5913>.
- Faes, L. and Porta, A. (2014). Conditional entropy-based evaluation of information dynamics in physiological systems. In Wibral, M. et al., editors, *Directed Information Measures in Neuroscience*, pages 61–86. Springer.
- Ferrer i Cancho, R. and Solé, R. V. (2001). The small world of human language. *Proceedings of the Royal Society B: Biological Sciences*, 268(1482):2261–2265.
- Fodor, J. A. (1975). *The Language of Thought*. Harvard University Press.
- Fretwell, S. D. and Lucas, Jr., H. L. (1969). On territorial behavior and other factors influencing habitat distribution in birds: I. Theoretical development. *Acta Biotheoretica*, 19(1):16–36.
- Fukai, T. and Tanaka, S. (1997). A simple neural network exhibiting selective activation of neuronal ensembles: From winner-take-all to winners-share-all. *Neural Computation*, 9(1):77–97.
- Gardner, M. (1970). The fantastic combinations of John Conway’s new solitaire game “life”. *Scientific American*, 223(4):120–123.
- Gardner, M. (1971). On cellular automata, self-reproduction, the Garden of Eden and the game “life”. *Scientific American*, 224(2):112–117.
- Gibson, J. J. (1979). *The Ecological Approach to Visual Perception*. Routledge.
- Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley.
- Goldberger, A. L., Rigney, D. R., and West, B. J. (1990). Chaos and fractals in human physiology. *Scientific American*, 262(2):43–49.
- Gould, S. J. (1994). The evolution of life on the earth. *Scientific American*, 271(4):84–91.

- Griffith, V. and Yaeger, L. S. (2006). Ideal free distribution in agents with evolved neural architectures. In Rocha, L. M. et al., editors, *Artificial Life X*, pages 372–378.
- Harnad, S. (1990). The symbol grounding problem. *Physica D*, 42(1–3):335–346.
- Hebb, D. O. (1949). *The Organization of Behavior: A Neuropsychological Theory*. Wiley.
- Hinton, G. E. (1992). How neural networks learn from experience. *Scientific American*, 267(3):144–151.
- Hinton, G. E., McClelland, J. L., and Rumelhart, D. E. (1986). Distributed representations. In Rumelhart, D. E. et al., editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*, chapter 3, pages 77–109. MIT Press.
- Hinton, G. E. and Nowlan, S. J. (1987). How learning can guide evolution. *Complex Systems*, 1(3):495–502.
- Hodgkin, A. L. and Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, 117(4):500–544.
- Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. University of Michigan Press.
- Holland, J. H. (1994). Echoing emergence: Objectives, rough definitions, and speculations for Echo-class models. In Cowan, G. A. et al., editors, *Complexity: Metaphors, Models, and Reality*, pages 309–342. Addison-Wesley.
- Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558.
- Horgan, J. (1995). From complexity to perplexity. *Scientific American*, 272(6):104–109.
- Hurst, L. D. (2002). The K_a/K_s ratio: diagnosing the form of sequence evolution. *Trends in Genetics*, 18(9):486–487.
- Izhikevich, E. M. (2003). Simple model of spiking neurons. *IEEE Transactions on Neural Networks*, 14(6):1569–1572.
- Kauffman, S. A. and Johnsen, S. (1991). Coevolution to the edge of chaos: Coupled fitness landscapes, poised states, and coevolutionary avalanches. *Journal of Theoretical Biology*, 149(4):467–505.

- Kirkpatrick, S., Gelatt, Jr., C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.
- Kitano, H. (1990). Designing neural networks using genetic algorithms with graph generation system. *Complex Systems*, 4(4):461–476.
- Kozachenko, L. F. and Leonenko, N. N. (1987). Sample estimate of the entropy of a random vector. *Problems of Information Transmission*, 23(2):95–101.
- Kraskov, A., Stögbauer, H., and Grassberger, P. (2004). Estimating mutual information. *Physical Review E*, 69(6).
- Langton, C. G. (1989). Artificial life. In Langton, C. G., editor, *Artificial Life*, pages 1–47.
- Langton, C. G. (1990). Computation at the edge of chaos: Phase transitions and emergent computation. *Physica D*, 42(1–3):12–37.
- Latora, V. and Marchiori, M. (2001). Efficient behavior of small-world networks. *Physical Review Letters*, 87(19).
- Lizier, J. T. (2014). JIDT: an information-theoretic toolkit for studying the dynamics of complex systems. *Frontiers in Robotics and AI*, 1.
- Lizier, J. T., Prokopenko, M., and Zomaya, A. Y. (2010). Information modification and particle collisions in distributed computation. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 20(3).
- Lizier, J. T., Prokopenko, M., and Zomaya, A. Y. (2014). A framework for the local information dynamics of distributed computation in complex systems. In Prokopenko, M., editor, *Guided Self-Organization: Inception*, chapter 5, pages 115–158. Springer.
- Lungarella, M. and Sporns, O. (2006). Mapping information flow in sensorimotor networks. *PLOS Computational Biology*, 2(10).
- McClelland, J. L., Rumelhart, D. E., and the PDP Research Group (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 2: Psychological and Biological Models*. MIT Press.

- McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5(4):115–133.
- McShea, D. W. (1991). Complexity and evolution: What everybody knows. *Biology and Philosophy*, 6(3):303–324.
- McShea, D. W. (1994). Mechanisms of large-scale evolutionary trends. *Evolution: International Journal of Organic Evolution*, 48(6):1747–1763.
- McShea, D. W. and Hordijk, W. (2013). Complexity by subtraction. *Evolutionary Biology*, 40(4):504–520.
- Minsky, M. L. and Papert, S. A. (1969). *Perceptrons: An Introduction to Computational Geometry*. MIT Press.
- Mitchell, M., Hraber, P. T., and Crutchfield, J. P. (1993). Revisiting the edge of chaos: Evolving cellular automata to perform computations. *Complex Systems*, 7(2):89–130.
- Montana, D. J. and Davis, L. (1989). Training feedforward neural networks using genetic algorithms. In Sridharan, N. S., editor, *IJCAI-89*, pages 762–767.
- Moriarty, D. E. and Miikkulainen, R. (1997). Forming neural networks through efficient and adaptive coevolution. *Evolutionary Computation*, 5(4):373–399.
- Murdock, J. and Yaeger, L. S. (2011). Identifying species by genetic clustering. In Lenaerts, T. et al., editors, *Advances in Artificial Life, ECAL 2011*, pages 564–572.
- Newell, A. and Simon, H. A. (1976). Computer science as empirical inquiry: Symbols and search. *Communications of the ACM*, 19(3):113–126.
- Newman, M. E. J. (2001). The structure of scientific collaboration networks. *Proceedings of the National Academy of Sciences*, 98(2):404–409.
- Newman, M. E. J. (2010). *Networks: An Introduction*. Oxford University Press.
- Nilsson, N. J. (2007). The physical symbol system hypothesis: Status and prospects. In Lungarella, M. et al., editors, *50 Years of Artificial Intelligence: Essays Dedicated to the 50th Anniversary of Artificial Intelligence*, pages 9–17. Springer.

- Oka, M. and Ikegami, T. (2013). Exploring default mode and information flow on the web. *PLOS ONE*, 8(4).
- Packard, N. H. (1988). Adaptation toward the edge of chaos. In Kelso, J. A. S. et al., editors, *Dynamic Patterns in Complex Systems*, pages 293–301.
- Packard, N. H. (1989). Intrinsic adaptation in a simple model for evolution. In Langton, C. G., editor, *Artificial Life*, pages 141–155.
- Packard, N. H. and Wolfram, S. (1985). Two-dimensional cellular automata. *Journal of Statistical Physics*, 38(5/6):901–946.
- Pattee, H. H. (1989). Simulations, realizations, and theories of life. In Langton, C. G., editor, *Artificial Life*, pages 63–77.
- Ray, T. S. (1991). An approach to the synthesis of life. In Langton, C. G. et al., editors, *Artificial Life II*, pages 371–408.
- Reynolds, C. W. (1987). Flocks, herds, and schools: A distributed behavioral model. In Stone, M. C., editor, *SIGGRAPH '87*, pages 25–34.
- Rosenblatt, F. (1962). *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan Books.
- Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986a). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.
- Rumelhart, D. E., McClelland, J. L., and the PDP Research Group (1986b). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*. MIT Press.
- Sandoval, Jr., L. (2014). Structure of a global network of financial companies based on transfer entropy. *Entropy*, 16(8):4443–4482.
- Schank, R. and Abelson, R. (1977). *Scripts, Plans, Goals, and Understanding: An Inquiry into Human Knowledge Structures*. Psychology Press.
- Schiff, S. J., Jerger, K., Duong, D. H., Chang, T., Spano, M. L., and Ditto, W. L. (1994). Controlling chaos in the brain. *Nature*, 370(6491):615–620.

- Schreiber, T. (2000). Measuring information transfer. *Physical Review Letters*, 85(2):461–464.
- Searle, J. R. (1980). Minds, brains, and programs. *Behavioral and Brain Sciences*, 3(3):417–457.
- Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423,623–656.
- Sims, K. (1994a). Evolving virtual creatures. In Schweitzer, D. et al., editors, *SIGGRAPH '94*, pages 15–22.
- Sims, K. (1994b). Evolving 3D morphology and behavior by competition. In Brooks, R. A. and Maes, P., editors, *Artificial Life IV*, pages 28–39.
- Skarda, C. A. and Freeman, W. J. (1987). How brains make chaos in order to make sense of the world. *Behavioral and Brain Sciences*, 10(2):161–195.
- Sporns, O. and Zwi, J. D. (2004). The small world of the cerebral cortex. *Neuroinformatics*, 2(2):145–162.
- Sprott, J. C. (2003). *Chaos and Time-Series Analysis*. Oxford University Press.
- Stanley, K. O. and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127.
- Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. MIT Press.
- Tononi, G., Edelman, G. M., and Sporns, O. (1998). Complexity and coherency: integrating information in the brain. *Trends in Cognitive Sciences*, 2(12):474–484.
- Tononi, G., Sporns, O., and Edelman, G. M. (1994). A measure for brain complexity: Relating functional segregation and integration in the nervous system. *Proceedings of the National Academy of Sciences*, 91(11):5033–5037.
- Ulam, S. (1962). On some mathematical problems connected with patterns of growth of figures. In Bellman, R. E., editor, *Mathematical Problems in the Biological Sciences*, pages 215–224.
- van Gelder, T. (1995). What might cognition be, if not computation? *The Journal of Philosophy*, 92(7):345–381.
- Vincente, R., Wibral, M., Lindner, M., and Pipa, G. (2011). Transfer entropy—a model-free measure of effective connectivity for the neurosciences. *Journal of Computational Neuroscience*, 30(1):45–67.

- von Neumann, J. (1966). *Theory of Self-Reproducing Automata*. University of Illinois Press. Edited and completed by Burks, A. W.
- Wagner, A. and Fell, D. A. (2001). The small world inside large metabolic networks. *Proceedings of the Royal Society B: Biological Sciences*, 268(1478):1803–1810.
- Walter, W. G. (1950). An imitation of life. *Scientific American*, 182(5):42–45.
- Walter, W. G. (1951). A machine that learns. *Scientific American*, 185(2):60–63.
- Watts, D. J. and Strogatz, S. H. (1998). Collective dynamics of ‘small-world’ networks. *Nature*, 393(6684): 440–442.
- Weizenbaum, J. (1966). ELIZA—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45.
- Wiener, N. (1948). *Cybernetics: or Control and Communication in the Animal and the Machine*. Wiley.
- Williams, S. and Yaeger, L. (2017). Evolution of neural dynamics in an ecological model. *Geosciences*, 7(3).
- Wolfram, S. (1983). Cellular automata. *Los Alamos Science*, 9:2–21.
- Yaeger, L. (1994). Computational genetics, physiology, metabolism, neural systems, learning, vision, and behavior or PolyWorld: Life in a new context. In Langton, C. G., editor, *Artificial Life III*, pages 263–298.
- Yaeger, L., Griffith, V., and Sporns, O. (2008). Passive and driven trends in the evolution of complexity. In Bullock, S. et al., editors, *Artificial Life XI*, pages 725–732.
- Yaeger, L., Sporns, O., Williams, S., Shuai, X., and Dougherty, S. (2010). Evolutionary selection of network structure and function. In Fellersmann, H. et al., editors, *Artificial Life XII*, pages 313–320.
- Yaeger, L. S. (2009). How evolution guides complexity. *HFSP Journal*, 3(5):328–339.
- Yaeger, L. S. and Sporns, O. (2006). Evolution of neural structure and complexity in a computational ecology. In Rocha, L. M. et al., editors, *Artificial Life X*, pages 330–336.

CURRICULUM VITAE

Steven C. Williams

- Education**
- Indiana University, Bloomington, IN 8/2009 to 10/2019
- Doctor of Philosophy in Cognitive Science and Informatics
- West Virginia University, Morgantown, WV 8/2002 to 12/2006
- Bachelor of Science in Civil Engineering
 - Bachelor of Science in Computer Science
- Career**
- NIOSH (contractor), Morgantown, WV
- Software developer 1/2007 to present
- Indiana University, Bloomington, IN
- Instructor 6/2013 to 7/2013
 - Associate instructor 8/2012 to 5/2013
 - Research assistant 8/2009 to 5/2010
- Publications**
6. Williams, S. and Yaeger, L. (2017). Evolution of neural dynamics in an ecological model. *Geosciences*, 7(3).
 5. Menzies, T., Brady, A., Keung, J., Hihn, J., Williams, S., El-Rawas, O., Green, P., and Boehm, B. (2013). Learning project management decisions: A case study with case-based reasoning versus data farming. *IEEE Transactions on Software Engineering*, 39(12):1698–1713.
 4. Yaeger, L., Sporns, O., Williams, S., Shuai, X., and Dougherty, S. (2010). Evolutionary selection of network structure and function. In Fellersmann, H. et al., editors, *Artificial Life XII*, pages 313–320.
 3. Green II, P., Menzies, T., Williams, S., and El-Rawas, O. (2009). Understanding the value of software engineering technologies. In *ASE 2009*, pages 52–61.

2. Menzies, T., Williams, S., Elrawas, O., Baker, D., Boehm, B., Hihn, J., Lum, K., and Madachy, R. (2009). Accurate estimates without local data? *Software Process: Improvement and Practice*, 14(4):213–225.
1. Menzies, T., Williams, S., El-Rawas, O., Boehm, B., and Hihn, J. (2009). How to avoid drastic software process change (using stochastic stability). In *ICSE '09*, pages 540–550.

Presentations

5. Williams, S. (2013). Modeling at the interface of dynamics and computation. Presented at the IGERT Research Showcase, Bloomington, IN.
4. Williams, S. (2012). Modeling minimally abstract cognition. Presented at the IGERT Research Showcase, Bloomington, IN.
3. Williams, S. (2012). Evolved neural networks approach the edge of chaos. Presented at the Intelligent Systems Seminar, Bloomington, IN.
2. Williams, S. (2011). Analysis and design of two artificial life models. Presented at the IGERT Research Showcase, Bloomington, IN.
1. Williams, S. and Yaeger, L. (2010). Dynamical systems analysis of evolving neural networks. Presented at Guided Self-Organization 3, Bloomington, IN.

Awards

National Science Foundation Integrative Graduate Education and Research Traineeship:
The Dynamics of Brain–Body–Environment Systems in Behavior and Cognition

- Trainee 8/2010 to 8/2012
- Associate 5/2010 to 8/2010