2019

Quality Diversity: Harnessing Evolution to Generate a Diversity of High-Performing Solutions

Justin Pugh
University of Central Florida

QUALITY DIVERSITY: HARNESSING EVOLUTION TO GENERATE A DIVERSITY OF HIGH-PERFORMING SOLUTIONS

by

JUSTIN K. PUGH
B.S. of Computer Science, University of Central Florida, 2012
M.S. of Computer Science, University of Central Florida, 2018

A dissertation submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in the Department of Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Spring Term
2019

Major Professor: Kenneth O. Stanley

ABSTRACT

Evolution in nature has designed countless solutions to innumerable interconnected problems, giving birth to the impressive array of complex modern life observed today. Inspired by this success, the practice of evolutionary computation (EC) abstracts evolution artificially as a search operator to find solutions to problems of interest primarily through the adaptive mechanism of *survival of the fittest*, where stronger candidates are pursued at the expense of weaker ones until a solution of satisfying quality emerges. At the same time, research in open-ended evolution (OEE) draws different lessons from nature, seeking to identify and recreate processes that lead to the type of perpetual innovation and indefinitely increasing complexity observed in natural evolution. New algorithms in EC such as MAP-Elites and Novelty Search with Local Competition harness the toolkit of evolution for a related purpose: finding *as many types of good solutions as possible* (rather than merely the single best solution). With the field in its infancy, no empirical studies previously existed comparing these so-called *quality diversity* (QD) algorithms. This dissertation (1) contains the first extensive and methodical effort to compare different approaches to QD (including both existing published approaches as well as some new methods presented for the first time here) and to understand how they operate to help inform better approaches in the future. It also (2) introduces a new technique for encoding neural networks for evolution with indirect encoding that contain multiple sensory or output modalities. Further, it (3) explores the idea that QD can act as an engine of open-ended discovery by introducing an expressive platform called Voxelbuild where QD algorithms continually evolve robots that stack blocks in new ways. A culminating experiment (4) is presented that investigates evolution in Voxelbuild over a very long timescale. This research thus stands to advance the OEE community's desire to create and understand open-ended systems while also laying the groundwork for QD to realize its potential within EC as a means to automatically generate an endless progression of new content in real-world applications.

For Hailey

x

ACKNOWLEDGMENTS

First, I would like to express my sincerest appreciation to my advisor Dr. Kenneth O. Stanley who, through countless hours of dedicated time and attention, improved my writing ability and helped direct my curious analytical mind towards cutting edge scientific research. Without his direction, I would not have been capable of producing this dissertation.

Special thanks to my parents who encouraged me to pursue my passion for learning and problem solving by pursuing a bachelor's degree in computer science here at the University of Central Florida. Unbeknownst to them at the time, this would set me on a ten year track culminating in a master's and doctoral degree as my passion refined and intensified over time. They also supported me through a great deal of ups and downs as I walked through this entire process.

I extend my gratitude to my other committee members Dr. Annie S. Wu, Dr. Gita Sukthankar, and Dr. Ivan Garibay. Their oversight and feedback helped improve the quality of this dissertation.

Thanks also to past and present members of the Evolutionary Complexity Research Group (Eplex), in particular Dr. Paul Szerlip, Dr. Lisa Soros, Gregory Morse, Navid Kardan, and Jonathan Brant. The thoughtful and interesting discussions we have had over the years have helped me sort through and develop fresh ideas into those worthy of productive investigation, many of which have led to publication. Their attention and constructive criticism has also improved my presentation abilities.

Finally, thank you to the undergraduate students I have worked with and helped supervise over the years: Joshua Bowren, Rafaela Frota, and Kevin Negy. Their input and involvement has led to several publications and it is my hope that their time here at Eplex has helped them begin productive careers as they move forward.

Much of the research in this dissertation was supported in part by the National Science Foundation

under grant no. IIS-1421925 (Chapters 6, 7, 8, 9, and 10). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

The work in this dissertation was also supported in part through a grant from the US Army Research Office (Award No. W911NF-11-1-0489) (Chapter 4). This publication does not necessarily reflect the position or policy of the government, and no official endorsement should be inferred.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

The products of nature have long served as inspiration for the investigation and practice of evolutionary algorithms and evolutionary robotics [17, 87, 115]. Yet the ability of such algorithms to match the complexity and sophistication of nature has frustratingly lagged, as researchers in the fields often observe [31, 117]. This observation then often becomes the motivation for developing more sophisticated algorithms and encodings. Yet even more fundamental than the question of how to abstract the most brilliant achievements of nature into an algorithm of commensurate power is a question less often explicitly asked: *for what practical purpose is evolution actually suited anyway?*

Until recently, throughout a large swath of the field, the implicit yet resounding answer to this question has been *objective optimization* [10, 28, 75]. Various early pioneers in evolutionary computation independently inferred from their observations of nature that evolution can serve when abstracted artificially as a powerful optimization algorithm [36, 39, 40, 48, 106]. The discoveries of evolution in nature, such as the flight of birds or the intelligence of the human brain, suggested to these pioneers that if fitness pressure is calibrated to push selection towards an ambitious objective then evolution can become a tool of directed design and creation.

The casting of evolution as an algorithm for optimization naturally pitted evolutionary computation against the many subfields of machine learning invested in optimization, leading to conflicts and critiques, sometimes portraying evolutionary computation as ad hoc, unprincipled, less effective than other more theoretically-based optimization methods, or even outdated (and therefore often given less room in modern texts on machine learning) [2, 10]. While sometimes overly harsh or uninformed, even if such critiques are accepted, the perplexing question still hangs in the background of how it is possible that evolution in nature has managed to produce artifacts far beyond

the capacity of *any* subfield of machine learning or optimization. If evolution is apparently so unmatched in power in nature then why is there even a debate about its ability to compete with other approaches to optimization?

While one possible answer is that we have yet to uncover the deepest principles that unlock its true potential as an objective optimizer, a more intriguing possibility is that the real virtue of evolution is not in the end optimization at all. This suggestion goes beyond Herb Simon's assertion that evolution is a *satisficer* rather than an optimizer [107], which casts evolution almost as a poor man's optimizer. Rather, the hypothesis is that evolution is indeed phenomenally virtuosic at *something*, but that something is simply not optimization. This perspective can help to explain how it could be possible for evolution to produce sensational results in nature yet frustratingly modest ones in computation: *we may be using it wrong*. Perhaps the analogy with optimization was a mistake.

Indeed, it is difficult to imagine that evolution in nature is structured in the same way as a conventional optimization algorithm: there is no obvious unifying objective and organisms are often rewarded for being different in addition to being better. For example, organisms that are sufficiently different from their predecessors may establish a new *niche* in which they enjoy greatly reduced competition and thus are more likely to survive [54, 63]. Contrary to the tendency of optimization algorithms to converge over time to a single "best" solution, natural evolution instead exhibits a remarkable tendency towards *divergence* – continually accumulating myriad different ways of being. This observation is the crux of an alternative perspective in evolutionary computation (EC) that has been gaining momentum in recent years: *evolution as a machine for diversification* rather than optimization.

Inspired by this alternate view of natural evolution's apparent strength as a diversifier, a new evolutionary algorithm called novelty search (NS) [59, 61] was introduced, which searches only for behavioral diversity without any underlying objective pressure (that is, it tries to find as many dif-

ferent behaviors as possible but does not favor any particular behavior over another). Surprisingly, in some domains (particularly those that are deceptive) NS quickly finds the global optimum even when objective-based approaches consistently fail. The counterintuitive result that NS can sometimes find the best solutions without explicitly searching for them has since sparked considerable research interest in applying NS and methods like it to solving problems that were previously considered to be too difficult [22, 42, 43, 55, 59–62, 68, 72, 77, 80, 81, 101]. Novelty search has effectively demonstrated that evolution's talent for diversification can itself be harnessed as a powerful tool for optimization, now contending with the conventional notion of "survival of the fittest" as the most salient feature of natural evolution to be abstracted in algorithmic form. However, this view ignores the intrinsic value of diversity itself, treating it merely as a "means to an end" of finding the global optimum as usual.

In a true departure from conventional optimization, which seeks only the single best-performing solution, a new search paradigm has begun to emerge within EC where the effort focuses instead on finding a wide variety of viable solutions, similarly to how evolution in nature has discovered over billions of years a vast assortment of unique species, each of which are capable of orchestrating the complex system of biological processes necessary to sustain life. More precisely, the goal of this new type of search, called *quality diversity* (QD), is to find a maximally diverse collection of individuals in which each member is as high-performing as possible. In service of this goal, QD algorithms such as novelty search with local competition (NSLC) [62] and MAP-Elites [78] carefully balance a drive towards increasing diversity with localized searches for quality in an analogy with nature where species face the strongest competition from within their own niche while new niches are continually created and discovered. In this way, search can move towards different behaviors while simultaneously improving behaviors that have already been discovered.

Although QD algorithms can be applied to traditional optimization-oriented tasks where they may even perform well due to their ability to overcome the problem of deception (a claim verified by

experiments in this dissertation where various QD algorithms find many solutions to various maze tasks that traditional fitness-based strategies fail to solve altogether), the deeper promise of quality diversity is to push beyond what is possible through simple optimization. In particular, QD has potential applications in the areas of computational creativity [12] and open-ended evolution [3, 112], where the hope is to automatically generate an endless procession of uniquely interesting artifacts. In more restricted search spaces where the different types of artifacts are well-defined beforehand, QD algorithms have been called "illumination algorithms" because they effectively reveal the best possible performance achievable in each region of this phenotype space [78]. Applied to more unbounded and creative tasks, QD may illuminate remarkable artifacts that we have yet to imagine.

Importantly, the types of problems inspired by QD (several of which are discussed in the next chapter) favor approaches that explore many promising directions at the same time and thus represent a new opportunity for evolutionary algorithms to rise to prominence through a more unique profile within the broader machine learning community where focused single-solution approaches such as backpropagation [103] and support vector machines [20] have historically dominated.

To help accelerate research in this emerging area, one of the the aims of this dissertation is to establish a standard framework for understanding and comparing different approaches to searching for QD. The hope is to unify early works in this emerging field and to promote the design of better QD algorithms in the future. To that end, a benchmark domain is introduced in the form of a series of maze-navigation tasks of varying difficulty paired with a quantifiable measure of the performance of QD algorithms called the *QD-score*. Experimental results in these benchmarks comparing current state-of-the-art approaches as well as several novel variants thereof reveal important insights into the application of QD algorithms that extend beyond individual methods.

Initial experiments examine a critical component shared by all QD algorithms that defines the space

that the "diversity" component of QD seeks to explore – the *behavior characterization* (BC). As results on a maze task show, some choices of characterization can make finding QD more difficult, which on sufficiently-deceptive problems can translate into an inability to find the best solutions altogether. An important quality of BCs called *alignment* is identified, informing which BCs may be most suitable for driving search towards higher quality solutions.

However, as observed in the literature, QD researchers are often interested in finding QD with respect to a non-optimal characterization (i.e. one that inhibits finding the best-performing individuals). Standard practice currently suggests driving search with the same notion of diversity that you are ultimately interested in discovering [22, 62, 78, 122]. Experiments on significantly more complex and deceptive mazes confirm that current standard practice in QD is inadequate for the most challenging problems. Further experiments explore the idea of driving search with one BC while collecting QD with respect to another. Additionally, several new QD algorithms are introduced and then empirically validated for driving search with multiple BCs at the same time, thereby allowing the collection of QD with respect to any arbitrary characterization of diversity without compromising the ability to find the best solutions.

These new multi-characterization approaches, together with an increased understanding of the types of characterizations that are ideal for driving search effectively, enable the successful application of QD algorithms to a breadth of more difficult and interesting domains than the maze navigation benchmarks presented herein.

Equipped with this new knowledge, this dissertation additionally seeks to push quality diversity algorithms towards more "open-ended" tasks where evolution's strength as a diversifier may potentially be harnessed as an endless source of discovery. In service of this goal, a new domain called Voxelbuild is introduced, based on the popular Minecraft[1] video game that is well-known

[1]Copyright (c) 2011 Mojang

for facilitating seemingly boundless creativity for human players. The hope is that the complexity and impressiveness of artifacts generated by artificial agents in Voxelbuild will be intuitively appreciable by non-experts just as human-made artifacts in Minecraft can be visually appreciated even by people not familiar with the game.

The successful realization of Voxelbuild as an experimental platform includes the capacity to support complex multimodal agents with a wide range of ways to sense and interact with their world. However, the conventional approach (called HyperNEAT [120]) for evolving the type of large and regularly-structured neural network controllers found in Voxelbuild agents often struggles to effectively handle such multimodality. Addressing this challenge, a new technique for encoding HyperNEAT networks is introduced that makes them both more effective and easier to design (as demonstrated by experiments on an unrelated multi-agent coordination task) by placing different types of inputs and outputs in their own separate geometric spaces. This new technique not only enables the agent architectures featured herein but also facilitates the addition of even more modalities in the future.

Initial experiments in Voxelbuild demonstrate the ability of QD algorithms to induce a form of *major evolutionary transition* (one of the hallmarks of open-endedness) where the appearance of certain traits in the gene pool trigger brief bursts of innovation and complexification, akin to the Cambrian explosion observed in Earth's evolutionary timeline half a billion years ago. A final long-running experiment further explores the potential of the Voxelbuild domain with a refined approach in pursuit of more advanced "human-like" results. This capstone investigation reveals the behavior of QD algorithms set forth in this dissertation at longer timescales through a detailed analysis of runs an order of magnitude longer than previously studied, spanning months of computation time on a several hundred core computing cluster. Together, these experiments help establish quality diversity evolution as a tool for open-ended discovery and Voxelbuild as a promising platform for experiments in artificial creativity.

In summary, this dissertation encompasses four key contributions to the field: (1) a thorough benchmark comparison of different QD algorithms resulting in guidance on the best ways to apply them, together with a suite of new approaches designed to expand the viability of QD to a wider variety of more difficult problems, (2) a new technique for designing the layout of neural networks in the HyperNEAT neuroevolution encoding that helps cope with controller architectures that have many different sensor and effector modalities, (3) the conception of the Voxelbuild open-ended creative task that provides a platform for exploring QD's potential as a source of perpetual innovation, and (4) a detailed investigation into the behavior of QD algorithms over very long time-scales, highlighting their ability to generate meaningful increases to complexity.

Nature has discovered organisms both diverse and highly optimized within their own niches. This kind of divergent creative phenomenon differs from the typical convergent objective-driven process seen in search algorithms across machine learning, evolutionary computation (EC), and evolutionary robotics. By bringing QD now to the forefront of EC and providing a framework for understanding and comparing its available algorithms as well as demonstrating its long-term effectiveness in a visually-appreciable creative task, the hope is that the field can progress more confidently from this initial foundation and move towards the ambitious goal in AI of open-ended discovery where evolution is uniquely situated to excel.

This dissertation continues in Chapter 2 with an introduction to the NEAT and HyperNEAT neuroevolution techniques, a survey of other work in learning neural controllers for robot agents, a review of work on multi-solution and divergent search techniques leading to the emergence of quality diversity, and an overview of the field of open-ended evolution. Chapter 3 introduces a new approach enabling HyperNEAT to evolve robot controllers with many sensory modalities that is then validated empirically in Chapter 4. Chapter 5 describes existing QD algorithms and introduces a suite of new multi-BC approaches designed to expand the reach of QD to more difficult problems and broader notions of diversity. Chapter 6 presents an investigation into the effect of

BC alignment on QD followed by an extensive study in Chapter 7 comparing different QD algorithms on several difficult maze-navigation tasks, demonstrating the ability of the new multi-BC approaches to cope with unaligned notions of diversity. Chapter 8 presents the new Voxelbuild domain along with a preliminary experiment demonstrating its potential for open-endedness. Chapter 9 describes a series of improvements to Voxelbuild enabling a final culminating experiment in Chapter 10 examining QD's ability to generate open-ended innovation over a long evolutionary timescale. Chapter 11 discusses the implications of the work as a whole followed by a conclusion in Chapter 12.

CHAPTER 2: BACKGROUND

This chapter contains a review of literature related to the content of this dissertation. Presented first is a brief introduction to neuroevolution and a review of the specific neuroevolution methods employed by experiments throughout this dissertation, followed by a general overview of other work evolving neural controllers for simulated robots with a focus on the problem of multimodality (which is addressed by a new approach introduced in Chapter 3). Then, much of this chapter describes research leading to the emergence of quality diversity within the evolutionary computation (EC) community. Finally, the chapter closes with a brief introduction to research on open-ended evolution necessary to contextualize how quality diversity stands to bridge the gap between open-endedness and the types of practical applications typically addressed by EC.

Neuroevolution: NEAT and HyperNEAT

Neuroevolution as a field stretches back decades (Yao [132] provides a good historical review). All of the experiments in the following chapters involve the application of artificial neural networks (ANNs) to control tasks (such as controlling how a robot drives through a maze). While the inputs and outputs to an ANN are evident from the definition of each task (e.g. each robot sensor corresponds to its own input node and each output node controls an effector that determines how the robot acts within its environment), the challenge is to find an appropriate intermediate structure of hidden nodes and connections (along with their corresponding weights) to solve the task as best as possible. This work employs a method called *neuroevolution* to search for effective ANNs to solve each task.

In neuroevolution, a population of candidate ANNs attempts to solve the task and each ANN is

assigned a score (usually commensurate with its performance on the task, in which case the score is called *fitness*). Parent ANNs are randomly selected (with replacement) from the population with a preference for higher scores and then mutated to form a new population. This process is repeated iteratively, with each iteration (or *generation*) discovering beneficial network mutations and discarding detrimental ones in search of increasingly higher-scoring ANNs.

Early methods focused on evolving only the weights of fixed-topology networks [27], while later approaches introduced a broad variety of ideas for evolving topologies [1, 47, 95]. However, for many such methods the right way to combine networks of different topologies for crossover remained a challenge [96]. The NEAT algorithm, described next, provided a potential solution.

Neuroevolution of Augmenting Topologies (NEAT)

One of the most popular modern neuroevolution algorithms is called NEAT (*Neuroevolution of Augmenting Topologies*) [116, 118]. NEAT evolves increasingly large ANNs by starting with a population of simple networks that then *increase in complexity* over generations by adding new nodes and connections through random mutations. By evolving ANNs in this way, the topology (structure) of the network does not need to be known a priori; NEAT searches through increasingly complex networks to find a suitable level of complexity. Because it starts simply and gradually adds complexity, it tends to find a solution network close to the minimal necessary size. However, the direct representation of nodes and connections in the NEAT genome (i.e. every node and connection is mutated independently) cannot scale up to large brain-like networks nor can it take advantage of domain geometry such as the relative location of sensors and effectors. This makes it competitive for problems with few inputs and outputs that require only modestly-sized networks (such as the maze-navigation tasks in Chapter 6 and Chapter 7) but potentially ineffective for large sensor or effector arrays (e.g. visual fields like those in Chapter 8). For a complete overview of

NEAT, see Stanley and Miikkulainen [116] or Stanley and Miikkulainen [118].

HyperNEAT

Neuroevolution methods like NEAT are *directly encoded*, which means each component of the phenotype is encoded by a single gene, making the discovery of repeating (e.g. multiple functional legs in an evolved morphology) or symmetrical (e.g. developing legs on the left side of a morphology that look and function similar to those on the right side) motifs expensive and improbable. Therefore, indirect encodings [9, 14, 50, 74, 117], which are a key instrument of a field of EC called *generative and developmental systems*, have become a growing area of interest in evolutionary computation.

One such indirect encoding designed explicitly for neural networks is in Hypercube-based NEAT (HyperNEAT) [37, 120], which is itself an indirect extension of the directly-encoded NEAT approach [116, 118] reviewed in the previous section. The geometric principles behind HyperNEAT lay the foundation for the multi-spatial approach introduced next. This section briefly reviews HyperNEAT; a complete introduction can be found in Stanley et al. [120] and Gauci and Stanley [37].

Rather than expressing connection weights as independent parameters in the genome, HyperNEAT allows them to vary across the phenotype in a regular pattern through an indirect encoding called a *compositional pattern producing network* (CPPN; Stanley [113]), which is like an ANN, but with specially-chosen activation functions (such as sine and Gaussian) that promote the expression of important motifs such as symmetry and repetition.

CPPNs in HyperNEAT *encode* the connectivity patterns of ANNs as a *function of geometry*. That is, if an ANN's nodes are embedded in a geometry, i.e. assigned coordinates within a space, then

it is possible to represent its connectivity as a single evolved function of such coordinates. While other indirect encodings preceding HyperNEAT also encoded geometric structures (e.g. Hornby and Pollack [50]), CPPNs are unique in encoding geometry as a function of explcit coordinates. In effect the CPPN paints a pattern of weights across the geometry of a neural network. Because the CPPN encoding is itself a network, it can be evolved in HyperNEAT by the NEAT algorithm, which is designed to evolve networks of increasing complexity. To understand why this approach is promising, consider that a natural organism's brain is physically embedded within a three-dimensional geometric space and that such embedding heavily constrains and influences the brain's connectivity. Topographic maps (i.e. ordered projections of sensory or effector systems such as the retina or musculature) in natural brains preserve geometric relationships between high-dimensional sensor and effector fields [51, 128]. In other words, there is important information *implicit* in geometry that can only be exploited by an encoding informed by such geometry.

In particular, geometric *regularities* such as symmetry or repetition are pervasive throughout the connectivity of natural brains. To similarly achieve such regularities, CPPNs exploit activation functions that induce regularities in HyperNEAT networks. The general idea is that a CPPN takes as input the geometric coordinates of two nodes embedded in the *substrate*, i.e. an ANN situated in a particular geometry, and outputs the weight of the connection between those two nodes (figure 2.1). In this way, a Gaussian activation function by virtue of its symmetry can induce symmetric connectivity across the substrate and a sine function can induce networks with repeated elements. Importantly, this allows multiple related components in the substrate to be discovered or modified in a single mutation (e.g. a single mutation can teach an agent to avoid walls on the left as well as avoid walls on the right). Note that because the size of the CPPN is decoupled from the size of the substrate, HyperNEAT can compactly encode the connectivity of an arbitrarily large substrate with a single CPPN. This makes HyperNEAT ideal for evolving controllers for agents with large numbers of geometrically-related sensors and effectors (such as those in the experiments featured

in Chapter 8) or for complex tasks requiring thousands or even millions of connections.

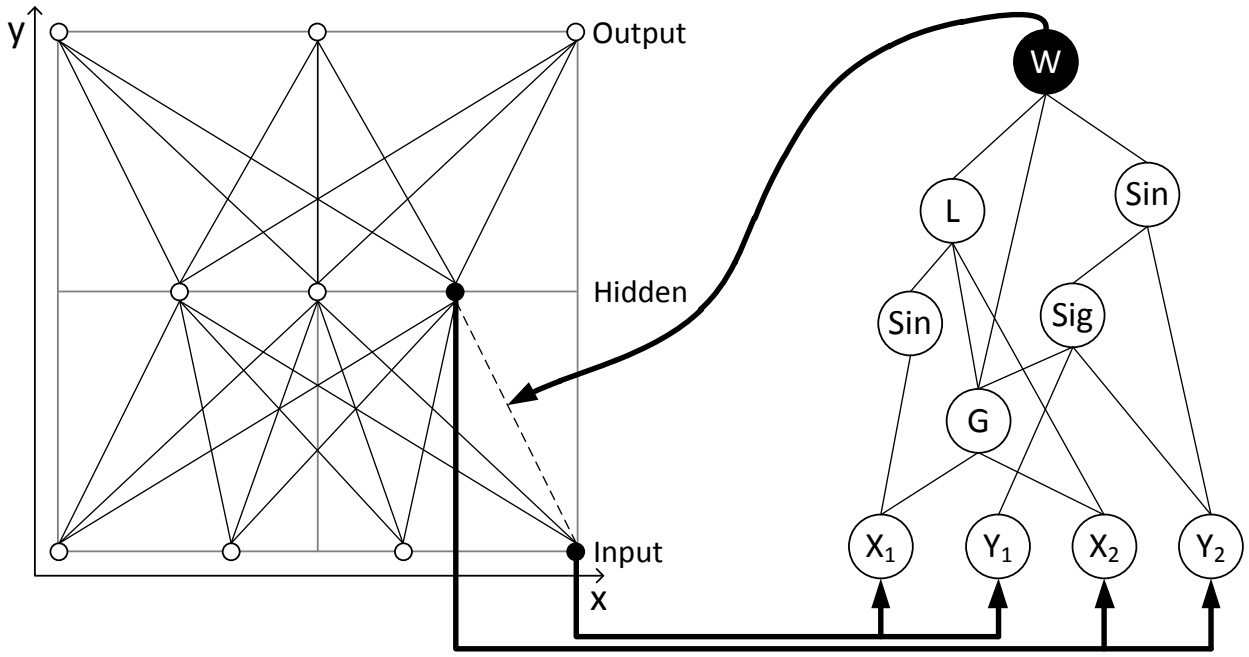


Figure 2.1: **HyperNEAT example.** An example substrate (left) for a simple ANN contains ten neurons that have been assigned (x, y) coordinates. The weight of every connection specified in the substrate is determined by the evolved CPPN (right): (1) The coordinates of the source (x_1, y_1) and target (x_2, y_2) neurons are input into the CPPN, (2) the CPPN is activated, and (3) the weight w of the connection being queried is set to the CPPN's output. Throughout this dissertation, CPPN activation functions can be *sigmoid (Sig), Gaussian (G), linear (L), or sine (Sin).*

Sensory Input in Neuroevolved Agents

Researchers have evolved neural-controlled agents for diverse domains ranging in complexity from single-agent maze navigation [35] to multiagent video game warfare [119]. Early work centered on behaviors that could be realized on available robot hardware, such as the Khepera. For example, Floreano and Mondada [35] evolve controllers for the Khepera robot that can navigate a simple maze without colliding with walls. These controllers feature only a single sensory modality: a set of infrared proximity sensors that surround the Khepera robot. In similar work, Nolfi [86]

evolves controllers for a Khepera robot equipped with a "gripper" module capable of grabbing and releasing objects. Successful integration of the gripper module requires the addition of an additional sensory modality: a single input neuron that is activated when an object is present inside the gripper claw.

Some neural controllers are evolved in simulation without any intention of physical implementation. For example, video game non-player characters (NPCs) often must process a relatively diverse set of sensory inputs. Stanley et al. [119] present such a video game called NERO in which human players attempt to evolve a team of ANN-controlled agents for a combat simulation against other teams of such agents. Agents in NERO have sensory inputs encompassing at least four distinct modalities: enemies, walls, hostile line-of-fire, and a boolean sensor that detects whether the agent's gun is currently aimed at an enemy target. Schrum and Miikkulainen [104] demonstrate video game NPC controllers evolved for a variety of multitask combat-related domains that have thirteen distinct sensory modalities. In general, controllers for more advanced tasks often require a greater variety of types of sensory input. As more sensors are added to the controller architecture, learning a successful control policy increasingly becomes a matter of disentangling sensor modalities that do not relate to one other. While researchers such as Clune et al. [19] and Verbancsics and Stanley [130] have studied the potential for the learning algorithm to discover modular divisions on its own, the user is generally left without the ability to provide some kind of a priori hint to the algorithm about the ideal groupings of neurons (such as when they belong to different modalities).

Neuroevolution methods based on an indirect encoding, such as HyperNEAT [120], offer a potential benefit to learning a large diversity of sensory input because geometric arrangements of neurons such as in HyperNEAT could in principle help to convey *belonging* to one modality or another. Furthermore, some modalities may share certain properties and thus benefit from information reuse. For example, in some cases modalities that consist of a single boolean value should be treated in a similar way that is different from the way that continuous rangefinder arrays should

be treated. However, current applications of HyperNEAT feature relatively few sensory modalities [18, 25, 26, 37, 99, 120]. For example, Stanley et al. [120] evolve agents with HyperNEAT for a food gathering task. In this application, agents only have a single array of eight rangefinders (corresponding to a single sensory modality). Similarly, the agents featured in Risi and Stanley [99] have only two sensory modalities: wall sensors and target sensors. Although multiagent domains often necessitate the addition of extra sensory information to facilitate communication and detecting the location or status of other friendly agents, multiagent HyperNEAT has so far only been used to evolve agents with one [25] or two [26] sensory modalities. The application of HyperNEAT that features the most diverse array of sensory information to date is Clune et al. [18]. In this application, HyperNEAT networks tasked with controlling the locomotion of a quadrupedal walker receive input from several modalities: the angles of the three joints in each of the legs, boolean values that are active when their corresponding leg is touching the ground, the rotation of the walker's body, and a sine input used to generate oscillations. With no known principle or precedent to follow, the arrangement of all these on a single substrate posed a difficult challenge to Clune et al. [18].

Perhaps the reason that HyperNEAT-evolved neural controllers often have few sensory modalities is that as more types of input are added, it becomes increasingly unclear how to organize their corresponding neurons on the substrate. The conventional approach to substrate design is to arrange the different modalities arbitrarily along one or two spatial dimensions, such as in Clune et al. [18]. The order and spacing of modalities along these dimensions can significantly impact the results. By convention, the best ordering is generally determined experimentally. However, this approach is computationally expensive and impractical.

One of the key contributions of this dissertation is to offer a principled solution called the *multi-spatial substrate (MSS)* that places each sensory modality in its own geometric space, similarly to (though much expanded from) how the input, hidden, and output layers are divided in past work

such as Gauci and Stanley [37] and Drchal et al. [33]. The basic idea of splitting neurons into different spaces appeared in Drchal et al. [33], but its significance as a solution to the problem of multimodality was not validated experimentally until the work included in this dissertation [89]. HyperNEAT together with the new MSS technique offers researchers the important ability to provide evolution with an a priori hint about the ideal logical grouping of neurons (e.g. according to which type of sensor they correspond to), thus enabling the confrontation of increasingly multimodal problems in the future. The MSS approach is explained in detail in Chapter 3 and validated against conventional approaches on a highly multimodal multiagent control task in Chapter 4.

Multi-solution Approaches Before QD

In the EC literature, early work in *multi-modal function optimization* (MMFO [70]) foreshadowed the later arrival of QD. The aim in MMFO is to discover multiple local optima within a search space, which naturally yields a diversity of solutions. However, MMFO has historically been concerned with relatively low dimensional mathematical functions [66] and as such, the exact techniques found in MMFO literature often cannot be directly translated into the high dimensional search spaces common in evolutionary robotics and neuroevolution (which may span thousands of parameters). Aside from differences in dimensionality, the main difference between MMFO and QD is that MMFO deals with *genetic* diversity (the only possible type of diversity in function optimization, where the "genotype" is a list of function inputs that maps directly to fitness), whereas QD reflects a later shift in interest towards *phenotypic* diversity (e.g. the observed behavior of a robot controlled by a neural network).

Unlike genetic diversity, phenotypic diversity cannot be sampled directly from the search space because the mapping from genotypes to phenotypes is often complicated and irregular. To illustrate, small changes in genotype can produce large changes in phenotype (e.g. slightly ad-

justing the weights in a neural network controlling a robot can cause the robot to behave in wildly different ways); this difficulty is especially amplified in the presence of indirect encodings [13, 49, 50, 113, 117]. Conversely, genomes that are different can nevertheless behave similarly (a phenomenon called genetic aliasing). While techniques to promote and maintain genetic diversity enjoy widespread use within neuroevolution as a means to overcome deception [116, 118], the unpredictable nature of the genotype to phenotype mapping precludes the possibility of achieving phenotypic diversity for free simply by maintaining genetic diversity; instead, more sophisticated approaches are required.

Another subject of research related to QD is multi-objective optimization (MOO) [30]. In MOO, the search algorithm aims to uncover the key trade-offs (called the Pareto front) among two or more objectives set by the user. Like QD, MOO returns a set of top candidates rather than a single winner, but MOO is still ultimately driven towards specific objectives (though more than one) and therefore is intrinsically convergent. In contrast, QD is a genuine *divergent* form of search driven explicitly to move *away* in the search space from where it has visited before. The unique effect is thus to reveal the full spectrum of possible behaviors latent in a search space, without having to specify each of them as targets a priori.

Early Divergent Search Algorithms

Interest in divergence in evolutionary algorithms was sparked initially by the surprising observation that some problems are solved more reliably by searching for novel behaviors than by searching for their objective [59, 61]. This approach, called *novelty search*, revealed just how pervasive and costly deception can be in otherwise unremarkable domains. It also showed that searching divergently instead of convergently can sometimes sidestep deception to uncover desirable parts of the search space. The benefits of divergent search were further confirmed through similar exper-

iments by Mouret and Doncieux [79] and Mouret and Doncieux [80], who use the synonymous term *behavioral diversity*.

These initial studies were followed by a wave of interest in divergent search algorithms as researchers explored their potential for discovery and open-endedness [32, 41–43, 45, 55, 57, 67, 68, 72, 76, 81, 100–102, 110, 131]. However, a clear missing ingredient from pure novelty or behavioral diversity techniques is a complementary notion of objective quality. The radical shift away from objectives towards divergence discards with it the ability to specify up front what is good and what is bad, and to have that notion influence the search. The loss of this convenient notion set the stage for its reemergence in QD algorithms.

Quality Diversity (QD) Algorothms

Early works on behavioral diversity [79, 80] and some more recent studies of combining novelty with objectives [44] investigate the idea of hybridizing novelty with fitness. The means of such combination is usually through a multiobjective framework [79, 80], though a weighted combination is also possible [44]. However, in either case the notion of fitness (i.e. quality) is *global*, which means in effect that the component of the search pushing towards quality focuses its effort exclusively on the part of the search space where performance is dominant over all other areas of the search space. In other words, the push towards quality in such a multiobjective hybridization reintroduces a strong convergent force into the search. While that may help in some cases if the aim is to discover a single near-global optimum, QD algorithms aim instead at finding more than just the global optimum.

In particular, the hope in QD is to uncover as many diverse behavioral niches as possible, but where each niche is represented by a candidate of the highest possible quality *for that niche*. That way, the

result is a kind of *illumination* of the best of all the diverse possibilities that exist [78]. The original QD algorithm of this type, called *novelty search with local competition* (NSLC), hybridizes novelty search with local fitness competitions only among individuals with similar behaviors, yielding a broad population of numerous simultaneous local competitions in diverse behavioral niches [62]. In its first demonstration, NSLC uncovered a collection of effective virtual creature morphologies and walking strategies in a single run, thereby demonstrating the unique benefits of QD [62].

The potential to uncover a diverse collection of high-quality alternatives in a single run inspired further investigations and algorithms. For example, Szerlip and Stanley [122] evolve diverse ambulating two-dimensional creatures made of sticks, also in a single run. Cully and Mouret [22] evolve a collection of walking behaviors for quadruped robots that functions as a repertoire of skills for a higher-level controller. In a hint at the wide applicability of QD even to fields outside of evolutionary robotics, Szerlip et al. [123] evolve diverse low-level feature-detectors for neural networks that are later aggregated into a combined classifier network. Revealing that QD encompasses more than a single algorithm, Mouret and Clune [78] and Cully et al. [24] introduce an alternate QD algorithm called *multi-dimensional archive of phenotypic elites* (MAP-Elites) that collects elite versions of each type of behavior within individual bins in a behavioral map. In yet another QD application, MAP-Elites collects diverse walking strategies for a robot that can be adapted in response to different kinds of damage [24]. Other applications of MAP-Elites include generating sets of images for fooling deep networks [84] and for exposing the space of concepts encoded inside a deep network [83].

The quickly-expanding set of applications of QD motivates the need for a systematic study of its best practices, which is one of the aims of this dissertation. For that purpose, a key concept in any QD algorithm is the *behavior characterization* (BC). The BC is usually a vector that describes the chronology of actions taken by an individual during its evaluation. This vector is then used to compute its novelty compared to other individuals (or its location in the behavior map in MAP-

Elites), thereby driving the diversity component of QD. An important property of the BC is its *alignment* with the notion of quality, which refers to the *degree to which finding novelty tends also to lead to higher fitness*. For example, in a maze, if the BC is based on the final position reached, then it is highly aligned because eventually an agent that continues to find new final positions will find the endpoint of the maze.

In contrast, interestingly, most published QD applications involve finding diversity with respect to an *unaligned* BC because usually the notion of diversity that we find interesting is not intrinsically aligned with quality [22, 62, 78, 122]. For example, seeking creatures of different morphologies or with different numbers of legs does not naturally lead to higher-quality walking, yet we are nevertheless interested in finding such creatures. In these early applications, QD has succeeded even despite a lack of such alignment, leaning heavily on the quality component of the algorithm to push otherwise unaligned notions of diversity towards good performance.

However, as the experiments in Chapter 6 will show, finding QD with an unaligned BC can often lead to a *slower* discovery of passable solutions, raising the question of whether the practice of searching with an unaligned BC breaks down entirely when the domain is sufficiently hard. This question is addressed in Chapter 7 where different approaches to QD are subjected to a succession of complex maze-navigation tasks. These studies not only reveal how QD holds up in more difficult domains, but also offer a number of new strategies for mitigating the effects of BC misalignment (which is usually the most desirable and intuitive way to set up QD). Such insight is critical to the future of QD as a nascent field, but also important for the general progress of machine learning outside of conventional convergent closed problems, where the potential might be open-ended and the aim to collect broad possibilities rather than to converge to a final answer.

Open-ended Evolution (OEE)

Achieving open-ended evolution has challenged the artificial life community since the field's inception. Open-endedness was initially viewed as a detectable property of a certain class of evolutionary systems; a system can exhibit a capacity for ongoing innovation on the order of biological evolution (which has historically been the inspiration for artificial open-ended systems) or something less productive. This perspective is reflected in the activity statistics test that aims to detect signatures of open-endedness [6, 15]. In recent years, however, a more pluralistic view of OEE has also emerged, admitting a wider variety of *kinds* of open-endedness [125].

Whatever perspective one takes, it seems that no artificial evolutionary system to date has displayed the capacity for open-endedness observed in biological and physical systems [29]. Digital evolution systems such as Tierra [97] and Avida [88], in which code-based lifeforms compete for computational resources on a minimal virtual computer, have proved fruitful for the purposes of studying evolutionary dynamics in a non-biological context. For example, Lenski et al. [65] demonstrate how digital organisms in Avida can learn to perform complex logic functions only when some of the simpler functions that comprise them provide some evolutionary advantage. However, systems such as Tierra and Avida have not yet produced definitive explosions of complexity that might be expected of OEE. Embodied systems such as the cell-based world Chimera (which in some configurations exhibits phase transitions to primitive multicellularity) [109] and the block-creature worlds of Division Blocks [111], Evosphere [73], and the seminal work of Karl Sims [108] offer great expressive potential through the individuals and the environment they inhabit. Yet there too a definitive open-ended result remains elusive.

It is also difficult to assess most systems' potential for major transitions simply because most published experiments run for a relatively short amount of time (at least compared to evolution on Earth). Chapter 8 introduces a novel experimental platform designed to support definitive transi-

tions that can be identified and tracked. Chapter 10 then presents a long-running experiment in this new platform spanning hundreds of millions of evaluations in an effort to detect true open-ended discovery.

Complexity and Open-Endedness

An ongoing challenge for the field of OEE is to identify the right kind of evidence to support that innovation and complexity continue to increase in systems hypothesized to be open-ended [4, 5, 8, 124]. While the most straightforward approach in principle would simply be to measure complexity to show that it goes up, in practice theoretical measures of complexity (e.g. Blum [11] and Gell-Mann and Lloyd [38]) are difficult or impossible to compute in OEE domains, and may also inadvertently shift focus to a formal concept with subtle differences from the intended dynamic of open-ended search. For example, OEE may be seeking not simply a higher Kolmogorov complexity, which could be exhibited in intricate but meangingless wallpaper, but rather evidence of designs directed at a meaningful purpose, such as a structure able to stand on its own. Thus, to confront this challenge in Chapter 10, where the open-endedness of QD is investigated, a number of different views of complexity in evolution, from quantitative to qualitative, are aggregated to support an interpretation of long-term dynamics.

CHAPTER 3: NEW APPROACH 1: MULTI-SPATIAL SUBSTRATES (MSS)

The method described in this chapter represents an original contribution of this dissertation, first published in Pugh and Stanley [89]. An experimental validation of the method can be found in Chapter 4.

The primary challenge in designing a HyperNEAT substrate is deciding how to spatially arrange several sets of geometrically unrelated neurons. This chapter introduces a new approach to substrate design called the *multi-spatial substrate (MSS)* that is explained here by example. The example first demonstrates how a simple network architecture with only three neuron groups (one input layer, one hidden layer, and one output layer) might be implemented following the conventional approach to substrate design and then proceeds by showing how the same architecture would instead be implemented by following the MSS approach.

Conventional "Crowded" Substrates

The most common example that appears in nearly every substrate design problem is deciding how the input, hidden, and output sets of neurons should appear in the substrate space relative to each other. Imagine a simple agent that has a set of five rangefinder sensors equally spaced along the front 180 degrees of the agent and three effectors to perform the actions of turning left, moving forward, and turning right. Each sensor corresponds to a single input neuron and each effector corresponds to a single output neuron. This neural controller also has a set of five hidden neurons to which the input neurons are fully connected and that are fully connected to the output neurons. A common design for such a substrate is shown in figure 3.1a. Because there exists a

clear geometric relationship between the inputs, a good practice is to arrange them in a straight line along one dimension of the substrate (such as parallel to the x-axis) in the same order that the corresponding sensors appear on the agent (figure 3.1a). A similar geometric relationship exists among the outputs, which are therefore often arranged in a line as well. Arranging the inputs and outputs in this way allows HyperNEAT to discover the concept of left-right symmetry along the inputs and outputs in a single step. The hidden nodes are often arranged in a line in the same way as the inputs and outputs to convey an identical left-right symmetry along the hidden nodes.

Overall, the user designing the substrate is left to situate three distinct sets of neurons, each arranged in a line parallel to the x-axis. However, it is unclear how to place these sets in the final substrate *relative to each other* because there is no clear geometric relationship between them. A common practice is to arrange them arbitrarily along another, orthogonal axis (such as the y-axis in figure 3.1a). This arbitrary placement of geometrically unrelated neuron groups is referred to here as the *crowded substrate* because it involves squeezing several unrelated types of neurons into the same space. To understand why it is a single shared space, notice that the CPPN that encodes the crowded substrate has a single weight output W and similarly a single bias output B (figure 3.1b). In effect, that means mathematically that all weights and biases are projected onto the same plane.

Multi-Spatial Substrate Design

The main contribution here is to introduce an approach to designing HyperNEAT substrates that avoids the arbitrary placement of geometrically unrelated neurons (which can be a challenging creative burden on the user for more complicated architectures). This placement problem is resolved in MSS by following two simple rules. First, (1) neurons that are logically related (e.g. they represent the same modality) are grouped together. That is, each input and output modality forms a separate group, and hidden neurons can be grouped according to their connectivity to other

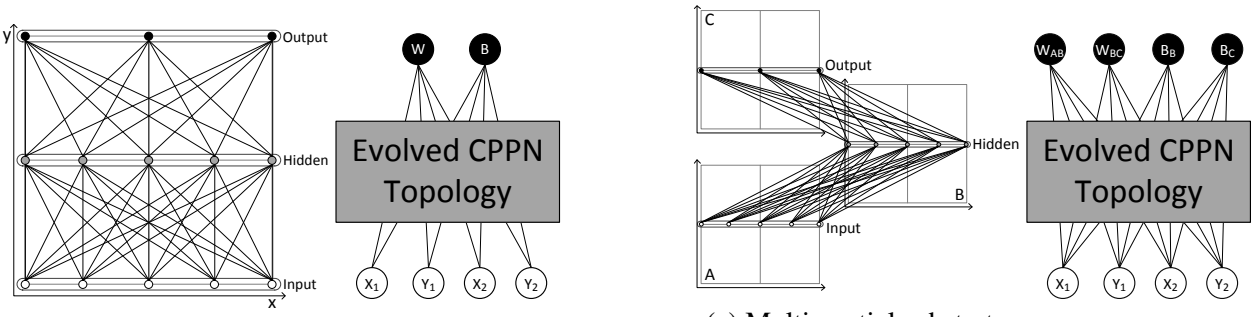
neuron groups in the network. Then, (2) each group is placed in a *separate* space according to the principle: *two groups of neurons with no obvious geometric relationship between them should not be placed in the same space.*

Figure 3.1 depicts a typical substrate configuration converted into a multi-spatial substrate. In this example, each "space" can be conceived as a separate plane and there are three logical groupings of neurons. Thus the resulting substrate contains three separate planes (figure 3.1c). Substrates organized in this way are called *multi-spatial substrates* (MSS) because geometrically unrelated sets of neurons are placed in different spaces. In practice, adding spaces to the substrate accordingly requires extra CPPN outputs to differentiate among the spaces being connected. While the crowded (single-spatial) substrate requires two CPPN outputs (one that is read when querying connection weights and one that is read when querying the implicit bias[1] on every non-input neuron), the multi-spatial substrate requires $N + M$ CPPN outputs where N is the number of unique space-space pairings (i.e. where there exist neurons from one space that are connected to neurons on the other space) and M is the number of planes that contain non-input neurons. It should be noted that a space with internal connections counts as a separate pairing (i.e. a space paired with itself) for the purposes of calculating N. Figure 3.1d shows the MSS encoding that is an alternative to the crowded encoding in figure 3.1b.

In addition to being more challenging to design, crowded substrates impose an evolutionary burden on HyperNEAT. If geometrically unrelated neuron groups are arranged along the y-axis, the HyperNEAT CPPN must learn a function of y that expresses the differences among these logical groups of neurons. This function becomes more complicated and thus more difficult to discover during evolution as more groups of neurons are added to the substrate (e.g. when the number

[1]The implicit bias is a convention in HyperNEAT that simulates a single bias neuron that is fully connected to all other neurons except the inputs [114]. The CPPN is queried for the bias of a neuron by setting the X_1 and Y_1 inputs to the neuron's coordinates and setting X_2 and Y_2 to zero. Implementing the bias in this way effectively places the bias at the center of a separate plane.

of input or output modalities increases, or when more layers of hidden neurons are added to the network). By providing a separate CPPN output for each logically separate set of connections, multi-spatial substrates do not need to learn a potentially complex function to differentiate among these sets. Rather, the separation of each set into its own CPPN output conveys this information a priori.



(a) Crowded substrate (CS) (b) CPPN (CS) (c) Multi-spatial substrate (MSS) (d) CPPN (MSS)

Figure 3.1: **Example: Converting a crowded substrate to a multi-spatial substrate.** Two substrate designs are depicted for a simple neural controller with one set of inputs, one hidden layer, and one set of outputs. The first substrate (a) follows the crowded substrate approach, which is typical for designing substrates: Neuron groups that are not geometrically related are arranged arbitrarily along an "extra" spatial dimension (in this case, the y-axis) that is otherwise not used to convey geometric information. The second substrate (c) follows the multi-spatial substrate approach, wherein geometrically independent neuron groups are placed in separate spaces (i.e. planes). Also shown are the corresponding CPPNs for both substrates (b, d). Note that the addition of extra planes in the multi-spatial substrate necessitates extra CPPN outputs (d). W_{AB} is queried when determining the weight of connections from plane A to plane B and W_{BC} is queried in the same way for connections from plane B to plane C. B_B and B_C are queried when determining the implicit bias for neurons on plane B and plane C, respectively (there is no B_A because input neurons do not have an implicit bias). In this example, the conversion to a multi-spatial substrate removes the need for the y-axis entirely; thus, the Y_1 and Y_2 inputs in (d) could be removed (not shown). However, the y-axis may also be repurposed (for example, to expand the hidden layer to a grid of neurons).

In the context of this dissertation, the multi-spatial substrate idea is important for the Voxelbuild domain introduced in Chapter 8. In Voxelbuild, agent networks have large input and output arrays

similar to a 3-dimensional visual field, as well as several distinct input and output modalities. These networks would be impractical to evolve with NEAT because of the large number of nodes and connections to contend with as well as the loss of information about the placement of sensors relative to each other in space. Instead, Voxelbuild networks are evolved with HyperNEAT where the MSS approach allows all modalities to easily coexist in the same network.

CHAPTER 4: EXPERIMENT: MULTI-SPATIAL SUBSTRATES VALIDATION

The experiments in the following chapter were originally published in Pugh and Stanley [89].

Among the long-term goals of the field of neuroevolution is to evolve artificial neural networks (ANNs) with capabilities similar to the human brain. Although current methods of evolving ANNs cannot approach the complexity of the brain, recent work focusing on indirect encoding has enabled the evolution of ANNs with thousands to millions of connections from genotypes with far fewer components (on the order of 20 to 30) [37, 120]. However, one aspect of the human brain that has not yet attracted significant attention in the neuroevolution literature is the diversity of senses. The human brain receives input from millions of receptors representing a wide array of sensory modalities (e.g. sight, touch, taste, smell, hearing, pain, temperature, balance, proprioception, etc.) [53]. This sensory richness provides the human brain with complementary representations of its environment that are critical to effective function.

As researchers continue to seek more advanced behaviors from evolved neural controllers, it will become increasingly important also to equip such controllers with additional modalities that can potentially each support its own high resolution. However, successfully integrating different kinds of sensory information in an ANN is challenging because there is no explicit way to denote to the network which sensors are grouped with which. Even outside of the evolutionary community, there has been interest in recent years in multimodal integration. For example, Loyola and Coldewey-Egbers [69] developed a stacked neural network architecture that merges sensor data for long-term climate research. Interestingly, the field of *generative and developmental systems* (GDS) [9, 14, 50, 74, 113, 117] may be able to help with multimodal integration in evolved ANNs by grouping different types of sensors in different parts of the generated phenotype. Furthermore, the processing

of different sensory modalities may share some principles with each other, enabling information reuse. For example, in a foraging domain, an agent should act in a similar manner when seeing a food source as when smelling a food source. Inspired by this possibility, an elaboration of the conventional HyperNEAT GDS approach for evolving large-scale ANNs [37, 120] called the *multi-spatial substrate* (MSS) enables a principled integration of several sensory modalities.

While it has long been unclear how or where to place different types of sensors or outputs on the same HyperNEAT substrate, the key new idea is to place different sensory modalities on sheets encoded in HyperNEAT as entirely disconnected geometric spaces. That way there is no risk of confusing or conflating one modality with another. Thus a solution is offered to the longstanding dilemma about how to configure complex HyperNEAT substrates that at the same time yields a principled approach to the problem of representing and learning from multiple modalities in general. This approach is demonstrated here in a multiagent coordination domain that requires agents to perceive three modalities at once: walls, targets, and communications from other agents. The MSS approach significantly outperforms more conventional configurations in which all modalities are placed on the same plane and is also simpler to set up. This principled ability to train in multimodal domains thereby opens up many such tasks to research that were perhaps prohibitive in the past.

For a full description of the MSS approach, see Chapter 3.

While CPPNs with multiple outputs have appeared before with little commentary, such as in Drchal et al. [33] and Gauci and Stanley [37], the novel insight here is that the separation provided by the multiple outputs is actually a *solution* to the problem of configuring and training networks of many modalities. In particular, the hypothesis of this experiment is that the additional information provided to the HyperNEAT CPPN by the MSS approach will improve evolutionary performance particularly in cases where there are many logically separate neuron groups (i.e. sensor and effector

modalities).

Multi-Spatial Substrates for Multimodal Controllers

Substrates for multimodal controllers tend to have more logically separate neuron groups than substrates for unimodal controllers. One class of domains that naturally requires multimodality is multiagent learning. Consider a team of coordinating robots with three input modalities: (1) target sensing, (2) communication sensing, and (3) wall sensing, as well as two output modalities: (1) steering and (2) a "voice box" for sending communication signals to other robots. Target sensors consist of five pie slice sensors across the front 180 degrees of the robot. Wall sensors are similarly arranged except rangefinders are used instead of pie slices. Communication sensors consist of ten pie slice sensors that surround the robot. Steering is determined by three outputs: turn left, move straight, and turn right. The "voice box" output activates other robots' communication sensors that point in the direction of the vocalizing robot. Additionally, a controller for such a robot may require a more complex system of hidden layers to successfully process its variety of sensory modalities. Due to the additional modalities and hidden layers, the resulting network architecture has significantly more distinct neuron groups than the simple unimodal example controller discussed in Chapter 3. Importantly, these neuron groups do not necessarily have obvious or appropriate geometric relationships to each other (e.g. Where should pie slice sensors be located on the substrate relative to the steering outputs? Where should hidden neurons be located relative to other neuron groups?).

Figure 4.1 depicts a possible logical architecture for the connectivity of such a controller. From a design perspective, actually realizing this logical configuration is difficult within a single-plane "crowded substrate" because it is unclear where groups should be placed relative to each other within the space and there are an unlimited number of possible placements. Several possible

crowded substrate designs for this architecture are shown in figure 4.2. In a crowded substrate where the entire network exists within a single space, the user is forced to establish arbitrary spatial relationships between neuron groups. In contrast, the multi-spatial substrate is simple to configure because different modalities are simply placed on different planes (figure 4.3).

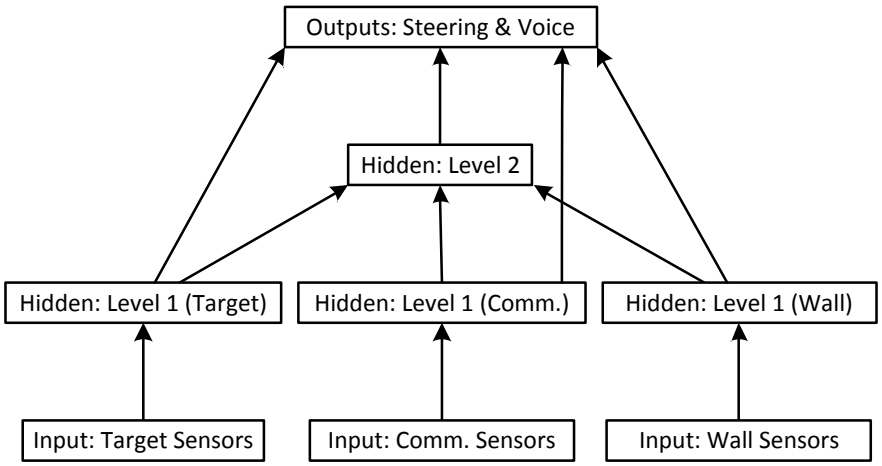
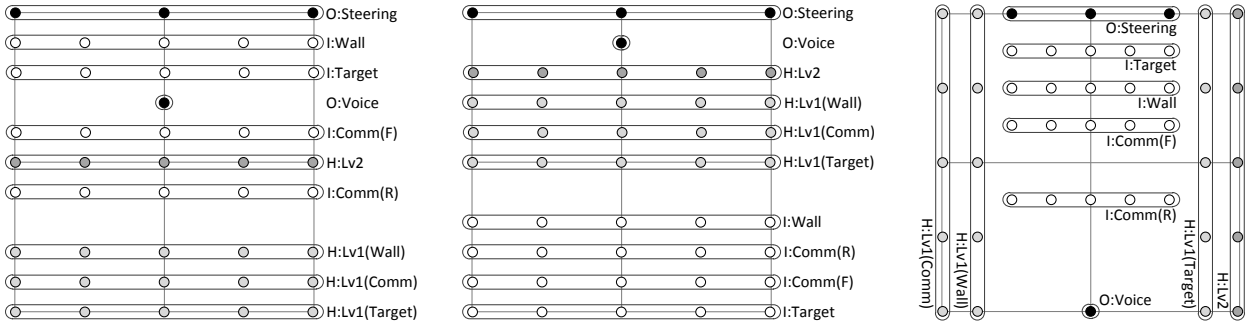


Figure 4.1: **Substrate logical connectivity.** The logical connectivity shared by all of the substrates featured in the following experiments is shown. Neuron groups connected by an arrow are fully connected (i.e. all neurons in the first group are queried for connections to all neurons in the second group). Each input modality is connected in turn to a dedicated "level 1" hidden layer. All level 1 hidden layers are connected to a single "level 2" hidden layer. Finally, all hidden layers are connected to the two output layers. While steering outputs and voice outputs are logically separated (and thus exist in their own separate spaces), they are merged in this diagram to reduce clutter.

The next section presents an experiment comparing the performance of a multi-spatial substrate to that of substrates designed with the conventional crowded substrate approach.

Experiment

The example multiagent neural architecture described in Section 4 is well-suited to illustrating the evolutionary advantage of the MSS approach because it relies strongly on multimodal input.



(a) CS1 (b) CS2 (c) CS3

Figure 4.2: **Alternate crowded substrates.** In each crowded substrate, the input, hidden, and output neurons are all placed together on a single plane, organized geometrically into groups of related neurons. Input neurons (denoted by *I*) are colored white, hidden neurons (denoted by *H*) are colored light gray (level 1) and dark gray (level 2), and output neurons (denoted by the letter *O*) are colored black. The communication (*Comm*) inputs are divided into two groups corresponding to front sensors (*F*) and rear sensors (*R*). Individual connections are omitted for clarity (logical connectivity is shown in figure 4.1).

For this purpose, teams of five agents matching the description in Section 4 are evolved with HyperNEAT[1] for a maze exploration and group coordination task.

The agents are placed in a maze containing one target point that is located somewhere outside of the initial vision range of the agents. The goal of the task is for the agents to navigate their way to the target point and then remain there. To avoid each agent needing to explore the entire map by itself, the best strategy requires one agent to call the others over after discovering the target's location. Fitness is accumulated as follows: Each agent earns one point per tick that it is at the target point. The duration of the simulation is $1{,}000$ ticks, which corresponds to a maximum fitness of $5{,}000$ for the five agents. However, due to the time required for the initial discovery of the target point and for all agents to move to it[2], the maximum achievable fitness is lower.

[1]Teams are homogeneous; all agents on a team have identical brains.

[2]Agents move at a maximum rate of 5 units per tick and have a maximum turn rate of 36 degrees per tick. The width of each environment's bounding walls is 900 units. Thus it takes 180 ticks for an agent to move across the world,

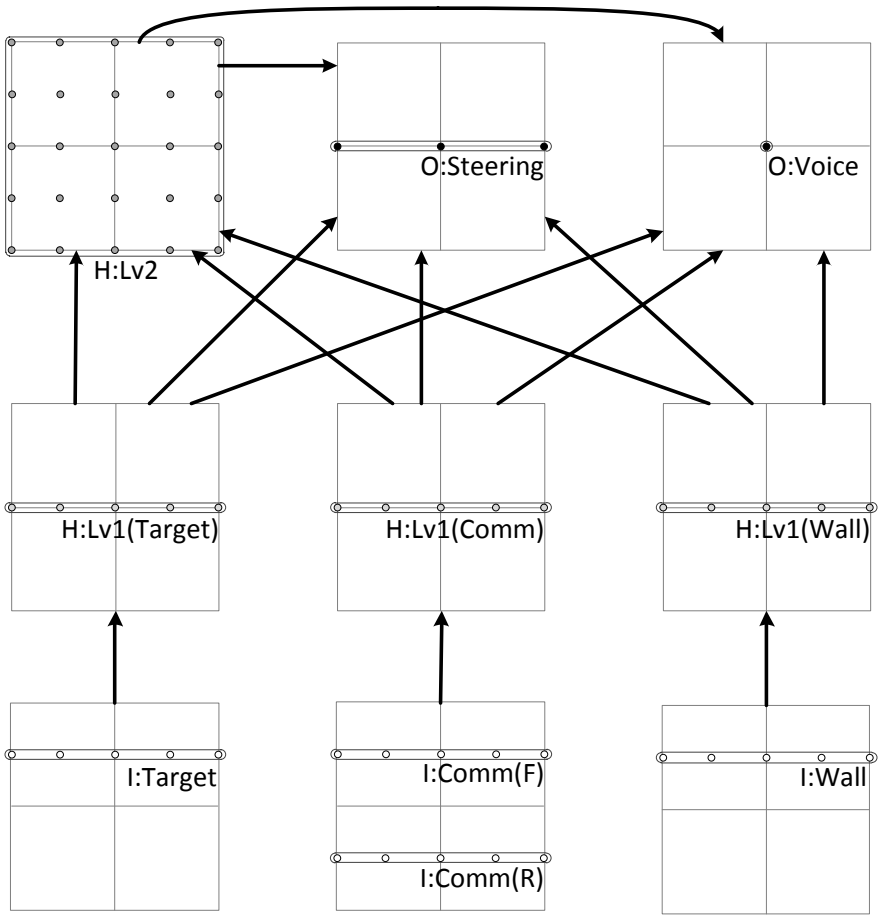


Figure 4.3: **MSS1 and MSS2 (not shown).** Each input and output modality and each hidden layer is placed on its own plane. The level 2 hidden layer (upper left) is expanded to a grid of neurons to allow the expression of more complex functionality. Front-facing sensors are located at $y = 0.5$ and rear-facing sensors are located at $y = -0.5$ in their respective input planes. Arrows indicate the existence of neural connections between two planes. Planes shown as connected are fully connected (all neurons on the first plane are queried for connections to all neurons on the second plane). The CPPN for this substrate has a total of 20 outputs (connections exist between 14 unique plane-plane pairs and there are 6 planes that have neurons with an implicit bias). MSS2 (not shown) is identical to MSS1 except its level 2 hidden layer (H:Lv2) only has a single line of five neurons located at $y = 0$.

Nearly perfect solutions wherein agents explore efficiently and move directly to the target point once it is found achieve a fitness of about $4{,}200$. Agent wall sensors and target sensors each have

while extra time is required to navigate around obstacles.

a maximum range of 150 units and are occluded by walls. However, agent communication sensors have unlimited range and are not occluded. The primary purpose of communication therefore is to share the location of the target point with other agents that are too far away (or else occluded by walls) to detect it on their own.

Teams are trained on a total of seven environments, shown in figure 4.4. The range of environments are designed to encompass a diversity of situations, including initial isolation from team members. During evolution, fitness is averaged over all seven.

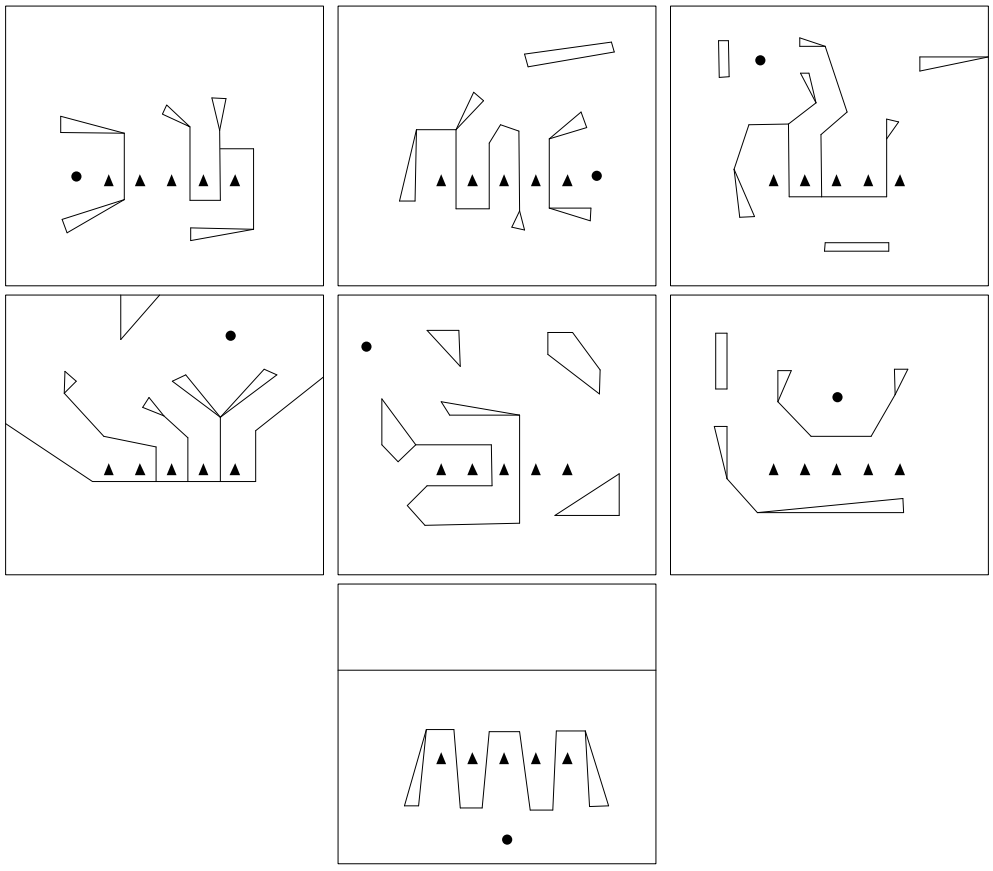


Figure 4.4: **Training environments.** Team performance is averaged over seven environments, depicted here. Agent starting positions are represented by triangles and target points are represented by circles. The ends of walls are widened to ensure agents can see them from all angles.

The experiment consists of a comparison between the evolutionary performance of HyperNEAT with each of the three crowded substrates as well as the two multi-spatial substrates (figures 4.2 and 4.3). The crowded substrates test several possible strategies for arranging neuron groups that have no obvious geometric relationship, while the multi-spatial substrates simply separate geometrically unrelated neuron groups by placing them in different coordinate spaces.

In the first crowded substrate, **CS1**, neuron groups are arranged such that front sensors appear on the positive y-axis and rear sensors appear on the negative y-axis (figure 4.2a). Additionally, the outputs are spaced reasonably far apart to prevent a correlation between the activation of the forward steering and voice box outputs from emerging during the early stages of evolution. **CS2** is designed such that all connections flow from neuron groups located at lesser values of y to neuron groups located at greater values of y (figure 4.2b). In **CS3**, horizontal neuron groups are compressed to make room for the hidden layers to be positioned vertically along the left and right edges of the substrate (figure 4.2c).

The first multi-spatial substrate, **MSS1**, is shown in figure 4.3. Here, the level 2 hidden layer (shown at upper-left) is expanded from a horizontal line of neurons into a grid of neurons (made possible because of the MSS approach), which allows for the expression of more complex functionality. **MSS2** (not pictured) is identical to MSS1 except that its level 2 hidden layer is not expanded, instead consisting of a single horizontal line of five neurons as in the crowded substrates. This variation helps to validate that any performance advantage for the MSS is not merely the result of expanding the level 2 hidden layer.

Experimental Parameters

Evolution proceeds for 500 generations, after which point fitness improvements stagnate even for runs that do not find a good solution. HyperNEAT is implemented in a modified version of the

public domain SharpNEAT package [46]. Because CPPN evolution in HyperNEAT differs from NEAT only by its set of allowable activation functions, it uses all of the same parameters [116]. The size of the population is 500 with 20% elitism. Two-parent offspring (50% chance) do not undergo any mutation aside from NEAT-style crossover. Single-parent offspring (50% chance) have a 96% chance of connection weight mutation, a 3% chance of adding a connection, and a 1% chance of adding a hidden neuron. The coefficients for determining species similarity are 1.0 for nodes and connections and 0.1 for connection weights. The available CPPN activation functions are sigmoid, Gaussian, linear, and sine, all with equal probability of being added to the CPPN. Parameter settings are based on standard SharpNEAT defaults and prior reported settings for NEAT [116, 118]. Preliminary experiments found these settings to be robust to moderate variation without affecting performance.

Results

The evolutionary performance of each substrate design, determined by the best fitness achieved at each generation, is averaged across 30 runs and presented in figure 4.5. The differences between MSS1 and MSS2 are not statistically significant.[3] Both MSS1 and MSS2 significantly outperform all three crowded substrates ($p < 0.001$) as determined by each of their best fitness achieved at generation 500. CS2 outperforms the other two crowded substrate designs ($p < 0.05$).

To provide further perspective on the disparity between CS and MSS, a "good solution" is defined as a fitness greater than $3{,}333$, which corresponds to all agents spending two-thirds of the total available time resting at the target point. This definition permits agents to cross the world approximately twice before finding the target point and excludes nearly all cases in which one or more agents never finds the target. The proportion of runs that find a good solution for each approach is

[3] Statistical significance is determined by an unpaired two-tailed Student's t-test in all reported cases.

presented in figure 4.6. Both MSS designs find a good solution almost every time, whereas even the best crowded substrate designs find a good solution only half of the time. The worst crowded substrate, CS3, finds a good solution in only two out of 30 runs.

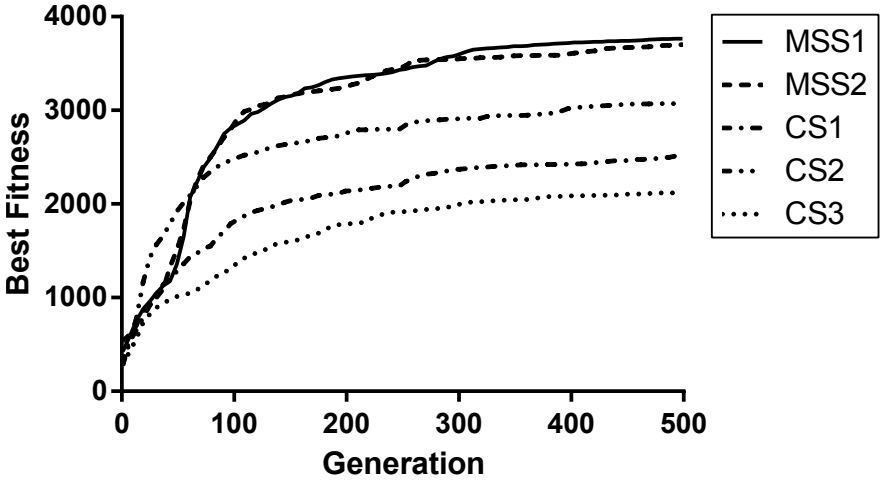


Figure 4.5: **Evolutionary performance by generation.** The best fitness achieved at each generation by each of the five substrate designs is shown, each averaged over 30 runs. The MSS variants reach significantly higher fitness by generation 500 ($p < 0.001$).

The agents on the best solution teams for MSS1 and MSS2 explore the map individually and only activate their voice box output after finding the target point. After the location is broadcast, the other agents move towards the signal, navigating around walls, until the entire team has reached the target. Thus in these teams agents effectively share information about the location of the target point with team members to expedite the exploratory process. Many of the best performing CS solution teams behave in the same way. However, some of the CS2 teams solve the problem without communication because the agents often choose to keep their voice box output activated at all times. These teams achieve slightly lower scores than communicating teams, but nonetheless exceed the success threshold.

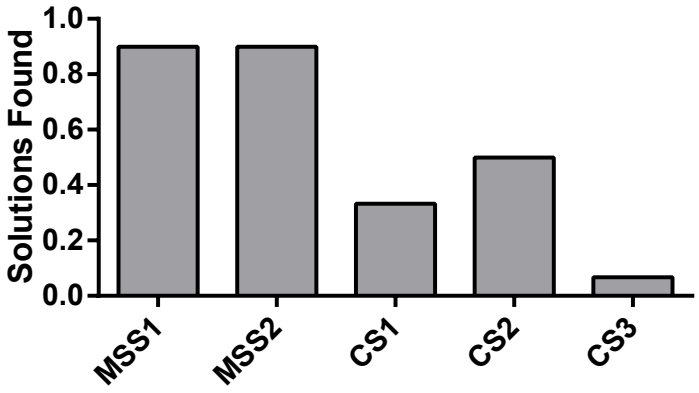


Figure 4.6: **Proportion of runs that find a good solution.** Proportions are taken out of 30 runs. A "good solution" is defined as exceeding a fitness threshold of 3,333. MSS finds such solutions 80% more often than the best crowded substrate.

Discussion

The crowded substrate design strategy faces the problem that the CPPN tends to assign similar weights (especially early in evolution) to connections that are near each other, such as those connecting to the voice box and forward movement neurons on the CS2 substrate (figure 4.2b). Such a correlation can become entrenched during the course of evolution if the candidate solution learns to rely on it, which explains why several of the best CS2 teams exhibit a correlation between forward movement and communication at generation 500 even though a higher score is possible without such correlation.

As more neurons are added to a crowded substrate to solve more complex problems, it is inevitable that unrelated neurons will have to be placed close enough together to cause unintended correlations in early generations. In contrast, neurons in a multi-spatial substrate easily avoid unintended correlations because each modality exists in an entirely separate space. In a MSS, logical separation is not conveyed by spatial distance on the substrate but by separate CPPN outputs, which is a cleaner distinction that is easily followed by HyperNEAT.

Figure 4.5 shows that the way crowded substrates are arranged can significantly impact their performance. Indeed, some CS substrate designs even prevent solving the problem altogether (e.g. CS3, figure 4.6). More complicated domains that necessitate additional modalities and hidden layer structure present an even greater design challenge to the user because the degree of neuron crowding and inappropriately imposed geometric relationships is greatly increased. In effect, as the number of modalities and associated modality-specific hidden layers increases, the burden on the user to design an effective crowded substrate increases until eventually the problem becomes intractable. While it is conceivable that the dimensionality of the crowded substrate could be expanded to reduce crowding (e.g. more CPPN inputs to represent a third or fourth dimension), this would only perpetuate the problem of deciding where to place unrelated modalities within the substrate in relation to each other.

On the other hand, multi-spatial substrates eliminate the need to make such difficult design decisions even while potentially improving performance. This result opens the door for future research evolving ANN controllers capable of processing a rich diversity of sensory information that was not previously possible which can impact domains ranging from biped walking to multiagent control. Furthermore, the advantages of the MSS technique are not restricted to multimodal controllers; there may be advantages to MSS whenever there are several a priori logical groupings of neurons within a substrate.

In the future it will be important to investigate the scalability of the MSS approach as well as how it may complement more organic approaches to modularity such as the Link Expression Output [130] or optimizing for connection costs [19].

Conclusions

This chapter highlights a principled approach to substrate design for the HyperNEAT algorithm called the multi-spatial substrate that provides a priori information to HyperNEAT about the distinctions among neuron groups, both improving evolutionary performance and reducing the creative burden on the user. The approach is validated through a comparison with conventional methods of substrate design on a multiagent domain wherein agents have three input and two output modalities; the multi-spatial substrate finds a solution 80% more often than even the best conventional substrate design in the tested domain. This new approach brings within the reach of neuroevolution a new tier of domains by providing a solution to the problem of multimodality that was not previously available. This approach is important later in Chapter 8 for evolving complex multimodal controllers in the Voxelbuild creative task.

CHAPTER 5: NEW APPROACH 2: MULTI-BC QUALITY DIVERSITY ALGORITHMS

While NEAT (and consequently HyperNEAT) generally constitute fully functional evolutionary algorithms on their own; in their usual formulations, parent selection from one generation to the next is proportional to the fitness (i.e. quality) of each individual. However, out of such selection mechanisms emerges a convergent search whereas QD is instead interested in *divergence*. In this work, NEAT (and HyperNEAT) simply serve as an encoding and mutation mechanism for evolution while more sophisticated selection and population maintenance is performed in the service of divergent QD. This chapter describes the various selection mechanisms relevant to QD. In addition, the chapter concludes with a discussion of how novelty archives are managed throughout this dissertation and its implications versus alternative methods, followed by a similar discussion of how QD is collected.

Selection Algorithms

Because the pursuit of quality diversity has only recently become a subject of persistent research interest, there are only two main QD algorithms initially represented in the literature: NSLC [62] and MAP-Elites [24, 78]. To provide a broader perspective on the untapped algorithmic potential in this developing field, the studies in Chapter 6 and Chapter chapter:unalignedqd additionally feature several novel variants of these core algorithms including some proposed improvements to MAP-Elites, which is mechanically very simple and thus particularly amenable to modification. Importantly, several other new variants called *multi-BC QD algorithms* specifically serve to address the problem of finding QD when the desired notion of diversity is unaligned with quality (and thus potentially incapable of finding the best solutions). The importance of *alignment* between quality

and diversity is empirically tested and discussed in Chapter 6.

With the exception of Fitness (Section 5), which is implemented generationally according to standard practice (the entire population is replaced on each tick), all other algorithms are implemented as steady state (only a small portion of the population is replaced at a time). In particular, batches of 32 genomes are evaluated at a time to facilitate a modest amount of parallelism without substantially disrupting the composition of the population between batches[1].

Pseudocode is provided for all algorithms introduced for the first time in this dissertation.

Controls

While not themselves QD algorithms, the following two controls are included in some experiments to establish the importance of specialized approaches that simultaneously balance drives towards behavioral diversity and locally-increasing quality. The controls thus respectively exemplify searching with quality pressure alone and with diversity pressure alone.

[Fitness] Neuroevolution of Augmenting Topologies

The first control, which includes only a quality component, represents a traditional objective-oriented optimization approach and is implemented as standard generational NEAT (NeuroEvolution of Augmenting Topologies) [116] with a population size of 500. Agents are rewarded according to the Euclidean distance between their final position and the goal point (this heuristic also underlies the quality component of all of the QD algorithms to follow). NEAT includes a

[1]Preliminary experiments revealed that running novelty-based algorithms generationally as opposed to steady-state interferes with the stable calculation of novelty scores and thus can lead to undesirable oscillations where the population is flooded with one type of individual and then subsequently with another, rather than maintaining a healthy spread across the behavior space.

sophisticated speciation and fitness-sharing mechanism for maintaining genetic diversity across the population. However, as the canonical HardMaze experiments [59] reveal, genetic diversity is often not enough to overcome the problem of deception on difficult maze-navigation tasks, which instead favor rewarding *behavioral* diversity (a critical component of QD algorithms).

[NS] Novelty Search

The *novelty search* (NS) [59, 61] algorithm represents the other extreme: searching only for behavioral diversity with no fixed objective at all (i.e. NS has no "quality component"). Novelty search works by rewarding *novelty* instead of fitness, where novelty measures how different an individual's behavior is from those that have been seen before. More formally, novelty is calculated by summing the distances to the k-nearest behaviors (here, $k = 20$) from a set composed of the current population and an *archive* of past behaviors. The distance between two behaviors is simply the Euclidean distance between those behaviors when represented as a vector of numbers (which is the origin of the term *behavior characterization* or BC). While there exist several different strategies for managing the archive [44], preliminary experiments indicated that a powerful strategy is to add all individuals to an archive with a maximum size that is enforced by deleting those with the lowest novelty (the novelty of all archive members is recomputed before each deletion). In this study, NS has a population size of 500 and a maximum archive size of 2,500.

Existing Quality Diversity Algorithms

Each of the following algorithms feature both of the essential components of QD: pressure to discover more behavioral niches and a tendency towards increasing performance in niches that have already been discovered.

[NSLC] Novelty Search with Local Competition

Perhaps the first true QD algorithm, *novelty search with local competition* (NSLC) [62] combines the diversifying power of NS with a localized fitness pressure called *local competition* (LC), calculated as the proportion of an individual's k-nearest ($k = 20$) behavior neighbors that have a lower fitness score. LC allows a quality-based reward to be assigned within behavioral neighborhoods without asserting that some neighborhoods are better than others. In NSLC, novelty and LC are combined by Pareto ranking following the practice of the NSGA-II multi-objective optimization algorithm [30].

[ME] Multi-dimensional Archive of Phenotypic Elites

An alternative approach to QD is an algorithm called *Multi-dimensional Archive of Phenotypic Elites* (MAP-Elites or ME) [24, 78]. The key difference in MAP-Elites is that niches are explicitly defined rather than passively emergent from a system of local competition; the behavior space is divided into a number of discrete behavior bins (often called a "grid" and created by discretizing each dimension of the BC) where each bin remembers the single fittest individual ever mapped to that bin. The set of filled bins constitutes the active population and evolution proceeds by selecting a bin at random (with equal probability) to produce an offspring, which is then mapped to a bin corresponding to its behavior where it may be either saved or discarded depending on whether its fitness is higher or lower than the current elite occupant. Because selection is uniform, the hope is to acquire diversity passively by virtue of the observation that more bins will tend to fill over time and, once filled, they will not be forgotten.

While MAP-Elites is a simple algorithm effective on its own and amenable to improvements, the requirement to explicitly define a grid of niches limits the possible dimensionality of the BC. In

the process of discretizing the behavior space into bins, the total number of bins is exponential in the dimensionality of the BC. In the minimal case, each dimension of the BC is discretized into only two parts. Even with such a course-grained discretization, the number of bins quickly becomes intractable as dimensionality increases. For example, under this discretization scheme, a BC with 10 dimensions would have $2^{10} = 1024$ bins. However, at a dimensionality of 15, the number of bins climbs to $2^{15} = 32,768$ and at dimensionality 20, the grid contains a staggering $2^{20} = 1,048,576$ bins. Beyond a dimensionality of 20, hardware memory and storage become a concern. With such a large number of bins, there are bound to be a low number of bin collisions (i.e. most everything is saved because it is mapped to an empty bin) and search essentially devolves into random selection.

Simple Extensions to MAP-Elites

One of the strengths of MAP-Elites is its simplicity, which makes it easy to add potentially performance-enhancing modifications. Two possible modifications are presented below.

[MENOV] MAP-Elites + Novelty

Unlike NS and NSLC, the original formulation of MAP-Elites does not preferentially explore under-represented behaviors, potentially causing it to lag in its discovery of new types of behaviors. Conveniently, it is easy to augment MAP-Elites with a stronger focus on diversity by simply making selection proportional to novelty. In this variant, called *MAP-Elites + Novelty* (MENOV), whenever offspring are generated, they are also added to an archive of past behaviors (with a maximum size of 2,500, managed in the same way as in NS) that enables calculating a novelty score for all members in the MAP-Elites grid.

Pseudocode: [MENOV] MAP-Elites + Novelty

Generate and evaluate B random genomes
for all $g \in B$ **do**
    Add g to R
    Map g to its corresponding grid bin
    if $g.fitness > bin.fitness$ **then**
        Save g in the bin (discard current occupant)
    else
        Discard g
    end if
end for
Calculate novelty scores for all $g \in P$ (against $R \cup P$)
loop
    Select $bsize$ parents from P with chance proportional to novelty score
    Generate and evaluate B offspring as asexual mutations of selected parents
    for all $g \in B$ **do**
        Add g to R
        Map g to its corresponding grid bin
        if $g.fitness > bin.fitness$ **then**
            Save g in the bin (discard current occupant)
        else
            Discard g
        end if
    end for
    Calculate novelty scores for all $g \in P$ (against $R \cup P$)

25: Calculate novelty scores for all $g \in R$ (against $R \cup P$)

26: **while** $R.count > rmax$ **do**

27: Discard $g \in R$ with lowest novelty score

28: **end while**

29: **end loop**

[MEPGD] MAP-Elites + Passive Genetic Diversity

The strict elitism at each bin in the original MAP-Elites formulation may eventually cause evolution to stagnate if all stepping stones to higher fitness require first making strides through lower-fitness space. Furthermore, genetic diversity is intrinsically limited when only a single individual is saved in each bin. Addressing both of these potential pitfalls without introducing any additional overhead, a new variant called *MAP-Elites + Passive Genetic Diversity* (MEPGD) saves two individuals in each bin instead of one[2]. Individuals with a lower fitness than the current elite still have a 30% chance of being saved in the second slot, regardless of their fitness, thus allowing MEPGD to explore the potential of some lineages that would have otherwise been discarded. Importantly, these extra slots have the property that they also coincide with the set of MAP-Elites bins, which guarantees their behavioral diversity.

Pseudocode: [MEPGD] MAP-Elites + Passive Genetic Diversity

1: Generate and evaluate B random genomes

2: **for all** $g \in B$ **do**

3: Map g to its corresponding grid bin

[2]The number of extra slots per bin can conceivably be expanded to any number, where all slots except the first are governed by random replacement.

4: **if** $g.fitness > bin.fitness$ **then**
5: Save g in the primary bin (discard current occupant)
6: **else if** random: 30% chance **then**
7: Save g in the secondary bin (discard current occupant)
8: **else**
9: Discard g
10: **end if**
11: **end for**
12: **loop**
13: Select $bsize$ parents from P (uniform random chance)
14: Generate and evaluate B offspring as asexual mutations of selected parents
15: **for all** $g \in B$ **do**
16: Map g to its corresponding grid bin
17: **if** $g.fitness > bin.fitness$ **then**
18: Save g in the primary bin (discard current occupant)
19: **else if** random: 30% chance **then**
20: Save g in the secondary bin (discard current occupant)
21: **else**
22: Discard g
23: **end if**
24: **end for**
25: **end loop**

New Contribution: Multi-BC Quality Diversity Algorithms

As discussed in Chapter 2, Section 2, BCs that are strongly aligned with quality encourage the discovery of better behaviors simply by finding different behaviors, effectively bypassing the problem of deception that causes optimization-oriented search processes to become trapped in local optima. However, no such advantage exists for *unaligned* BCs, which often represent the most interesting and desirable types of diversity in practice. The experiments presented in Chapter 6 demonstrate that such unaligned BCs actually negatively impact the performance of QD algorithms, which on sufficiently-hard problems may translate into an outright failure to find the best-performing solutions. This observation raises the important question of how QD practitioners can find unaligned diversity without losing the ability to circumvent deception offered by aligned BCs.

This section offers a promising answer in the form of *driving search with multiple BCs simultaneously*. To explore this idea, the following algorithms represent various options for adapting QD to support more than one BC. In other words, candidate solutions have more than one behavior (each defined by its respective BC) that both influence the search process.

[NS-NS] Multi-BC Novelty Search

Novelty search can be extended to support multiple BCs simultaneously by calculating a separate novelty score for each BC and then combining these scores in a multi-objective formulation (via NSGA-II of Deb et al. [30]). Each BC maintains its own independent archive against which individuals are evaluated to determine their novelty score for that BC. There is only a single population where the breeding potential for each member is decided by a Pareto ranking according to the various novelty scores. Although NS is not itself a QD algorithm because it lacks a mechanism for discovered behaviors to increase in quality, when extended to include multiple BCs, NS-NS

can conceivably achieve some success if one BC serves to find diversity while another promotes increasing quality (e.g. an unaligned BC paired with an aligned BC). Note that even such a pairing does not quite embody the spirit of QD because any tendency towards increasing quality that emerges from an aligned BC is not explicitly *local*.

Pseudocode: [NS-NS] Multi-BC Novelty Search

Generate and evaluate B random genomes
Add B to $R1$ (aligned BC)
Add B to $R2$ (unaligned BC)
Add B to P
for all $g \in P$ **do**
    Calculate novelty score $g.nov1$ (against $R1 \cup P$ with aligned BC)
    Calculate novelty score $g.nov2$ (against $R2 \cup P$ with unaligned BC)
end for
Pareto-rank all $g \in P$ according to $(g.nov1, g.nov2)$
loop
    Select $bsize$ parents from P with chance inversely proportional to Pareto rank
    Generate and evaluate B offspring as asexual mutations of selected parents
    Add B to $R1$ (aligned BC)
    Add B to $R2$ (unaligned BC)
    Add B to P
    for all $g \in P$ **do**
        Calculate novelty score $g.nov1$ (against $R1 \cup P$ with aligned BC)
        Calculate novelty score $g.nov2$ (against $R2 \cup P$ with unaligned BC)
    end for

Pareto-rank all $g \in P$ according to $(g.nov1, g.nov2)$
Calculate novelty score $g.nov1$ for all $g \in R1$ (against $R1 \cup P$ with aligned BC)
Calculate novelty score $g.nov2$ for all $g \in R2$ (against $R2 \cup P$ with unaligned BC)
while $P.count > psize$ **do**
    Discard $g \in P$ with worst Pareto rank
end while
while $R1.count > r1max$ **do**
    Discard $g \in R1$ with lowest novelty score $g.nov1$
end while
while $R2.count > r2max$ **do**
    Discard $g \in R2$ with lowest novelty score $g.nov2$
end while
end loop

[NS-NSLC] Multi-BC Novelty Search with Local Competition

The idea of NS-NS can then be expanded to include a drive towards locally-increasing quality by adding a LC objective (in the same way as NSLC) where behavioral neighbors are decided by the unaligned BC that expresses the notion of diversity that the user is ultimately interested in collecting. The resulting NS-NSLC algorithm therefore includes three distinct objectives (combined via NSGA-II): (1) a quality-aligned novelty score to facilitate overcoming deception, (2) an unaligned novelty score for discovering new behaviors of interest, (3) an unaligned LC score to promote finding the best solution within each behavioral niche. This formulation can be thought of as augmenting unaligned-NSLC with an aligned BC to help achieve better solutions.

Pseudocode: [NS-NSLC] Multi-BC Novelty Search with Local Competition

Generate and evaluate B random genomes
Add B to $R1$ (aligned BC)
Add B to $R2$ (unaligned BC)
Add B to P
for all $g \in P$ **do**
    Calculate novelty score $g.nov1$ (against $R1 \cup P$ with aligned BC)
    Calculate novelty score $g.nov2$ (against $R2 \cup P$ with unaligned BC)
    Calculate local competition $g.lc$ (against $R2 \cup P$ with unaligned BC)
end for
Pareto-rank all $g \in P$ according to $(g.nov1, g.nov2, g.lc)$
loop
    Select $bsize$ parents from P with chance inversely proportional to Pareto rank
    Generate and evaluate B offspring as asexual mutations of selected parents
    Add B to $R1$ (aligned BC)
    Add B to $R2$ (unaligned BC)
    Add B to P
    for all $g \in P$ **do**
        Calculate novelty score $g.nov1$ (against $R1 \cup P$ with aligned BC)
        Calculate novelty score $g.nov2$ (against $R2 \cup P$ with unaligned BC)
        Calculate local competition $g.lc$ (against $R2 \cup P$ with unaligned BC)
    end for
    Pareto-rank all $g \in P$ according to $(g.nov1, g.nov2, g.lc)$
    Calculate novelty score $g.nov1$ for all $g \in R1$ (against $R1 \cup P$ with aligned BC)
    Calculate novelty score $g.nov2$ for all $g \in R2$ (against $R2 \cup P$ with unaligned BC)

25: **while** $P.count > psize$ **do**

26: Discard $g \in P$ with worst Pareto rank

27: **end while**

28: **while** $R1.count > r1max$ **do**

29: Discard $g \in R1$ with lowest novelty score $g.nov1$

30: **end while**

31: **while** $R2.count > r2max$ **do**

32: Discard $g \in R2$ with lowest novelty score $g.nov2$

33: **end while**

34: **end loop**

[ME-ME] Multi-BC MAP-Elites

In an effort to maintain its characteristic simplicity, MAP-Elites is extended to support multiple BCs by maintaining a separate grid for each BC (with similar maximum sizes). On each iteration of ME-ME, an equal amount of parents are selected from each grid and their resulting offspring are mapped to *both* grids (where the decisions to save or discard them are performed independently; e.g. between two grids, a single offspring may be saved in one, both, or neither).

Pseudocode: [ME-ME] Multi-BC MAP-Elites

1: Generate and evaluate B random genomes

2: **for all** $g \in B$ **do**

3: Map g to its corresponding grid bin in $P1$ (aligned BC)

4: **if** $g.fitness > bin.fitness$ **then**

5: Save g in the bin (discard current occupant)

6: **else**

7: Discard g

8: **end if**

9: Map g to its corresponding grid bin in $P2$ (unaligned BC)

10: **if** $g.fitness > bin.fitness$ **then**

11: Save g in the bin (discard current occupant)

12: **else**

13: Discard g

14: **end if**

15: **end for**

16: **loop**

17: Select $bsize/2$ parents from $P1$ (uniform random chance)

18: Select $bsize/2$ parents from $P2$ (uniform random chance)

19: Generate and evaluate B offspring as asexual mutations of selected parents

20: **for all** $g \in B$ **do**

21: Map g to its corresponding grid bin in $P1$ (aligned BC)

22: **if** $g.fitness > bin.fitness$ **then**

23: Save g in the bin (discard current occupant)

24: **else**

25: Discard g

26: **end if**

27: Map g to its corresponding grid bin in $P2$ (unaligned BC)

28: **if** $g.fitness > bin.fitness$ **then**

29: Save g in the bin (discard current occupant)

30: **else**

31: Discard g

end if

end for

end loop

[MENOV-MENOV] Multi-BC MAP-Elites + Novelty

Finally, MENOV is extended to MENOV-MENOV similarly to ME-ME except where each grid also maintains its own novelty archive and selection within each grid is proportional to novelty.

Pseudocode: [MENOV-MENOV] Multi-BC MAP-Elites + Novelty

Generate and evaluate B random genomes

for all $g \in B$ **do**

    Add g to $R1$ (aligned BC)

    Add g to $R2$ (unaligned BC)

    Map g to its corresponding grid bin in $P1$ (aligned BC)

    if $g.fitness > bin.fitness$ **then**

        Save g in the bin (discard current occupant)

    else

        Discard g

    end if

    Map g to its corresponding grid bin in $P2$ (unaligned BC)

    if $g.fitness > bin.fitness$ **then**

        Save g in the bin (discard current occupant)

    else

        Discard g

end if

end for

Calculate novelty scores $g.nov1$ for all $g \in P1$ (against $R1 \cup P1$ with aligned BC)

Calculate novelty scores $g.nov2$ for all $g \in P2$ (against $R2 \cup P2$ with unaligned BC)

loop

    Select $bsize/2$ parents from $P1$ with chance proportional to novelty score $g.nov1$

    Select $bsize/2$ parents from $P2$ with chance proportional to novelty score $g.nov2$

    Generate and evaluate B offspring as asexual mutations of selected parents

    for all $g \in B$ **do**

        Add g to $R1$ (aligned BC)

        Add g to $R2$ (unaligned BC)

        Map g to its corresponding grid bin in $P1$ (aligned BC)

        if $g.fitness > bin.fitness$ **then**

            Save g in the bin (discard current occupant)

        else

            Discard g

        end if

        Map g to its corresponding grid bin in $P2$ (unaligned BC)

        if $g.fitness > bin.fitness$ **then**

            Save g in the bin (discard current occupant)

        else

            Discard g

        end if

    end for

    Calculate novelty scores $g.nov1$ for all $g \in P1$ (against $R1 \cup P1$ with aligned BC)

    Calculate novelty scores $g.nov2$ for all $g \in P2$ (against $R2 \cup P2$ with unaligned BC)

Calculate novelty scores $g.nov1$ for all $g \in R1$ (against $R1 \cup P1$ with aligned BC)
Calculate novelty scores $g.nov2$ for all $g \in R2$ (against $R2 \cup P2$ with unaligned BC)
while $R1.count > r1max$ **do**
    Discard $g \in R1$ with lowest novelty score $g.nov1$
end while
while $R2.count > r2max$ **do**
    Discard $g \in R2$ with lowest novelty score $g.nov2$
end while
end loop

New Directions for Novelty Archive Management

Many of the QD approaches featured herein are based on the core novelty search algorithm whose functionality critically depends on a core component: the novelty archive (a memory of previously visited parts of the behavior space). Importantly, the archive must decide when and how to admit new individuals and discard old ones. While Section 5 describes how this collection is managed in the experiments to follow, it is worth briefly exploring alternative methods and how they may affect the search.

In its first published iteration, novelty search calculates the novelty scores against an archive that indefinitely increases in size as new individuals are admitted [59]. It does this by adding all individuals whose initial novelty score exceeds a threshold, effectively remembering all parts of the behavior space that have ever been explored. However, in this dissertation all collections (including the novelty archive) enforce a maximum capacity whose primary purpose is to keep utilization of computation resources (CPU cycles and memory) relatively constant in the interest of allowing search to run indefinitely. This saving of resources comes at the cost of *forgetting* information over

time, requiring an individual be removed from the archive every time a new one is added. Here, we remove individuals from the archive by calculating their novelty score and discarding the lowest, ensuring that newly explored areas of the behavior space come to be represented in the archive at the expense of old areas that are over-represented. Over time, the novelty archive spreads to cover more and more of the behavior space, at a progressively courser granularity (a larger space between individuals). Eventually, old areas of the space may become sparse enough to admit new individuals to fill the gaps. In this way, novelty search with a fixed-capacity archive may sometimes be encouraged to re-visit previously explored behaviors later on in search – a practice that is strongly discouraged by an archive that does not forget (i.e. no individuals are ever removed) even if it might sometimes be advantageous. For example, individuals discovered later in search have different genomes (that are likely more complex) than those discovered earlier. These new genomes may contain evolutionary stepping stones to access new areas of the behavior space that were not accessible before. However, when the archive saves individuals forever, search is discouraged from visiting any old areas, opening the possibility of "walling off" unexplored regions of the behavior space where all stepping stones into that region are already well-represented in the archive and thus will never have a chance to breed.

Enforcing archive capacity with random tournaments (as described in Section 5) not only saves CPU cycles over the more accurate alternative, re-calculating novelty scores for the entire archive after each addition and then deleting the lowest; it also creates opportunities for holes to sometimes open up in the archive even in areas that are already sparsely-represented, further encouraging the re-visitation of old behaviors. Re-visitation can be tuned by the user by changing the tournament size. With larger tournaments, such holes open up less often.

Archive capacity may also be enforced by deleting the *oldest* individual each time a new one is added, essentially guaranteeing the re-visitation of old behaviors. However, under this regime, search may get stuck in a loop searching the same set of behaviors over and over.

For a detailed empirical study comparing different methods of configuring novelty search, see Gomes et al. [44].

New Perspectives on QD Collection Mechanisms

While each of the algorithms described in Section 5 are designed to explore the entire behavior space while simultaneously finding improved solutions within each known region, the question remains how to best sort through all individuals encountered during a run to return a set of the best and most diverse results: *the QD collection*. Unlike conventional fitness-driven evolutionary algorithms where the final result of a run is simply the solution with the highest fitness, QD algorithms must return a collection of individuals that represents the best individuals encountered in each region of the search space.

As search may focus on different regions of the behavior space at different points in the run, it is inadequate to simply return the final population at the end of the run. Instead, a more sophisticated mechanism must consider each new individual to decide whether it should be admitted to the collection so that the final collection represents the best and most diverse of all individuals encountered over the entire run. Furthermore, as the size of the known behavior space expands over the course of the run and new individuals are generated potentially indefinitely, it is also necessary to consider which members to delete from the QD collection as new members are admitted to keep the total collection at a reasonable size; such deletion rules must be carefully designed to minimize the loss of high-quality results from important regions visited in the past.

The two QD collection strategies (termed "structured" and "unstructured" appearing in the next sections describe rules for sorting through the many thousands or millions of individuals encountered over the course of evolution and returning to the user a succinct representation of the best

results at the end of the run (or of the best results encountered so far if the QD collection is sampled in the middle of a run). Importantly, the QD collection and it's rules are kept separate from the evolutionary algorithm and do not interfere in the search in any way. In other words, the QD collection strategies described here may be viewed as a form of post-hoc analysis.

Structured QD Collection

One convenient method of managing the QD collection is to manage it in the style of MAP-Elites [78]. First, the entire behavior space is divided into a grid of bins formed my discretizing each dimension of the behavior characterization into N equally-sized units. Every new individual encountered by evolution is mapped to the grid bin corresponding to its behavior where it is added only if it has a higher fitness value than the individual currently occupying that bin (in which case the previous occupant is removed from the collection). An individual mapped to an empty bin is always added. In this way, at any point in the run the QD collection contains the highest quality specimen seen so far in every region of the behavior space.

This style of QD collection, termed here as the *structured QD collection* (sharing the naming convention for diversity containers set forth in Cully and Demiris [21]) can be applied even to algorithms other than MAP-Elites (e.g. NSLC) because the grid is maintained in the background and does not interfere in the search process.

The size of the QD collection is set to a reasonable level for later analysis by adjusting the value of N. Just as in MAP-Elites, the number of bins in the grid is an exponential function of the number of dimensions in the behavior characterization:

$$\text{number of bins} = M^N$$

where M is the dimensionality of the BC and N is the granularity of discretization.

While the structured QD collection strategy is desirable because of its simplicity and even coverage of the behavior space, it is ideal only for low-dimensional BCs (< 6). For higher-dimensional BCs, the granularity of discretization is forced to be prohibitively small such that even very different behaviors may map to the same bin. Furthermore, under the regime of the structured QD collection, the granularity of sampling is fixed at the beginning of the run. However, in some domains (such as those where large portions of the behavior space are initially inaccessible) it may be desirable that the granularity of sampling grow dynamically as new regions of the behavior space are discovered.

To address these shortcomings and facilitate QD even for high-dimensional BCs, an alternate QD collection strategy is proposed here, called the *unstructured QD collection*.

Unstructured QD Collection

Several strategies have been explored in the literature for returning a QD collection without discretizing the behavior space. The simplest such strategy is to return the population or the novelty archive at the end of the run [62]. However, the population or archive at any given point in a run (including the last generation) is not necessarily representative of everything that has been explored over the entire run.

An alternate strategy is to pare down the dimensionality of the behavior vectors post-hoc via principal component analysis (PCA) where the lower-dimensional space can then be evenly sampled [122]. However, PCA does not always work well for high dimensional vectors [52]; additionally, this strategy requires saving all individuals encountered during a run which may be prohibitive because of memory or disk space limitations.

Cully and Mouret [22] introduces a variant of NSLC's novelty archive called the *behavior reper-*

toire wherein archive members are continually replaced by better-performing individuals as they are encountered during search. One concern with this approach when applied to the archive against which novelty scores are computed is that polluting the archive with fitness pressure may interfere with its ability to accurately calculate novelty. For example, continually replacing archive members with better-performing individuals may cause them to migrate away from low-fitness areas of the behavior space, thus making such areas appear more novel than they actually are (the reverse is true for high-fitness areas).

To preserve the fitness-agnostic properties of novelty search, here a modification of the behavior repertoire called the *unstructured QD collection* mechanism operates in the background independently of the novelty archive, considering all individuals that the search encounters without interfering with the search itself. A small collection is maintained that is initially populated by the first few iterations of search until it reaches capacity. Once the collection is at capacity, the goal is to admit new members and remove old members such that the total diversity within the collection (i.e. its coverage of the behavior space) continually increases while simultaneously improving the quality of represented regions.

For every new individual encountered during evolution, an *insertion tournament* is held between the new individual and a number of randomly selected members of the QD collection. The tournament participants each calculate their novelty scores against the QD collection in the same way that NS computes novelty against the archive: by calculating their behavioral distance against all members of the QD collection and summing the n (here, $n = 12$) smallest distances. Then, the participant with the lowest novelty score compares its fitness against its closest neighbor and the individual with higher fitness is allowed into the QD collection while the lower fitness individual is removed or discarded. This process effectively discards a low fitness individual from the least novel region of the collection. In this way, the QD collection continually expands its coverage of the behavior space while simultaneously *locally* improving fitness. Tournaments are held instead

of full novelty calculations so that QD collection can operate quickly regardless of collection size. Additionally, tournaments introduce some noise into the process, therefore allowing the QD collection to shift around the behavior space over time in case the space is too vast to be adequately covered by a single snapshot of the collection.

CHAPTER 6: EXPERIMENT: BEHAVIOR CHARACTERIZATION ALIGNMENT

The experiments in the following chapter were originally published in Pugh et al. [90] as a first study comparing different approaches to QD.

In the traditional optimization paradigm, the user is interested in finding the best possible solution to a given problem. Candidate solutions are evaluated according to how well they satisfy the problem and are given a score commensurate with their success. For example, when trying to find an effective controller for a bipedal robot, each controller may be scored based on how far it can make the robot walk in a given time frame – the goal being to find the controller that makes the robot walk the furthest. The function that assigns scores to candidate solutions based on their task performance is called the *fitness function* and must be defined by the user.

Unlike in simple optimization where the sole interest is to find the single best solution, the goals in quality diversity are twofold: (1) to find as many different kinds of solutions as possible (diversity) and (2) to find the best possible solution of each kind (quality). While the second goal is easily serviced by a fitness function, addressing the first goal requires the user to define what is meant by "different kinds of solutions."

Only by specifying characteristics (or *features*) of interest can we measure the similarity (or dissimilarity) between two candidate solutions. For example, if a user wants to find many different kinds of virtual creatures, they must first specify what traits of the virtual creatures they want to see differ across the collection. Possible traits might include color, height, mass, or number of legs. By specifying "number of legs" as a feature of interest, the user is saying that they want to see creatures with many different numbers of legs, which requires knowing how similar two creatures

are in terms of their number of legs (QD would thus look for dissimilar numbers of legs). For the purposes of measuring the degree of similarity between two creatures, each characteristic should be measurable by a scalar value. Together, all of the features of interest combine to form the *behavior characterization* (BC). A vector of numbers formed by concatenating the scalar values for each feature in the BC is called a *behavior* (even if it refers to concrete aspects of the phenotype such as the size or number of legs in an evolved morphology). The difference between two behaviors can then be quantified as the Euclidean distance between two BC vectors. In this way, the behavior characterization defines a coordinate space called the *behavior space* where each behavior is one point in the space.

The diversity goal of quality diversity, "to find as many different kinds of solutions as possible," can be restated as: to explore all regions of the behavior space by finding solutions in all parts of the space. The ideal end product of quality diversity is a collection of solutions spanning all regions of the behavior space to some granularity, where each member of the collection represents the highest potential fitness in their respective region. In the virtual creatures example, the goal of quality diversity may be to find the fastest moving creatures of all different shapes and sizes. In another application, the goal may be to find the fastest moving creatures of all different numbers of legs; in yet another: all different ways of walking. The diversity goal in QD depends on the features selected by the user in the BC.

In quality diversity, the BC has another, secondary purpose alongside defining the type of diversity that will be collected by QD. In the original experiments that introduced Novelty Search, Lehman and Stanley [59] show that searching for behavioral novelty alone can solve a deceptive maze task faster and more reliably than an objective-oriented fitness function. In these experiments, the behavior characterization (the final location of a robot in the maze at the end of its trial) defines a behavior space in which finding new regions eventually leads to finding the goal, avoiding the problem of *deception* (wherein chasing higher fitness tends to lead search *away* from the solution)

altogether. In this way, the BC serves the secondary purpose of driving search towards higher performing solutions where fitness alone may be insufficient (e.g. due to deception in the fitness landscape).

An important property of the BC is its alignment with the notion of quality, which refers to the *degree to which finding novelty tends also to lead to higher fitness*. For example, in a maze, if the BC is based on the final position reached (as in the original NS experiments), then it is highly aligned because continually finding agents that reach new final positions will eventually find an agent that reaches the endpoint of the maze. While BC alignment can be difficult to measure a priori (just as the shape of fitness landscapes are not known a priori for any challenging problems of interest), a BC's degree of alignment can be anticipated by considering two key properties of highly-aligned BCs: (1) each behavior is associated with only a narrow range of fitness values (e.g. a robot's final position in a maze is associated with exactly one fitness value: it's closeness to the goal) and (2) the maximum possible fitness in adjacent regions of behavior space correlates (e.g. nearby positions in a maze generally have similar fitness).

Searching for novel endpoints is known to be an adept strategy for solving a maze task; however, users of QD algorithms may be interested in finding a diversity of maze-driving robots with respect to *other* (perhaps more interesting) behavior characterizations that may be less well-aligned with the objective of solving the maze. For example, users might want to find maze-solving robots with different styles of driving or that follow different trajectories through the maze. Can novelty search or QD algorithms in general solve a deceptive maze task with any arbitrary behavior characterization (in particular one that is unaligned with quality)? Furthermore, are they able to find many *different* solutions?

While the literature analyzing the nuances of algorithms in EC and elsewhere for the purpose of optimization is vast, almost no studies to date investigate the tradeoffs among different approaches

to QD or how they each interact with different behavior characterizations.

The following experiment begins such an investigation by comparing the two main QD algorithms, Novelty Search with Local Competition (NSLC) and Multi-dimensional Archive of Phenotypic Elites (MAP-Elites), along with a new variant proposed here called MAP-Elites + Novelty (ME-Nov), on a new benchmark maze-solving task designed to allow multiple correct paths to the goal. Additionally, while not themselves actual QD algorithms, both NS and Fitness search are included as controls, representing searching purely for diversity and purely for quality respectively. Full descriptions of each algorithm can be found in Chapter 5, Section 5. To explore the effect of varying the behavior characterization, each algorithm is tested with a range of different BCs with varying levels of alignment in order to examine both the effect of alignment on solving the task as well as how alignment interacts with each candidate algorithm.

QD-Maze

To effectively tease out the differences between competing quality diversity (QD) algorithms, a test domain is needed with both (1) a concrete measure of quality, and (2) a diversity of legitimate solutions. No standard such benchmark domain yet exists in the area of QD, and thus one of the contributions of this dissertation is to propose one suited specifically to the concerns of QD.

The first experiments with novelty search [59, 61] introduced a domain called *HardMaze* that has since become popular within the literature because of its explicit visual depiction of deception within a fitness landscape (figure 6.1). The catch in HardMaze is that for a robot to navigate to the goal point of the maze it is necessary first to navigate *away* from the goal multiple times along the correct path. Additionally, there is an easily-reachable cul-de-sac that is close to the goal in terms of Euclidean distance but is actually farther away from the goal in terms of distance

along the correct path. Importantly, searching for behavioral novelty alone (novelty search) in the HardMaze solves the maze much faster than a conventional objective-based search wherein fitness is the Euclidean distance to the goal point at the end of a trial (which, unlike novelty search, is easily drawn into the cul-de-sac). Thus, in HardMaze, Euclidean distance serves as a concrete measure of quality. However, HardMaze only contains *one* correct path through the maze (i.e. the diversity of legitimate solutions is limited), diminishing its utility for testing QD algorithms, which are intended to seek many different solutions.

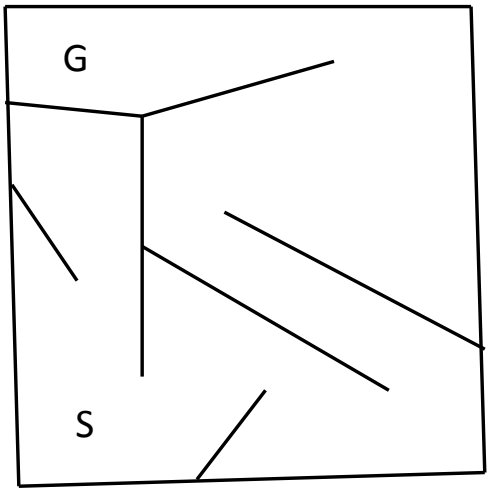


Figure 6.1: **HardMaze (Not in this experiment).** This deceptive maze task was first introduced in Lehman and Stanley [59] to demonstrate the ability of novelty search to solve a deceptive problem that objective fitness could not. Individuals start at the point S and must navigate to point G. The cul-de-sac above the start point serves as a *deceptive trap* that appears close to the goal but is actually in the wrong direction for solving the maze.

Instead of HardMaze, in this experiment we augment the idea of a maze navigation domain that uses Euclidean distance as a concrete measure of quality to also support a diversity of possible solutions to the maze. This new domain, QD-Maze, still contains several deceptive traps (albeit less difficult than those in HardMaze: all algorithms here tend to find at least one solution) while also allowing many possible "correct paths" to the goal, thus facilitating the expression of a diversity

of solutions (figure 6.2).

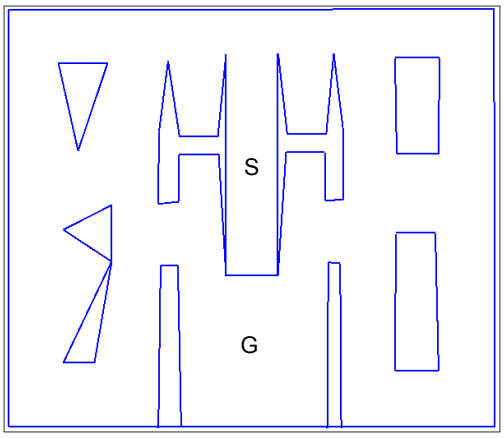


Figure 6.2: **Quality diversity maze (QD-Maze).** Individuals start at the point S and the highest quality solutions navigate to point G. While the maze is deceptive enough to challenge objective-based algorithms, it also contains a variety of solutions and thus serves as a benchmark for combining quality with diversity.

Experiment

Aside from the change in map configuration, the experimental setup for QD-Maze is identical to that in the well-established HardMaze domain. A robot agent is given a fixed amount of time to move around the maze before the trial ends and the agent's *quality* is scored according to the Euclidean distance between its final position and the goal point (the score is maximum if the goal point is reached). Reaching the goal point prematurely ends the trial. Agents are equipped with a set of six rangefinder sensors, five of which are equally spaced across the front-facing 180 degrees and the sixth facing the rear (figure 6.3). Additionally, four pie slice sensors (front, right, rear, and left -facing) detect the direction of the goal by activating the sensor facing the goal. The agent always moves forward at maximum speed and is equipped with a single output to control turning left or right. In the following experiment, agents are driven by artificial neural networks evolved by

NEAT [116] with crossover removed. That is, controller networks begin with the ten input sensors directly connected to the single left-right turning output and the connection weights and network structure are evolved over time by random mutations.

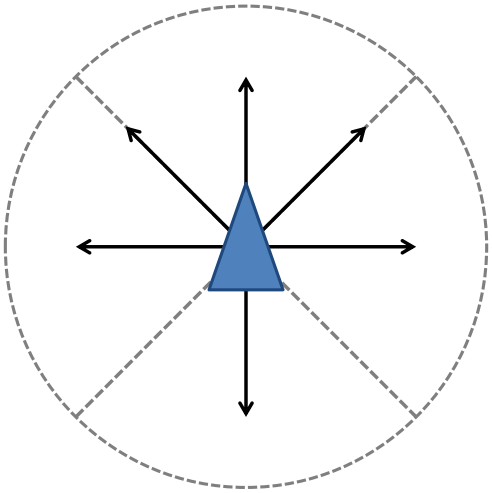


Figure 6.3: **Agent Sensor Configuration.** Maze-navigating agents are equipped with two sets of sensors: (1) six rangefinder sensors (depicted by black arrows) that activate upon intersection with a wall and (2) four pie slice sensors (represented by gray dotted lines) that activate according to the distance and direction of the goal.

Behavior Characterizations (BCs)

The nature of the "diversity of solutions" available for discovery within the search depends heavily on how behaviors are characterized. In the original HardMaze experiments [59], agent behaviors are characterized according to their final position at the end of the trial (**EndpointBC**). This BC is notable for its strong alignment with the implicit objective of a maze. In other words, the neighborhood of behaviors around any particular trajectory endpoint generally share a similar level of quality, thereby aligning behavior with a notion of quality. Under the pressure of a strongly-aligned BC, searching for novelty *alone* may often be enough to find not only diversity, but also

high-quality solutions to the domain, without even the need to explicitly optimize each behavioral neighborhood independently. To investigate the effect of varying the degree of the BC's alignment with the objective, three additional BCs are also tested, each less aligned than the previous: FullTrajectoryBC, HalfTrajectoryBC, and DirectionBC.

Under **FullTrajectoryBC**, the agent's position is sampled at three points throughout its trial: one-third, two-thirds, and at the end of the trial; all three (x, y) points are concatenated together to form the six-dimensional BC vector. This BC is less strongly aligned with the notion of quality than EndpointBC because only the final sampled point has a strong correlation to the final score within the domain. Even less aligned again is **HalfTrajectoryBC**, wherein the agent's position is sampled at three points throughout *only the first half* of its trial. There is enough time remaining in the second half of the trial for an agent to reach the goal from any location within the maze, rendering HalfTrajectoryBC even less aligned with the objective than FullTrajectoryBC because in theory, every behavior under HalfTrajectoryBC can potentially achieve a maximum quality score (i.e. ultimately solving the maze).

The final and least-aligned BC, **DirectionBC**, samples the direction the robot is facing on each tick (north, east, south, or west) and divides the duration of the trial into five equal-length time slices. The BC then consists of five dimensions, each corresponding to the direction that the agent was facing most often during each of the five time slices (north $= 0.125$, east $= 0.375$, south $= 0.625$, west $= 0.875$)[1]. This characterization is almost completely agnostic to the notion of quality because the goal point of the maze can be approached from any direction and the agent is free to wander around the maze and face any direction at any time. Under DirectionBC, every possible behavior in the behavior space can potentially achieve a maximum score on the objective of reaching the goal and there is little to no correlation between quality and behavioral neighborhoods.

[1]For the purpose of calculating behavioral distance with DirectionBC, the range 0.0 to 1.0 is assumed to "wrap around" such that value 0 is maximally distant from 0.5 but zero distance from value 1.0.

QD Algorithms

A variety of algorithms can collect QD in a given domain, some designed explicitly to do so and some more as a side effect. In this experiment, we compare the performance of three existing algorithms: novelty search (NS), novelty search with local competition (NSLC), and MAP-Elites (the latter two designed explicitly for QD), as well as a new variant of MAP-Elites introduced here where selection is proportional to novelty (MAP-Elites + Novelty; MENOV). A conventional fitness-based search (which is not expected to excel at QD) also serves as a baseline for comparison. Each of these algorithms is described in more detail in Chapter 5, Section 5.

To normalize comparisons across the algorithms as much as possible, the SharpNEAT [46] distribution of the NEAT algorithm [116] serves as the underlying neuroevolution platform for all of them. Therefore, the neural networks are able to evolve increasing complexity from minimally-complex starting networks, a central feature of NEAT. However, except in the pure fitness-driven baseline runs, to minimize potentially confounding factors, the conventional genetic diversity component of NEAT (also called *speciation*) is excluded. That way, the diversity facet of QD is exclusively focused on *behavior*, which is the focus of the majority of recent innovation in non-objective search algorithms and has exhibited significantly more impact on performance in such algorithms than genetic diversity [80]. Furthermore, because of preliminary experimental evidence that generational evolution can lead to instability or "jumping around the behavior space" in non-objective search populations, all algorithms are run in *steady state* mode, which means that only a subset of the population (in this case 32 individuals evaluated in parallel) is replaced at a time instead of the entire generation (which is similar to the rtNEAT [119] variant of NEAT). In total five algorithms are compared.

The classic **novelty search (NS)** [59, 61] algorithm *only* searches for behavioral diversity with no notion of quality. It rewards behaviors that are maximally distant from previously discovered

behaviors. However, if the BC is aligned with a notion of quality, then it is possible that NS without any special augmentation can itself collect QD effectively, a possibility thereby investigated in this experiment.

NS is augmented to explicitly seek QD in the **novelty search plus local competition (NSLC)** [62, 122] algorithm. In this variant, a multi-objective formulation implemented through NSGA-II [30] casts classic novelty as one objective and *local competition* (LC), which means a quality rank relative only to those with a *similar* BC, as another. The LC component encourages unique behaviors to improve in quality within their local behavioral neighborhoods, thereby aligning the search explicitly with the notion of QD. The LC approach contrasts with the idea of combining novelty with a *global competition* objective (NSGC), which has been previously shown less effective than NSLC for discovering a diversity of solutions within a search space [62] because global competition explicitly prunes out the diversity contained within areas of lower fitness potential, and is therefore not considered in this experiment.

Unlike NS and NSLC, the recent **MAP-Elites (ME)** algorithm [23, 78, 82] divides the behavior space of the BC into discrete bins. Each bin then remembers the fittest (elite) genome ever to occupy it, and only one genome occupies any given bin at one time. Thus the elites within each bin capture the notion of quality and the whole set of bins captures the notion of diversity. In the classic formulation of ME, selection is very simple: each filled bin has an equal chance of being selected to produce an offspring. Thus search effort tends to be spread uniformly across the known behavior space (rather than explicitly biased by novelty). One technical limitation of ME is that the BC must be constrained to a low number of dimensions because the total number of bins grows exponentially with BC dimensionality; for this reason and to keep the comparisons fair, none of the BCs featured in this experiment have more than six dimensions.

While the simplicity of selection in ME is appealing, its effect is that evolution is more likely

to concentrate resources in regions of the behavior space that are already filled (because that is where most of the bins are occupied). While this approach has proven effective in ME applications so far, it is easy to augment ME to concentrate search effort on more novel regions by adding novelty pressure. In this **MAP-Elites+Novelty (ME-Nov)** variant tested for the first time here, the probability of selecting an occupied bin for reproduction is proportional to its novelty, creating a stronger pressure towards diversity than in the original ME. However, once all the bins are filled in ME-Nov, it collapses back to an approximation of regular ME. The question is whether the added initial pressure towards novelty might enhance ME's drive towards QD. Also, while ME has effectively filled most bins in experiments published so far [23, 82], in future domains in which admission to some bins requires very advanced behavior and therefore these bins are unlikely to all be filled, the additional novelty pressure in ME-Nov could prove instrumental in covering as much of the space as possible.

Finally, a regular **fitness-based search (fitness)** (implemented as NEAT [116]) simply searches for objectively superior performance (though with NEAT's conventional genetic speciation fitness-sharing mechanism to give it at least some hope of maintaining diversity). This variant, which is susceptible to the trap of deception [61] and has no mechanism for generating behavioral diversity, helps to highlight the need for specialized approaches to the problem of QD.

Metric: QD Score

To compare each treatment in terms of their ability to collect QD, a quantitative metric is needed that captures both the diversity of solutions discovered so far as well as the quality of those solutions. Diversity can be quantified by the number of niches that have been discovered out of all possible niches (i.e. coverage of the behavior space at some granularity of niche size) and quality can be quantified by the best fitness individual found within each niche. Both of these components

are present in a MAP-Elites grid: the entire behavior space is divided into a number of discrete bins (niches) where each bin remembers the highest fitness individual it has encountered so far. Thus, for MAP-Elites, the "amount of QD" collected at any point in time can be calculated by adding together the highest fitness seen by each bin (unvisited bins are regarded as 0.0 fitness).

Interestingly, such a MAP-Elites style grid can be maintained in the background for *all* variant algorithms, even those that are not MAP-Elites. As long as the BC has few enough dimensions to be discretized (as is true for all BCs in this experiment), the best-performing individual found for every bin in the behavior space (regardless of the method through which it was found) can simply be stored for later analysis.

To support QD collection as well as effective operation of the MAP-Elites algorithm itself (where the grid constitutes the active population), the behavior space for each BC is divided into an appropriately small number of bins by discretizing each dimension of the space into n units such that the resulting number of bins falls within the same order of magnitude as the size of the breeding population in the NS, NSLC, and fitness-based treatments: 900 bins for EndpointBC, 729 for FullTrajectoryBC and HalfTrajectoryBC, and 1,024 for DirectionBC. For each treatment, all individuals encountered by evolution are considered for inclusion in this grid of elites. The performance (*QD score*) of each treatment at any given point during the run is then measured as *the total fitness across all filled grid bins within the QD collection* (where higher fitness means the robot ending its trial closer to the goal in terms of Euclidean distance).

Throughout evolution, QD score can therefore be improved by either (1) discovering higher fitness solutions for existing bins (increasing quality) or (2) discovering more bins (increasing diversity). To achieve the highest possible QD score, an algorithm must excel with respect to both quality and diversity.

Experimental Setup and Parameters

Each algorithm (NS, NSLC, ME, ME-Nov, fitness) is combined with each of the behavior characterizations (EndpointBC, FullTrajectoryBC, HalfTrajectoryBC, DirectionBC) to comprehensively test the effects of both factors on QD. Except for the DirectionBC variants, each combination was evaluated over 20 runs capped at $250{,}000$ evaluations. Because the DirectionBC variants take longer to converge to stable scores and a change in algorithm ranking takes place around 250,000 evaluations, DirectionBC runs are capped at 750,000 evaluations instead.

The population size for NS, NSLC, and fitness is 500; NS and NSLC also keep an associated novelty archive of maximum size 2,500, after which the least novel entry is removed when a new entry is added. For MAP-Elites, as is standard, the population size at any given time is the number of occupied bins (the maximum number of which are listed in the previous section for each BC). To enable parallel evaluation and more stable novelty scores, evolution proceeds as steady state with a batch size of 32.

Following the settings in the original novelty search experiments [61], connections are mutated with probability 60%, connections are added with probability 10%, and neurons are added to the network with probability 0.5%. Evolved networks are constrained to be strictly feedforward. All other settings follow standard SharpNEAT 1.0 defaults [46]. Source code for the entire experimental setup is available at `http://eplex.cs.ucf.edu/uncategorised/software#QD`.

Performance for each method and BC combination is measured according to the total fitness across all filled grid bins (Section 6). Results are averaged across 20 runs for each of the 20 algorithm$\times$BC combinations (for a total of 400 runs). Additionally, pairwise comparisons are performed after every 160 evaluations to determine when exactly there are significant differences between methods. Tukey's test (with $p < 0.05$) is used to establish significance instead of the Student's t-test to

account for the statistical problem of multiple pairwise comparisons.

Results

Figures 6.4, 6.5, 6.6, and 6.7 show for each BC the total quality diversity (QD) discovered over time by each method, averaged over 20 runs. Fitness, which is included only as a baseline for comparison, unsurprisingly performs significantly below all other methods. Among the other four methods (NS, NSLC, ME, and ME-Nov), a yellow bar at the top of each graph highlights the period during which significant differences are observed between the best and worst performing methods at that point in time. Note that in terms of both QD score and significance testing, numerical comparisons are only meaningful between methods *within the same BC* because the total possible QD score is different for each BC. The main result is that the ranking of methods depends on the alignment of the BC with the notion of quality, suggesting that choosing a QD method for a particular problem can potentially become a principled decision in the future.

With EndpointBC (figure 6.4), which is highly aligned with the notion of quality (i.e. closeness to the endpoint), the NS-based methods (NS and NSLC) significantly outperform all other methods for most of the run, with NS ending significantly (though only slightly) above NSLC. ME-Nov also significantly outperforms original ME, though is still significantly below NS and NSLC. The superior performance demonstrated here by novelty alone (NS) highlights that when the BC is strongly aligned with fitness, additional sophisticated machinery designed to simultaneously reward quality and diversity may be unnecessary (perhaps depending on the difficulty of the domain) as achieving diversity by itself yields higher quality solutions as a side-effect.

However, as BC alignment is reduced, quality-seeking mechanisms become more important. In FullTrajectoryBC (figure 6.5), which is still fairly aligned with the notion of quality (because

the last point in the trajectory is EndpointBC itself), NS, NSLC, and ME-Nov are tied (i.e. not significantly different from each other) throughout most of the run. Here, only ME performs significantly worse than the other methods. NS retains its utility as a top choice, but no longer holds an advantage over NSLC or ME-Nov. ME's low performance on highly-aligned BCs can be explained by its novelty-agnostic selection. In highly-aligned behavior spaces, finding different solutions inevitably leads to finding better solutions. However, ME does not exploit this property and thus lags behind other approaches that do (including ME-Nov).

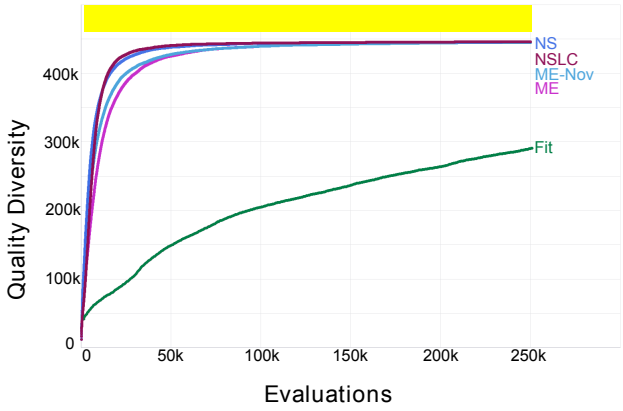


Figure 6.4: **EndpointBC (very high alignment).** The QD score over time for each method×EndpointBC combination is depicted, averaged over 20 runs. The yellow strip along the top of the graph indicates periods during which there is a significant difference at least between the best and worst performing methods (excluding fitness, which is always significantly worse than all other methods). The method labels are listed from top to botom in their rank order during the period of significance. Note that because of the large QD scale, sometimes significant differences exist even when not visually apparent. For EndpointBC, NS performs best, followed by NSLC.

HalfTrajectoryBC (figure 6.6) is even less aligned with quality, which leads to an interesting phenomenon seen only with this BC: after a very brief period at the start of the run, all methods effectively perform the same (although ME still appears slightly worse, the difference is not significant). In effect, HalfTrajectoryBC hits just the right balance of misalignment that the advantages and disadvantages of different methods becomes a wash. Surprisingly, even NS alone remains competitive, despite its lack of an explicit quality-seeking component.

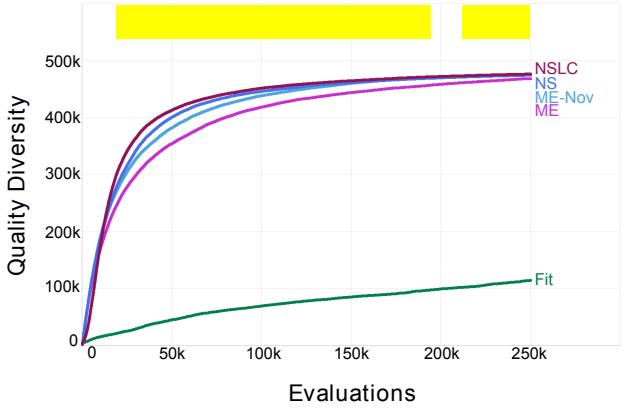


Figure 6.5: **FullTrajectoryBC (high alignment).** The QD score over time for each method×FullTrajectoryBC combination is depicted, averaged over 20 runs. For this BC, which is slightly less aligned with fitness than EndpointBC, NS, NSLC, and ME-Nov are effectively tied for most of the run, with ME lagging significantly behind.

Finally, DirectionBC (figure 6.7), which only tracks the direction the agent is facing, hardly aligns with fitness at all because it is almost completely detached from the agent's position on the map (and thus its proximity to the goal). For example, an agent can pause and face in any direction for a majority of ticks in each timeslice before continuing on a completely unrelated trajectory. In other words, any point in the DirectionBC behavior space can achieve practically any fitness. Note that such low alignment is not unprecedented in serious applications. For example, a QD algorithm might search for diversity in terms of the height and mass of virtual creatures, which can be hardly predictive of their ability to walk [62].

The results with DirectionBC suggest that an unaligned BC can upend the utility of NS as an implicit quality-seeking mechanism. In this scenario, NS drops to the bottom of the ranking, performing significantly below other approaches (each of which have explicit quality-seeking components). Furthermore, while NSLC exceeds NS significantly (demonstrating that the LC component of NSLC can indeed provide an advantage in QD), both ME and ME-Nov significantly outperform NSLC. ME effectively ties with ME-Nov for the entire run, further suggesting that the advantage

of novelty pressure is diminished in the face of an unaligned BC.

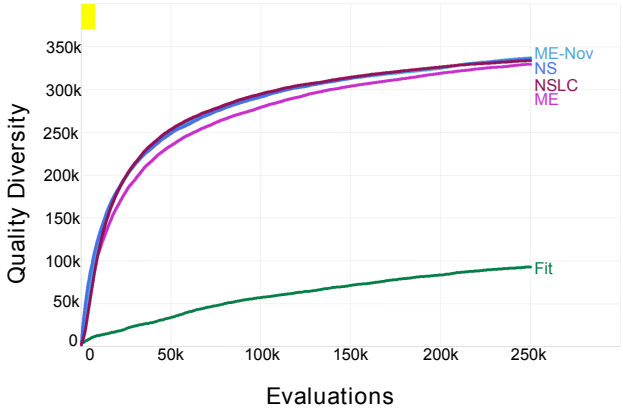


Figure 6.6: **HalfTrajectoryBC (modest alignment).** The QD score over time for each method×HalfTrajectoryBC combination is depicted, averaged over 20 runs. All methods except fitness are tied with no significant difference for the entire run.

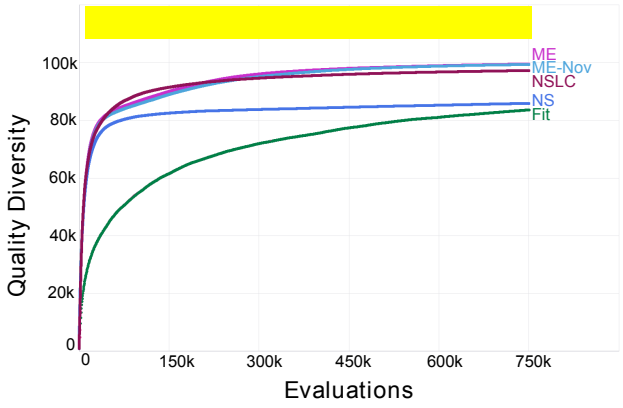


Figure 6.7: **DirectionBC (low alignment).** The QD score over time for each method×DirectionBC combination is depicted, averaged over 20 runs. With an almost complete lack of fitness-alignment in this BC, ME and ME-Nov tie for first place, and NS trails far behind.

Visualizing Behavior Spaces

Recall that for all variant search algorithms, the highest-performing genomes are collected in a discretized behavior space "grid" (inspired by the MAP-Elites grid). Collecting elite behaviors in

this way enables a convenient visualization of each algorithm's ability to collect QD within the domain. The visualization technique, proposed by Cully et al. [23], uses two dimensions to display an entire behavior space grid by nesting two-dimensional grids within other two-dimensional grids, each two-dimensional grid representing two dimensions of the behavior space. As an example, figures 6.8, 6.9, and 6.10 depict the final state of the QD collection grid at the end of one typical run of Fitness, ME, and NS in the FullTrajectoryBC. These visualizations reveal that: (1) Fitness (figure 6.8) and MAP-Elites (figure 6.9) exhibit a diminished ability to fill the grid for this BC (corresponding to a diminished pressure towards diversity) while the highest performing algorithm (on FullTrajectoryBC), NS (figure 6.10), is able to fill the grid almost entirely after $250{,}000$ evaluations. The visualizations also expose (2) the relatively strong alignment of the behavior space with the fitness measure through the smooth gradient in the outer two dimensions from low fitness at the top of the grid (near the start point) towards high fitness at the bottom-middle of the grid (near the goal point).

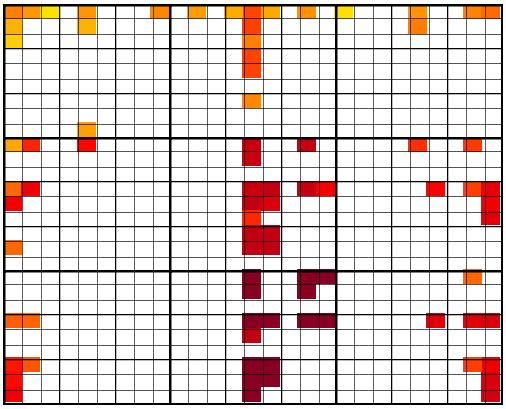


Figure 6.8: **Fitness in the FullTrajectoryBC.** In this example grid, the six-dimensional behavior space (FullTrajectoryBC) (discretized into three bins per dimension for a total of 729 bins) is visually depicted as a series of nested two-dimensional grids (each of which are 3×3). The color of each grid box corresponds to the quality of the solution found by the search algorithm after $250{,}000$ evaluations: yellow corresponds to low quality, dark red to high quality, and white to unfilled bins. Fitness finds very few of the possible behaviors for this BC.

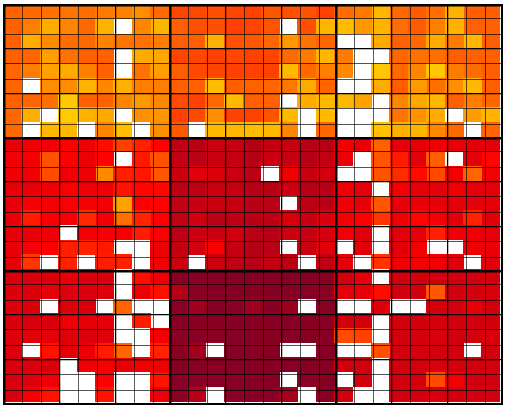


Figure 6.9: **MAP-Elites in the FullTrajectoryBC.** Compared to Fitness (figure 6.8), ME discovers far more QD under FullTrajectoryBC. Some bins remain unfilled (white), corresponding to the lack of pressure towards diversity within the ME algorithm.

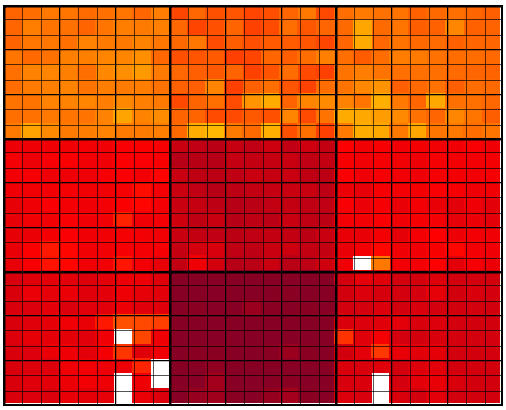


Figure 6.10: **Novelty Search in the FullTrajectoryBC.** Of the five compared algorithms, NS performs the best under the FullTrajectoryBC (featuring relatively high alignment between the BC and the objective) because it focuses exclusively on pursuing diversity. This conclusion is supported in the QD collection grid by almost all bins being filled (i.e. non-white).

These results are accompanied by a website where sample interactive behavior space visualizations (including the ability to browse through the discovered behaviors within each bin) are available for all 20 method$\times$BC combinations (including the three shown here):

`http://eplex.cs.ucf.edu/QD/GECCO-15/compare.html`

Discussion

The main insight to emerge from the experiment concerns the two key forces operating in QD algorithms: the push for novelty and the push for quality. If the BC, which supplies the main push for novelty, is sufficiently aligned with a notion of quality, then novelty-based approaches are most effectively empowered. In contrast, if the BC is orthogonal to a notion of quality, a strong push for quality (as in the elitism of ME) becomes the more instrumental factor in QD performance. The apparent ability under some BCs for NS *alone* to succeed, even without any explicit push for quality, and ME *alone* in other BCs to succeed without any explicit push for novelty, highlights just how profoundly this apparent principle applies. Yet it also appears that augmenting these most simple methods with a component to cover what they lack, e.g. augmenting NS with LC and ME with novelty, carries generally little risk. In fact, in some cases it improves their results, especially when they are being applied in their respectively less ideal alignment circumstances, hinting that aiming for "best of both worlds" is not unrealistic and may be important for the design of future QD algorithms yet unrealized (such as some of those introduced in the next chapter).

However, the results in this initial study should be qualified by the observation that the QD-Maze domain is relatively easy; all methods except for fitness eventually find many different ways of solving the maze (and in fact achieve nearly maximum QD-scores in some cases). Because more ambitious applications of QD in the future are likely to be significantly more challenging (imagine e.g. mazes with longer and more deceptive solution paths), a natural question is whether the results here will extend to future experiments on such higher difficulty problems. This question is addressed in the next chapter.

While even in much harder domains all of the methods considered here (and indeed, even random search) are likely to eventually explore all possible behaviors as the length of runs approaches infinity, it is important to consider how *quickly* each method realizes the maximum QD-score.

The relatively simplicity of this maze facilitated running evolution long enough for all methods to eventually catch up to each other (however, significant differences remain even at the end of each run). On more complex domains, each evaluation may take longer to compute, limiting the the duration of each run in practicality. Thus it is important to also observe the stratification of methods at points earlier in each run to determine which methods are suitable when computation is more expensive.

In all cases except for the least-aligned DirectionBC, the ranking between methods remains the same for the entire duration. On DirectionBC however, NSLC noticeably pulls ahead early on while only falling behind the MAP-Elites variants on an extended timescale. This early advantage may be due in part to NSLC expanding more quickly across the behavior space with an explicit diversity pressure that MAP-Elites lacks. However, even selecting for novelty in the ME-NOV variant does not mitigate the disadvantage that MAP-Elites suffers early on. Instead, it is possible that because of its strict elitism, MAP-Elites takes longer to escape the draw of the deceptive trap and achieve a maximum score at nearly all points in the behavior space. If true, on harder problems the strict elitism of MAP-Elites may cause a further breakdown in performance versus methods with a more forgiving selective pressure. This possibility is the primary motivation for the ME-PGD variant explored in the next chapter, where each bin in the ME grid instead contains *two* slots where the second slot is not governed by elitism.

There are some other considerations outside QD performance alone that may arise in specific domains. For example, because it must divide the behavior space into discrete bins, ME (and ME-Nov) become increasingly difficult to apply to higher-dimensional BCs as the number of bins increases exponentially as a function of BC dimensionality – an issue not present in NS or NSLC. Yet beyond its use as a search method, the ME grid also turns out to be an excellent mechanism for collecting QD results, enabling QD to be easily visualized and quantified even for methods that are not ME. Thus even users of NSLC or other QD approaches may find it advantageous to

limit BC dimensionality to enable the convenient collection of QD in an ME grid running in the background.

These experiments reveal that the alignment of the BC with the objective is a critical factor in the performance of QD algorithms. While diversity here is easily achieved under the unaligned DirectionBC, it takes these runs significantly longer to achieve the *quality* latent in the behavior space. In general, as alignment progressively decreases from one BC to the next, the maximum QD score is more slowly realized (and on the least aligned BC some treatments fail to reach the maximum score at all). Given that this is true even on the relatively easy QD-maze task, the question arises whether unaligned BCs will be capable of governing search on more difficult problems.

Conclusions

The aim of this study was to help to unify the emerging *quality diversity* (QD) research area by beginning to introduce benchmarks and principles for analyzing and contrasting methods in the area. For this purpose a new domain with many parallel solutions called the QD-Maze was introduced, and several QD methods (including a new variant called MAP-Elites+Novelty) were compared in the QD-maze under different behavior characterizations with varying degrees of alignment. The results begin to establish initial principles for understanding QD, namely that the alignment of the behavior characterization with the notion of quality significantly influences which QD methods will be most effective in a particular setup. This insight gives hope that an even broader understanding of QD can be reached in the future, thereby helping to expand the reach and impact of evolutionary computation into this burgeoning field where its power for *diversification* is appreciated just as much (or more) than its power for optimization.

CHAPTER 7: EXPERIMENT: UNALIGNED QUALITY DIVERSITY ON DIFFICULT PROBLEMS

The experiments in this chapter appeared in part in Pugh et al. [91] and Pugh et al. [93] and in full in Pugh et al. [92].

The following study continues the effort initiated in the previous chapter to establish a standard framework for understanding and comparing different approaches to QD. To that end, this chapter adds two additional maze-navigation benchmark tasks of substantially greater difficulty, designed to challenge even the best QD algorithms to the point that some may fail to find the highest-performing solutions altogether.

As we learned in the previous chapter, the degree of the BC's alignment with the objective has a strong influence on the performance of QD algorithms (specifically, their ability to find high quality solutions). The hypothesis is that searching for novelty in highly-aligned BCs assists fitness pressure in achieving good solutions by helping to overcome deception (as initially revealed in Lehman and Stanley [59] and Lehman and Stanley [61]); however, the fitness-assisting effect is less present in *unaligned* BCs. This hypothesis is challenged in this chapter by introducing mazes with more deception than any previous study.

The apparent weakness of unaligned BCs presents a potential problem for researchers interested in finding QD with respect to such a non-optimal characterization because standard practice suggests driving search with the same notion of diversity that you are ultimately interested in discovering [22, 62, 78, 122, 126, 127]. In fact, a review of the literature reveals that finding unaligned QD is often (if not usually) the goal of QD practitioners (perhaps because aligned QD is less interesting in many applications).

Reflecting its widespread presence in the literature, this chapter addresses only the problem of *finding QD with respect to an unaligned BC* (specifically, *DirectionBC* from Chapter 6). While experiments reveal that conventional QD algorithms indeed suffer under the guidance of an unaligned BC, a solution is presented here in the form of new algorithms that drive search with multiple characterizations *simultaneously* (thus enabling the discovery of unaligned QD without losing the fitness-assisting qualities of an aligned BC) and then empirically validated on tasks specifically designed to be challenging enough to cause search driven solely by unaligned BCs to break down entirely, thus establishing a definitive stratifications between treatments. These new multi-characterization approaches, together with an increased understanding of the types of characterizations that are ideal for driving search effectively, enable the application of QD algorithms to a wider variety of more difficult domains in the future.

Maze-navigation Domain

Successfully navigating a maze can be challenging because it requires learning a complex mapping between sensors and effectors. In part because sources of deception (and thus problem difficulty) are often visually apparent (e.g. dead ends that lead closer to the goal without reaching it), maze navigation has become a canonical domain for evolutionary robotics experiments (e.g. Lehman and Stanley 61, Velez and Clune 129). Another benefit of maze domains is that they tend to be computationally inexpensive, allowing many more evaluations than more intensive physics-based simulations. For this reason, maze domains are ideal for studying evolutionary dynamics over long timescales – an endeavor that would be impossible in a more computationally expensive domain.

In the study presented here, three mazes of varying difficulty assess the efficacy of QD algorithms as problem difficulty increases. In an important departure from mazes designed as an optimization problem (e.g. the HardMaze domain in Lehman and Stanley 61), where there is often only a single

"correct" path, the mazes here are intentionally designed with *multiple* viable paths to reach the goal. This unusual maze design makes it possible to investigate evolution's potential for finding QD. Thus the task in this study is not simply to find an agent that reaches the goal, but to find all of the different ways of driving through the maze (including those that may not necessarily reach the goal at all).

An important feature of all three mazes is the *deceptive trap* – areas of the maze that appear to lead closer to the goal (in terms of Euclidean distance) but, because of the presence of obstructions, do not actually offer a shorter drivable path to the goal. These traps, often in the form of an easily-accessible corridor that lead in the direction of the goal point but ultimately terminate at a dead end, represent local optima in the search space and serve to deceive algorithms that simply follow a gradient of increasing fitness [61]. Thus, mazes containing such traps serve as a metaphor for complex domains in general, where successfully solving the problem requires a clever search algorithm aimed at innovation rather than straightforward optimization.

The first and least difficult maze was introduced in Chapter 6 as the "QD-maze" but is referred to here as the **small maze** (Figure 7.1) to signify its contrast with the other larger and more difficult mazes presented next. The small maze provides an initial example of this multi-path maze design. In this maze, individuals must escape from a single deceptive trap surrounding the start point after which the maze opens up, allowing a wide variety of possible strategies for reaching the goal. Importantly, agents are given a considerable amount of time in this maze to allow for the expression of complicated and roundabout strategies (such as crossing back and forth across the map several times following different routes before ultimately driving to the goal). The ideal QD algorithm would eventually find *all* strategies for driving around the maze, including those that do not end up reaching the goal.

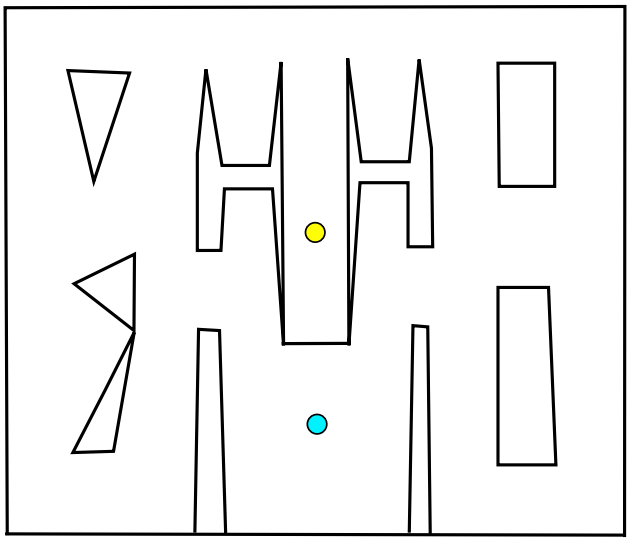


Figure 7.1: **Small maze.** Individuals start at the yellow circle (top) and must navigate to the goal point, marked with a blue circle (bottom). The relative openness and lenient time constraint allow a range of different techniques for reaching the goal.

Gauntlet Mazes

Of course, the most interesting real-world problems are often complex and challenging, and if maze navigation domains are to serve as proxies for such problems then they too should display nontrivial complexity. For this reason, two additional *gauntlet* mazes are introduced. While each maintains the important design principle of multiple viable paths to the goal, these mazes are intentionally made more difficult by (1) adding several larger and more pronounced deceptive traps, (2) setting strict maximum evaluation times such that individuals cannot wander all the way down a major deceptive trap and subsequently return with enough time to continue on to reach the goal, and (3) placing the goal point at the end of a series of chained sub-mazes that are each approximately as hard as the small maze and the original HardMaze from Lehman and Stanley [61] (judged visually by their size and number and severity of deceptive traps). Navigating to the end of the **asymmetric gauntlet** (Figure 7.2) and **symmetric gauntlet** (Figure 7.3) thus requires

evolving complicated behavioral strategies and overcoming significant levels of deception. In this way, these mazes test both the ability to overcome multiple successive deceptive traps *and* to cover the space of possible solutions in the same run.

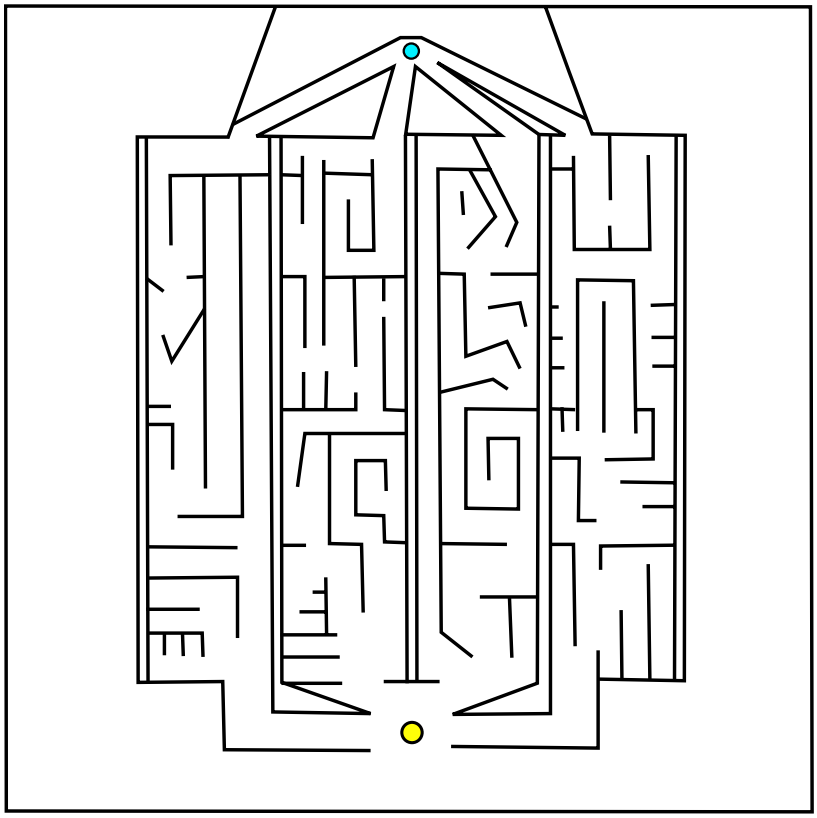


Figure 7.2: **Asymmetric gauntlet.** Individuals start at the bottom point and must navigate to the goal at the top of the maze. Because of the variation between maze legs, reaching the goal is easier by some routes than others. The presence of multiple paths through the maze increases the variety of potential driving strategies that can reach the goal.

Maze Parameters

Parameter settings for each maze domain are presented in table 7.1, which describes differences in maze size, time constraints, and agent sensor radius. Note that the high maximum evaluation time

in the small maze relative to its size is one factor that makes this domain considerably easier than the two gauntlet mazes.

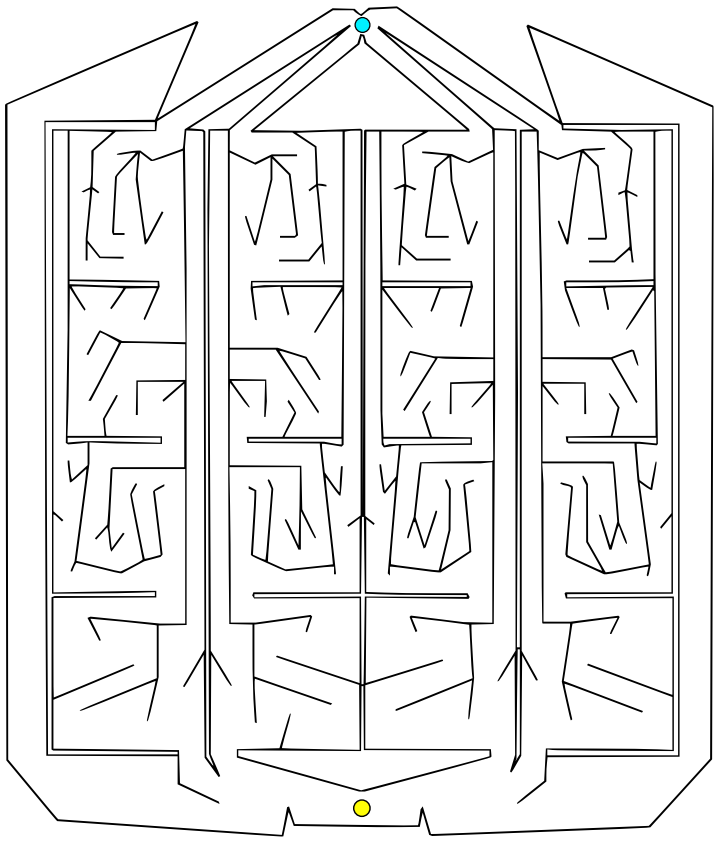


Figure 7.3: **Symmetric gauntlet.** Individuals start at the point at the bottom and must navigate to the goal at the top of the maze. Each leg of the maze is an approximate mirror image of the neighboring legs (thus all paths through the maze are similarly difficult to achieve). The presence of multiple legs allows a variety of different driving strategies to be successful.

Table 7.1: **Maze-specific parameters.** Reflecting differences in design and difficulty, some parameters vary between mazes.

	Small maze	**Asymmetrical gauntlet**	**Symmetrical gauntlet**
Evaluation time (ticks)	800	1300	2700
Maze dimensions (units)	900×770	6000×6000	4600×5280
Rangefinder length (units)	100	200	200
Velocity (units/tick)	6	6	6
Max turn rate (rads/tick)	0.21	0.21	0.21

Algorithms

In the following experiments, several variant QD algorithms are considered, including all of those featured in the study in Chapter 6: **Fitness**, novelty search (**NS**), novelty search with local competition (**NSLC**), MAP-Elites (**ME**), and MAP-Elites + Novelty (**MENOV**). Additionally, a second augmentation of standard MAP-Elites is introduced called *MAP-Elites + Passive Genetic Diversity* (**MEPGD**). All of these algorithms including the ones to follow are described in detail in Chapter 5, Section 5.

MAP-Elites + Passive Genetic Diversity (MEPGD)

MAP-Elites + Passive Genetic Diversity is a simple augmentation introduced for the first time here to help mitigate the potential for the strict elitism of standard MAP-Elites to cause search to stagnate indefinitely when the entire population is stuck in deceptive traps. In MEPGD, each bin contains two slots instead of one and replacement for each is governed by different rules. In the first slot, individuals are replaced based on the standard elitism rule. However, the second slot is managed by random replacement: if this slot is already filled, each newly-encountered individual mapped to the bin has a 30% chance of of being saved (and the current resident evicted). The

second slot therefore allows exploration to continue even if the first slot is deceived by a dead end.

Multi-BC QD Algorithms

As revealed in Chapter 6, unaligned BCs may negatively impact the performance of QD algorithms. Specifically, QD algorithms depend on a BC aligned with the objective to help avoid deception and thus lead the search to find higher quality solutions than would be found with fitness pressure alone. When this alignment is not present, QD may become susceptible to being stuck in local optima. While this difficulty was eventually overcome in the small maze, the more challenging gauntlet mazes may render algorithms driven by an unaligned BC unable to find good solutions. Because QD practitioners are often (if not usually) interested in finding unaligned diversity, an approach is needed that harnesses the deception-evading power of aligned BCs without losing the ability to achieve the desired unaligned diversity.

This dissertation offers a promising answer in the form of a new approach to QD: *driving search with multiple BCs simultaneously*, thereby allowing the application of both an aligned and unaligned BC at the same time. The following algorithms represent possible options for realizing this approach.

[NS-NS] Multi-BC Novelty Search

In *multi-BC novelty search* (**NS-NS**), novelty search is extended to support multiple BCs by calculating a separate novelty score for each BC and then combining these scores in a multi-objective formulation (as in NSGA-II of Deb et al. [30]) where individuals are ranked in a series of Pareto fronts. Each BC maintains its own independent archive against which individuals are evaluated to determine their novelty score for that BC. There is only a single population where the breeding

potential for each member is decided by a Pareto ranking according to the various novelty scores. Although NS is not itself a QD algorithm because it lacks a mechanism for discovered behaviors to increase in quality, when extended to include multiple BCs, NS-NS can conceivably achieve some success if one BC serves to find diversity while another promotes increasing quality (e.g. an unaligned BC paired with an aligned BC). Note that even such a pairing does not quite embody the spirit of QD because any tendency towards increasing quality that emerges from an aligned BC is not explicitly *local*.

[NS-NSLC] Multi-BC Novelty Search with Local Competition

The idea of NS-NS can then be expanded to include a drive towards locally-increasing quality by adding a LC objective (in the same way as NSLC) where behavioral neighbors are decided by the unaligned BC that expresses the notion of diversity that the user is ultimately interested in collecting. The resulting algorithm, called *multi-BC novelty search with local competition* (**NS-NSLC**), therefore includes three distinct objectives (combined via Pareto-ranking): (1) a quality-aligned novelty score to facilitate overcoming deception, (2) an unaligned novelty score for discovering new behaviors of interest, (3) an unaligned LC score to promote competition within niches of similar behaviors.

[ME-ME] Multi-BC MAP-Elites

In *multi-BC MAP-Elites* (**ME-ME**), the characteristic simplicity of MAP-Elites is preserved by simply maintaining a separate grid for each BC (with similar maximum sizes). On each iteration of ME-ME, an equal amount of parents are selected from each grid, and their resulting offspring are mapped to *both* grids (where the decisions to save or discard them are performed independently; e.g. between two grids, a single offspring may be saved in one, both, or neither).

[MENOV-MENOV] Multi-BC MAP-Elites + Novelty

Finally, MENOV is extended to *multi-BC MAP-Elites + novelty* (**MENOV-MENOV**) similarly to ME-ME except where each grid also maintains its own novelty archive and selection within each grid is proportional to novelty.

Experiment

The behavior characterization (BC) determines the form of pressure that ultimately drives the diversity component of the search and thus must be selected carefully to complement the evolutionary algorithm. The experiments here explore the two BCs introduced in Chapter 6 that are respectively most-aligned and least-aligned with the notion of quality (how close an agent is to arriving at the goal in terms of Euclidean distance): **EndpointBC**, which is simply a two-dimensional vector containing the x and y coordinates of an individual's location at the end of its trial, and **DirectionBC**, a five-dimensional vector with entries indicating whether the individual was most frequently facing north (0.125), east (0.375), south (0.625), or west (0.875) during each fifth of its evaluation time. EndpointBC is highly aligned with the objective of navigating to the maze goal point because the continual discovery of new endpoints will eventually lead to finding the goal. On the other hand, DirectionBC is largely orthogonal to quality because the direction a robot faces at a particular time step does not necessarily determine whether or not it solves the maze (more importantly: it is possible to visit all of the behaviors in the DirectionBC space without ever reaching the goal).

Consistent with the observation that the goal of QD applications in practice is often to find diversity with respect to an unaligned BC, the assumption in this study is that the goal of the user is always to find QD with respect to DirectionBC. Therefore, exploring how different approaches to QD interact with each of EndpointBC and DirectionBC makes it possible to address several important

questions:

1. How well does the approach suggested by current literature (i.e. driving search with the very notion of diversity that you are interested in collecting) work? Given that experiments in Chapter 6 suggest that DirectionBC is suboptimal for achieving the highest quality solutions even in the relatively simple small maze, the hypothesis is that this conventional approach will further break down on more complex and challenging domains such as the gauntlet mazes.

2. Can diversity with respect to DirectionBC be found without searching for it explicitly? That is, EndpointBC has been shown to be effective for driving novelty search to find solutions to deceptive mazes [59]. However, it is unknown whether such an approach can be similarly effective at finding QD when diversity is measured on a separate unaligned metric (i.e. unaligned diversity is only collected passively). The hypothesis is that algorithms driven by EndpointBC should do well in terms of progress towards the goal, but may not necessarily result in high diversity with respect to DirectionBC.

3. Finally, can multiple BCs successfully be *combined* to compensate for the shortcomings of highly aligned BCs (lower diversity) and unaligned BCs (lower performance)?

Each of the algorithms from Section 7 is implemented with some combination of DirectionBC and EndpointBC for a total of fifteen treatments, enumerated in table 7.2. Each treatment is run 20 times on the small maze, symmetric gauntlet, and asymmetric gauntlet for a total of 900 runs (300 per maze). Small maze runs ended after 250,000 evaluations and gauntlet runs ended after 1,000,000 evaluations (in each case more than enough time for all algorithms to reach a performance plateau). Networks are evolved with a modified version of SharpNEAT 1.0 [46] with the same mutation parameters validated in Chapter 6: 60% mutate connection, 10% add connection,

and 0.5% add neuron. Networks are feedforward only and restricted to asexual reproduction (no crossover); other settings follow SharpNEAT 1.0 defaults.

Table 7.2: **Fifteen treatments compared on three mazes.** Algorithms in different columns differ by the BC that drives search. Conventional QD algorithms are tested with each of DirectionBC and EndpointBC driving search (denoted by the subscripts d and e, respectively), while a new class of multi-BC QD algorithms drives search with both DirectionBC and EndpointBC simultaneously.

No BC	**DirectionBC**	**EndpointBC**	**Multi-BC**
Fitness	NS_d	NS_e	NS_eNS_d
	$NSLC_d$	$NSLC_e$	NS_eNSLC_d
	ME_d	ME_e	ME_eME_d
	$MENOV_d$	$MENOV_e$	$MENOV_eMENOV_d$
	$MEPGD_d$	$MEPGD_e$	

Metric: QD-score

In traditional maze navigation domains, an appropriate metric would be whether or not a robot was eventually able to navigate to the goal. However, this metric does not speak to an algorithm's propensity for discovering diversity in a search space. For this reason, Chapter 6 introduced a new QD metric (which is similar to the "global reliability" metric in Mouret and Clune [78]) that reflects both the quality and diversity of individuals found by evolution (including solutions and non-solutions). Diversity in this metric, called the *QD-score*, is measured with respect to a BC. In the experiments here, DirectionBC always characterizes diversity for the purpose of computing the QD-score *regardless of the behavior characterization driving search*. This approach reflects the usual idea that the desire is to see a wide diversity of solutions at the end of a run with respect to an arbitrary BC that is not necessarily aligned with solving the problem (and often is not).

To quantify how much of the space is explored by an algorithm for the QD-score, the entire be-

havior space is first discretized into a collection of t bins[1] $\{N_1, \ldots, N_t\}$ as in the MAP-Elites algorithm described previously. Each bin corresponds to a unique combination of features from the individual's BC (in this case, DirectionBC) and represents a niche in the behavior space. Diversity is then quantified as the number of bins filled over the course of an evolutionary run. By *summing the highest fitness values found in each grid bin*, where Q_i represents the highest fitness achieved in bin N_i, it becomes possible to simultaneously quantify both quality and diversity as

$$\text{QD-score} = \sum_{i=1}^{t} Q_i \ .$$

This DirectionBC-based QD-score[2] is the primary metric in all mazes. Note that for the purpose of calculating QD-score, fitness is defined in a way that reflects the shortest *drivable* path between an agent's ending location and the goal, respecting that agents cannot drive through walls. This special QD-score fitness is calculated by a breadth-first flood fill from the goal point, assigning fitness to locations in the maze that decreases linearly at each layer of the flood fill. Importantly, this fitness value, which draws a perfect non-deceptive gradient over the maze, is not available to any algorithms to drive search (which are instead driven by the naive Euclidean distance fitness measure) but merely appears during post-hoc analysis to give an accurate accounting for how close each collected behavior is to solving the maze.

Metric: Total Maze Progress

Due to the overwhelming historical focus on optimization and the recent realization that behavioral diversity can itself be a powerful tool for optimization [59, 79, 80], this study includes an additional

[1] In this study, $t = 1024$ when discretizing the DirectionBC behavior space.

[2] The original small maze results from Chapter 6 instead always measure diversity with respect to the BC that drives search and thus do not explore the idea of driving search with BCs other than those which characterize the dimensions of interest.

performance metric for the two gauntlets[3] that captures the spirit of this more conventional search paradigm: the *total maze progress* metric measures how close a run is to solving all four legs of the maze. More precisely, total maze progress is the sum of four progress measures (one per leg) where progress for each leg increases linearly as endpoints are discovered further along their solution path (calculated by means of the flood fill distance to the goal point mentioned previously). If E_i is the set of all flood fill fitness scores associated with endpoints discovered inside leg i then total maze progress can be quantified as

$$\text{total maze progress} = \sum_{i=1}^{4} \max E_i \ .$$

A maximum score is achieved by solving all four legs. Total maze progress therefore addresses the important question of how well-suited QD algorithms are to the task of optimizing towards a series of predefined targets[4].

Results

In all of the figures presented in this section, treatments are color-coded according to which BC drives search: DirectionBC (subscript d) is drawn in blue, EndpointBC (subscript e) in yellow, multi-BC in green, and Fitness (which is not driven by any BC) in gray. For each treatment, results represent an average over 20 runs; error bars represent the standard error of the mean and can be interpreted to infer which differences are statistically significant when p values are not explicitly provided. In all reported cases, statistical significance is determined by an unpaired two-tailed

[3]No such metric is defined for the small maze because all treatments consistently and quickly find solutions and thus it does not represent a challenging optimization problem.

[4]Total maze progress is related to the interests of multimodal optimization, where the goal is often to find all of the global optima in a fitness landscape without regard to the behaviors or phenotypes that get there.

Student's t-test[5].

Total maze progress

Figures 7.4 and 7.5 depict the total maze progress achieved after all evaluations in the asymmetric gauntlet and symmetric gauntlet respectively. A maximum possible score of 400 corresponds to solving all four legs in the same run, although such scores are not observed in practice. Unlike in the small maze, where all treatments consistently find solutions in every run, many gauntlet runs (particularly those of Fitness or DirectionBC-driven treatments) do not find any solutions at all, reflecting the increased difficulty in the gauntlet mazes. Of the two gauntlet mazes surveyed by this metric, higher scores are obtained by all treatments on the asymmetric gauntlet, indicating that the asymmetric gauntlet is comparatively easier to solve, thus establishing a continuum of difficulty between the three maze domains featured in this study: small maze (easiest), asymmetric gauntlet (harder), symmetric gauntlet (hardest).

While in both gauntlets the performance of Fitness is sub-par as expected (it is known to struggle with deception in maze domains; Lehman and Stanley [59]), a perhaps more surprising result is that all DirectionBC-driven treatments perform significantly worse than Fitness in terms of their ability to find solutions ($p < 0.05$) to both the asymmetric gauntlet and the symmetric gauntlet (figures 7.4 and 7.5). Indeed, of these treatments on the symmetric gauntlet, only MEPGD$_d$ finds any solutions at all (two solutions are found across all 20 runs).

On the other hand, all approaches that include an aligned BC (EndpointBC) perform significantly better than Fitness ($p < 0.001$). Of those treatments that include EndpointBC, single-BC ap-

[5]The simple Student's t-test is chosen intentionally to avoid Type II errors, which are more likely when adjusting for multiple comparisons. As such, the results here are intended to highlight potential differences between treatments, not to establish a definitive ranking.

proaches tend to perform better (with respect to solving the maze) than multi-BC approaches that also include DirectionBC (figures 7.4 and 7.5). This phenomenon is especially apparent on the harder symmetric gauntlet, where NSLC$_e$ and NS$_e$ perform significantly better than the multi-BC variants ($p < 0.05$). Of particular interest is that specialized QD algorithms such as NSLC$_e$ are competitive with the currently-accepted method for overcoming deception on maze tasks: NS$_e$ [59, 61]. On the symmetric gauntlet there is some evidence that NSLC$_e$ may actually be *better* than NS$_e$, although the evidence is not strong enough to establish statistical significance ($p = 0.093$).

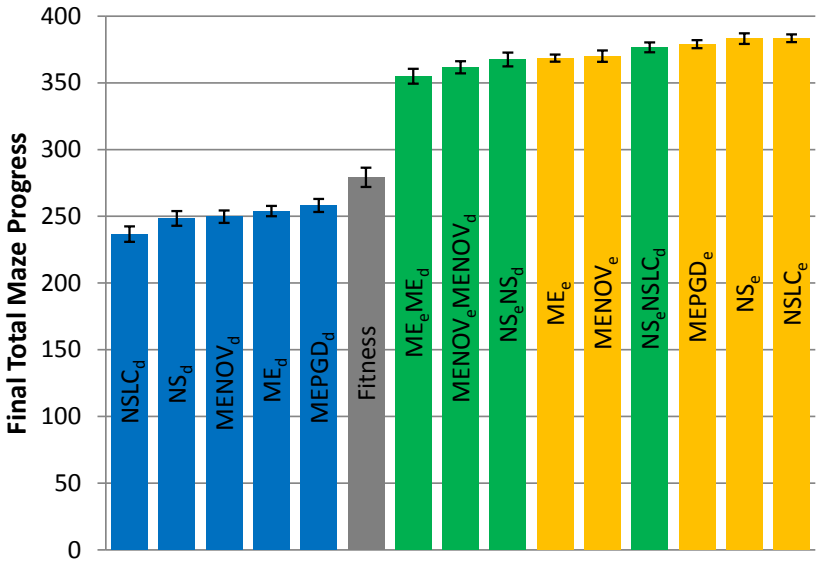


Figure 7.4: **Total maze progress (asymmetric gauntlet).** The final total maze progress achieved by each of fifteen treatments (table 7.2) after 1,000,000 evaluations on the asymmetric gauntlet is shown (averaged over 20 runs). Bars are color-coded according to which BC drives search and error bars represent standard error. A maximum possible score of 400 corresponds to solving all four legs of the maze.

Of the successful MAP-Elites variants (those driven by EndpointBC), MEPGD$_e$ performs significantly better than the core ME$_e$ on both gauntlets, while MENOV$_e$ in neither case is significantly different than ME$_e$ (figures 7.4 and 7.5).

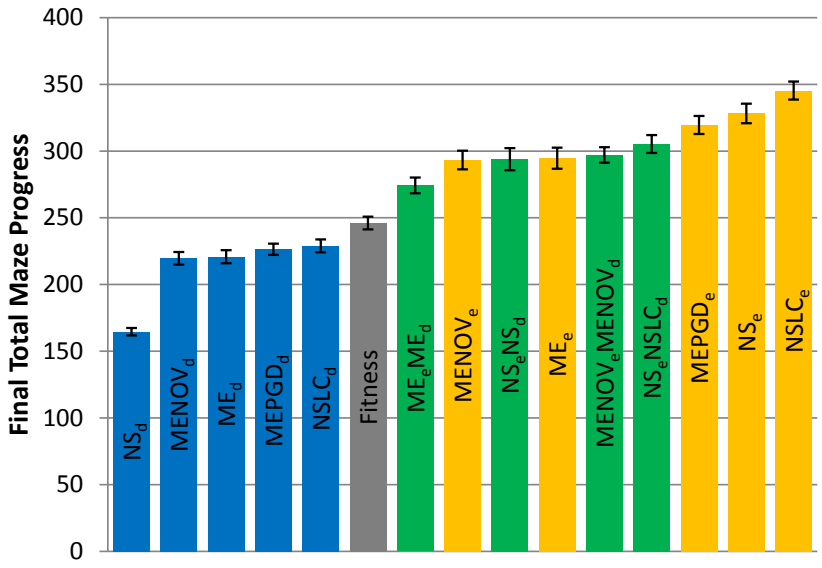


Figure 7.5: **Total maze progress (symmetric gauntlet).** The final total maze progress achieved by each of fifteen treatments (table 7.2) after 1,000,000 evaluations on the symmetric gauntlet is shown (averaged over 20 runs). Bars are color-coded according to which BC drives search and error bars represent standard error. A maximum possible score of 400 corresponds to solving all four legs of the maze.

QD-score

The final QD-score achieved by each treatment after all evaluations is depicted in figures 7.6, 7.7, and 7.8.

Small maze

Reflecting its lack of challenging complexity, the best treatments on the small maze consistently achieve near the maximum possible QD-score of 1,024,000, which corresponds to finding a maze solution (score $= 1000$) in each of the 1,024 bins (figure 7.6). On this relatively simple task, QD-score is dominated by the multi-BC QD approaches (ME$_e$ME$_d$, MENOV$_e$MENOV$_d$, NS$_e$NSLC$_d$)

and by $NSLC_d$. EndpointBC-driven approaches perform poorly in comparison, though not as poorly as Fitness and NS_d.

Of interest is the comparatively large variance on the MAP-Elites variants (when driven by DirectionBC) versus their NS-based counterparts (figure 7.6). In particular, the most extreme such variance, on ME_d, is caused by three outliers. While most runs of ME_d score between 950K and 1000K, the outlier runs obtain scores of 791K, 680K, and 421K. In each of these runs, the grid is completely filled (representing maximum diversity), but many bins contain low-quality behaviors (mostly representing agents that exclusively drive around inside the main deceptive trap). Thus the higher variance observed by ME_d here is indicative of MAP-Elites sometimes becoming stuck in a local optima.

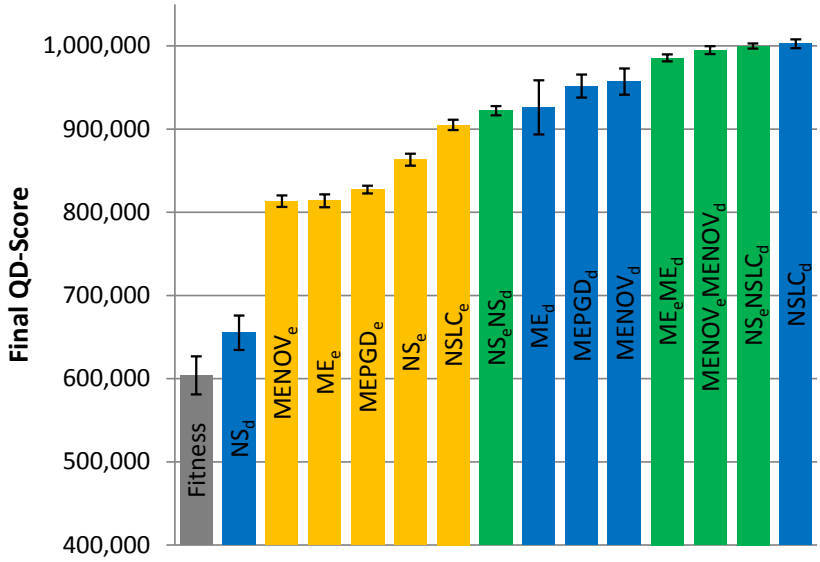


Figure 7.6: **Final QD-score (small maze).** The QD-score achieved by each of fifteen treatments (table 7.2) after 250,000 evaluations on the small maze is shown (averaged over 20 runs). Bars are color-coded according to which BC drives search and error bars represent standard error. In all cases, QD-score is measured with respect to DirectionBC. The maximum possible QD-score on the small maze is 1,024,000, corresponding to a perfect solution in all 1,024 bins.

Gauntlet mazes

Because of the strict time constraints on the gauntlet mazes, many types of behaviors (i.e. bins in the QD grid) can never represent full solutions even in theory (e.g. those that spend the majority of their time facing south while the goal point is located to the north); thus the maximum possible QD-score on the gauntlet mazes is lower than on the small maze and also difficult to achieve in practice. This limitation is reflected by the comparatively lower scores observed in both the asymmetric gauntlet and the symmetric gauntlet (figures 7.7 and 7.8).

While DirectionBC-driven QD achieves the highest scores on the small maze, this trend is reversed as domain difficulty increases (figure 7.6). On the asymmetric gauntlet, many treatments driven by EndpointBC perform similarly to those driven by DirectionBC (figure 7.7). On the comparatively more difficult symmetric gauntlet, the trend is completely reversed, with EndpointBC-driven approaches performing significantly better than those that are driven only by DirectionBC (figure 7.8).

In all three mazes, the QD-score metric is dominated by the best multi-BC treatments. Specifically, in all three mazes NS_eNSLC_d is consistently among the best performing treatments (figures 7.6, 7.7, and 7.8). On the asymmetric gauntlet, its lead is unmatched and on the symmetric gauntlet it is tied with NSLC_e for first place. The difference between NSLC_e and NS_eNSLC_d is not statistically significant ($p = 0.163$).

However, it turns out that even though their final QD-scores are similar on the symmetric gauntlet, NSLC_e and NS_eNSLC_d exhibit different learning curves. To highlight this difference on the symmetric gauntlet, it is instructive to graph the development of QD-score over the course of evolution. Figure 7.9 depicts the QD-score over time for the best-performing method from each of the three classes: EndpointBC, DirectionBC, and multi-BC (table 7.2). The general trend displayed by

each treatment is representative of the other treatments in each respective class: e.g. DirectionBC-driven treatments tend to increase quickly and then plateau at a low score. On the other hand, NSLC_e increases relatively slowly before ultimately reaching a much higher score. Combining the best of both of these options, NS_eNSLC_d quickly reaches high scores (figure 7.9). Thus, overall, the hybrid NS_eNSLC_d proves a competitive choice for maximizing QD-score on all the variant mazes. While in this experiment other methods were able to catch up to the QD-score achieved by NS_eNSLC_d, the rapid collection of available QD is important for domains where evaluations are computationally expensive, thus limiting the feasible duration of a single run.

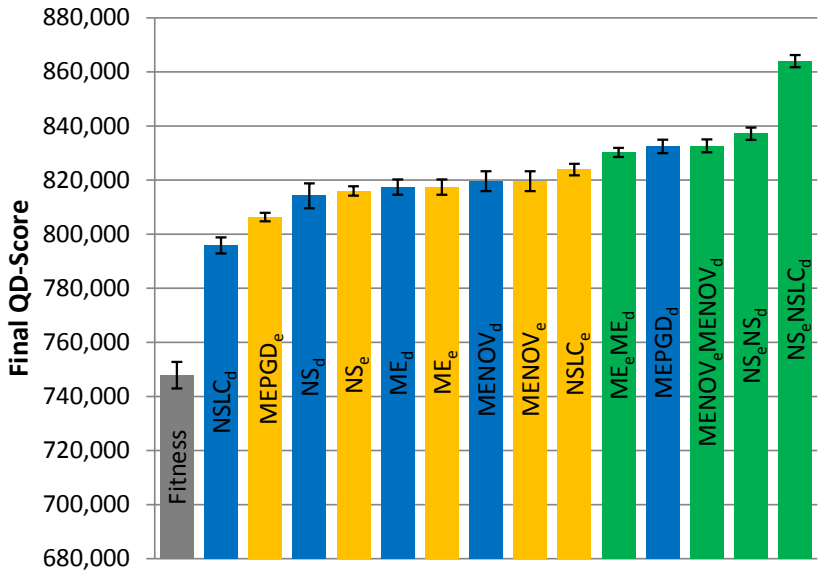


Figure 7.7: **Final QD-score (asymmetric gauntlet).** The QD-score achieved by each of fifteen treatments (table 7.2) after 1,000,000 evaluations on the asymmetric gauntlet is shown (averaged over 20 runs). Bars are color-coded according to which BC drives search and error bars represent standard error. In all cases, QD-score is measured with respect to DirectionBC.

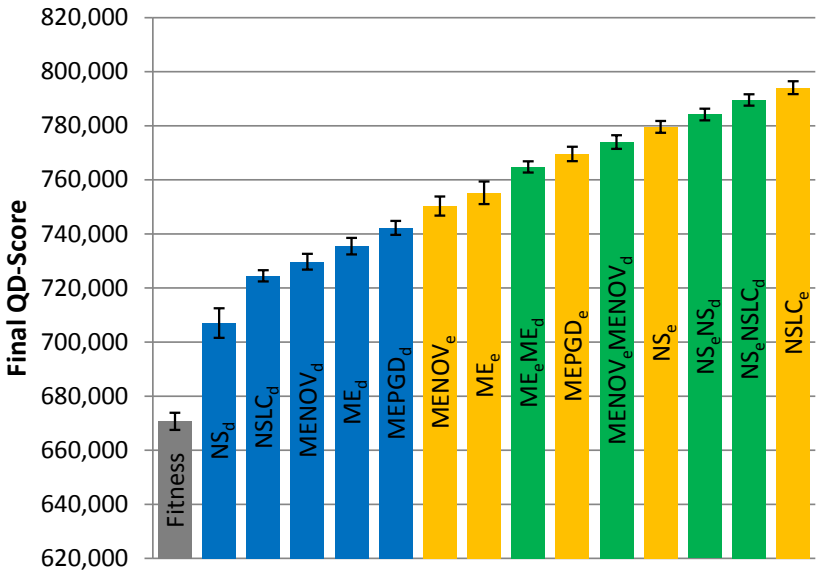


Figure 7.8: **Final QD-score (symmetric gauntlet).** The QD-score achieved by each of fifteen treatments (table 7.2) after 1,000,000 evaluations on the symmetric gauntlet is shown (averaged over 20 runs). Bars are color-coded according to which BC drives search and error bars represent standard error. In all cases, QD-score is measured with respect to DirectionBC.

Discussion

Within the wide-ranging field of EC, two distinct types of research goals are relevant in the context of this investigation. The first and historically most dominant focus of EC is the task of optimization: harnessing the powerful natural mechanism of "survival of the fittest" to reach some predefined target (or series of targets). It is this goal that lies at the heart of the vast majority of EC literature, thereby seemingly aligning the ambitions of EC with the broader practice of machine learning where optimization is treated as the essence of learning itself. However, having only recently begun to garner attention, the second type of research goal is less familiar, though it offers promising new opportunities for discovery and advancement that are uniquely accessible to EC. This goal, called *quality diversity*, represents a fundamental departure from optimization because

instead the idea is to explore all of what is possible rather than to find only the best option. While the primary intention of this dissertation is to promote the theory and practice of QD, the results of this study have implications relevant to both paradigms.

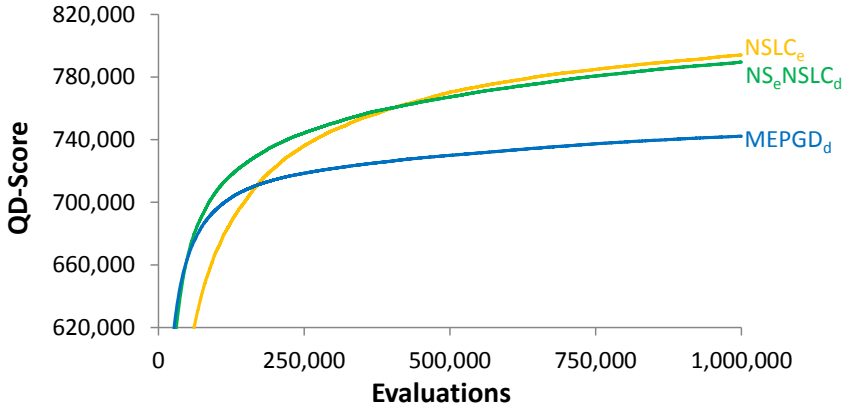


Figure 7.9: **QD-score over time (symmetric gauntlet).** The progression of the QD-score on the symmetric gauntlet for the best-performing treatment from each class (according to the BC that drives search; DirectionBC: MEPGD$_d$, EndpointBC: NSLC$_e$, multi-BC: NS$_e$NSLC$_d$) is graphed over time (averaged over 20 runs). The main result is that multi-BC treatments exhibit the best characteristics of each component BC: both increasing quickly and reaching high scores.

The recent realization that pursuing behavioral diversity can help to overcome deception [59, 61, 79, 80] has offered a source of hope to solving the problem of deception that permeates almost every optimization space of interest. However, deception does not exclusively affect optimization and not all notions of "behavioral diversity" are equally capable of thwarting it. As this study reveals, simply focusing on QD does not make evolution immune to the problem of deception because the quality-seeking *component* of QD algorithms can itself fall victim to it. In this study, which features maze tasks of varying difficulty, QD algorithms following the compass of an unaligned BC are incrementally less able to find QD as the level of deception increases from one maze to the next (figures 7.6, 7.7, and 7.8). Further highlighting that the problem of deception exists even in QD, surprisingly, as maze difficulty increases, it actually becomes more effective to drive search with an aligned BC even when you are collecting unaligned QD (figure 7.8). The primary lesson

is that searching for diversity with respect to an unaligned BC does not circumvent the problem of deception. Instead, diversity-seeking is mostly orthogonal to overcoming deception. Similar to the shortcomings of pursuing *genetic* diversity, it is possible to find a great diversity of behaviors with respect to an unaligned BC that do not actually differ in terms of fitness and all of which may exist within a deceptive local optima. However, for QD practitioners, the magic bullet cannot be to simply drive search with some other, better-suited BC because that only leaves the diversity of interest to be collected *coincidentally* (such as in the EndpointBC-driven approaches here). Instead, this study offers the promising new idea of driving search with multiple BCs simultaneously. So-called multi-BC algorithms (such as NS-NSLC) allow search to be driven both by the desired notion of diversity and a separate BC that is well-equipped to circumvent deception, thus unlocking the best parts of the search space for discovery.

An alternative approach to multi-BC algorithms not tested here is to simply concatenate both an aligned and unaligned BC into a single BC with more dimensions. In considering this option, it is important to remember that the behavior space grows *exponentially* with the size of the BC. Thus, such an approach may prevent the application of MAP-Elites because the number of bins would also grow exponentially until the grid no longer resembles a reasonably-sized population. With especially large BCs, it may be problematic to fit the entire ME-grid in memory and if the grid is vast enough that new individuals never collide with existing bins, the algorithm essentially devolves into random search. While concatenating multiple BCs into a single monolithic BC is still tractable with NS-based algorithms, the resulting vast behavior space may present an additional challenge over the multi-BC approaches tested here.

Following the lesson originally presented in Lehman and Stanley [59], this study reconfirms that a powerful strategy for finding solutions to a difficult maze is to abandon the objective entirely and simply search for behavioral diversity (e.g. NS_e in figures 7.4 and 7.5). This conclusion itself has implications outside of maze solving that apply more generally to all of optimization.

However, an important observation is that not just any type of behavioral diversity is successful. Indeed, with regard to being able to solve difficult mazes, searching for diversity with respect to an unaligned BC (such as DirectionBC) does even worse than purely objective search in all cases (figure 7.5). The more general lesson is that *BCs that are aligned with the notion of quality* (such as EndpointBC in this study) are the key ingredient to overcoming deception on difficult problems. Furthermore, this study offers the additional insight that QD algorithms themselves offer a means for "the objective-less search for behavioral diversity" (i.e. novelty search) to be rectified with their missing objective in a way that allows search to respect the ultimate goal (e.g. solving the maze) without re-introducing the problem of deception. Specifically, NSLC$_e$ demonstrates that it can optimize in the presence of strong deception at least as well as regular novelty search (figure 7.4). In fact NSLC$_e$ might even be better than novelty search as demonstrated in the symmetric gauntlet (figure 7.5). This result suggests that while the ultimate goals of QD are distinct from optimization, advancements in QD can themselves directly benefit optimization.

As a relatively new approach to QD, MAP-Elites remains a largely unexplored paradigm at the time of this writing. The core MAP-Elites algorithm offers the significant appeal of a very simple algorithm (requiring in effect only a few lines of code) that powerfully distills the essence of QD. Because it is relatively new, its performance in this study serves to confirm that it is largely impacted by BCs similarly to NS-based methods (suggesting some general principles for QD across different algorithms), but of course much room remains for MAP-Elites in particular to be improved. MENOV and MEPGD in this study both suggest that the core ME algorithm can be fruitfully augmented to help mitigate the restrictive effects of strict elitism. Some ideas not tested here include MEPGDNOV (combining genetic diversity with novelty) and MEPGD-MEPGD (ME with genetic diversity and multiple BCs). The genetic diversity component might also benefit from expansion to more than one diversity candidate per bin or by explicitly selecting for greater genetic distance from the elite candidate rather than only admitting genetic diversity passively.

An important question raised by the results is how to decide whether a particular domain is "hard" (therefore likely requiring more than one BC to enable finding the best performing solutions across the behavior space) and in such cases whether the chosen aligned BC is sufficiently aligned to overcome the threat of deception. One way to decide whether it may be necessary to include multiple BCs is simply to run a pure fitness-based (objective-driven) search. If such a search consistently finds solutions in multiple runs then the domain can be considered sufficiently easy that multiple BCs may be unnecessary and QD can simply be applied with the notion of diversity that is ultimately desired. On the other hand, if the domain proves too difficult for fitness, then it may be important to include a sufficiently aligned BC to aide search in overcoming deception. However, the question immediately arises: *how do you know whether a BC is aligned or unaligned with the objective?* One way to test for alignment is to try running simple NS only with the candidate BC. If the alignment is effective, such a search should at least do better than fitness-based search, which suggests that the candidate BC can complement an unaligned BC in a multi-BC hybrid.

Interestingly, while the results support that in sufficiently easy domains a more naive single-unaligned-BC approach may work to collect QD, it does not appear harmful to add an aligned BC in a multi-BC formulation. Thus even though it has been customary so far in the QD-related literature to rely on a single unaligned BC [22, 62, 78, 122], it may be possible to revisit some of the domains of the past with multiple BCs to achieve better performing behaviors that were previously undiscovered..

More broadly, the multi-BC approach and our new understanding of the implications of alignment can help us in the future to achieve more impressive results with QD than seen in the past. Even domains that might have seemed inexplicably out of reach might now become accessible through the application of multiple BCs. Thus because QD represents a promising direction exclusive to evolutionary techniques, it is in the interest not just of those working in QD, but of EC and

evolutionary robotics as a whole to seek out and propose such future domains. The potential for such ideas is foreshadowed by the wide range of applications already present in existing QD literature: morphology evolution [62, 78, 122], robot control and adaptation [22, 24, 78], image generation [84, 85], and three-dimensional object evolution [64]. With increasing interest in the field, these domains may be just the beginning for QD.

Conclusions

In an attempt to unify and investigate the emerging field of quality diversity (QD) and to better understand the factors that impact their performance, this chapter compared a variety of QD algorithm variants and controls in three different maze domains, two of a higher level of complexity than previously seen in such maze-based studies. By pushing maze difficulty beyond that seen before in QD before, the study was able to expose conditions under which QD effectively breaks down. It turns out that driving the diversity component of QD algorithms exclusively by a BC unaligned with quality (which is heretofore common practice) performs relatively poorly under such difficult conditions. However, as a solution to this apparent weakness of QD, new methods that combine more than one BC (one aligned and one unaligned) tend to perform well at the same time as effectively discovering the desired diversity of behaviors, suggesting a promising path forward for QD in the future as it is applied in increasingly ambitious domains.

CHAPTER 8: NEW RESEARCH PLATFORM: APPLYING QD TO THE VOXELBUILD OPEN-ENDED CREATIVE TASK

The experiments in this chapter were originally published in Pugh et al. [94].

This dissertation introduces the emerging field of QD as a promising new frontier for evolutionary computation, bringing with it a new class of applications where finding *many* solutions is just as important as finding *good* solutions. On such tasks, evolution's tendency for diversification is celebrated, setting it apart from other approaches within machine learning whose only goal is simple optimization. Most applications of QD (such as the mazes featured in previous chapters) operate in a closed behavior space where the expectation is that all regions will eventually be explored and optimized (e.g. the MAP-Elites algorithm declares this expectation explicitly by dividing the entire space into a finite number of bins with the intent of finding the best-performing individual in every bin).

In some respects, QD problems demand a level of *creativity* from the search algorithm, tasking it to produce (and subsequently refine) a procession of novel forms. However, in closed problems, creativity is effectively bounded by the size and scope of the behavior space. If QD algorithms can find a collection of high-performing individuals spanning an entire behavior space, what would happen if these algorithms were applied to problems where the space of possible behaviors is vast enough that thorough coverage is impossible? On such problems, QD's potential to act as an agent of creativity might be fully realized, generating and optimizing an endless number of new ideas.

While maze-navigation has served in the previous chapters as a good benchmark to compare different approaches to QD and thus begin to understand the factors that contribute to their success or failure, the spaces of different driving behaviors explored in these maze domains were not only

closed but also perhaps somewhat contrived and uninteresting. Indeed, for experimental purposes it was sufficient in previous chapters to distill the results down to a single quantifiable performance metric without looking in depth at the specific behaviors discovered by each run. In other words, the types of domains explored here thus far do not effectively showcase the creative potential of QD both because they are limited in scope and because they do not produce results that can be appreciated intuitively.

Instead of maze-navigation, this chapter takes QD in a different direction inspired by the field of *open-ended evolution* whose overarching goal is to design evolutionary processes that *continuously generate novel forms of increasing complexity*.

Open-endedness and Major Transitions

Open-ended evolution (OEE) as a field is inspired by the premise that life on earth arose through a natural process beginning from the simplest of forms and culminating in the vast diversity of living creatures that we observe today. To better understand how complex life developed out of nothing, the drive in OEE is to try to recreate this phenomenon through artificial means.

The emergence of complexity is arguably the most impressive and elusive feature of natural evolution. What *exactly* spurs the growth of functional organization from chaos and simplicity? Szathmáry and Maynard Smith [121] identify *major transitions* to help describe and explain the general increase in complexity observed in biological evolution. These transitions, including the development of prokaryotes into eukaryotes and eventually the development of human societies bound by a common language, deal primarily with changes in the encoding and transmission of information, leading eventually to increased complexity at the phenotypic level. The claim is that complexity is the emergence of tightly-bound aggregate entities composed of simpler parts. This

phenomenon of complexity-increasing evolutionary mechanisms is central to the idea of open-ended evolution (OEE), which roughly corresponds to indefinitely scalable evolutionary processes. According to de Vladar et al. [29], the presence of a major evolutionary transition is an important marker that delineates *weak* from *strong* OEE. While artificial life worlds have demonstrated the capacity for at least *some* features of OEE, understanding major evolutionary transitions and the mechanisms and conditions that enable them remains an important open problem for OEE [7, 125].

Over 20 years have passed since Szathmáry and Maynard Smith's publication on the topic, and in that time the artificial life and OEE communities have undergone substantial changes. Two important perspective shifts have been gaining momentum in recent years: (1) there may be many different kinds or degrees of open-ended evolution, and (2) open-ended evolution (of some kinds) may be observed in non-biological domains (such as some artificial life worlds and even high-level real world systems such as technological innovation) as well as in traditional biologically-inspired systems [125]. In their original formulation of the major transitions, Szathmáry and Maynard Smith focus on heredity because it is the biological means for information to be stored and transmitted. To study artificial life systems however, the concept of major transitions can beneficially be defined more broadly and may even take forms deviating largely from that observed in biological evolution. For this reason, it makes sense to broaden our interpretation of major transitions to admit mechanisms leading to indefinitely increasing complexity in non-biological domains as well. That is, it is conceivable that in artificial life domains, the major transitions that lead to an inevitable increase in complexity might be defined in a different manner than in biological systems.

Accordingly, in this chapter we follow the alternative model of major transitions proposed by Koonin [56] wherein major transitions are characterized as "brief bursts of innovation", or *a rapid diversification of novel and complex forms*. This formulation turns out to be useful for studying evolution in a cross-disciplinary context (which is ideal for artificial life) because it admits comparisons of complexity-increasing transitions in systems that are both biologically inspired and

not.

In biological system, one obstacle to reproducing the phenomenon of major transitions is the sheer amount of time required. The Cambrian explosion, for instance, occurred over 3 billion years after the development of the first cell on Earth [71]. Another difficulty arises from the lack of controllability of complex biological systems. In contrast, artificial life systems allow us to perform both tractable and controllable experiments and thereby test with rigor scientific theories about major open problems in evolutionary theory. However, not all simulations have the capacity to express levels of functional organization considered to be the hallmarks of complexity. One of the goals of the OEE community is to design "open-ended" simulations that not only have such expressive capacity but in which emergent complexity is actually realized. While there is much debate about when a system can be considered open-ended, one potential test is whether transitions to higher orders of complexity are observed within the system.

In an effort to create an open-ended platform for demonstrating the creative potential of QD, this chapter introduces a new domain called *Voxelbuild* where robots are evolved to build a diversity of block structures within a space of nearly endless possibilities. While the vast (potentially infinite) world and complete freedom of action afforded to the robots enables a great deal of expressive potential, this alone does not constitute a capacity for open-endedness. The experiment that follows demonstrates that when coupled with the evolutionary framework of quality diversity, transitions are observed in Voxelbuild in some runs wherein bursts of innovation are followed by a propagation of fundamentally more complex forms. The presence of major transitions in Voxelbuild indicates a degree of open-endedness, suggesting that future experiments applying QD in this domain may indeed yield a continuous discovery of novel and increasingly complex forms. Furthermore, because some runs achieve major transitions while others do not, the following experiment represents an opportunity to investigate hypothesized prerequisites for such transitions.

Voxelbuild

For the purpose of studying evolutionary transitions and long-term/open-ended evolution in the context of evolutionary robotics, this chapter introduces a new experimental platform called *Voxelbuild*, inspired by the Minecraft [1] video game that is well-known for facilitating seemingly boundless creativity. Voxelbuild, like Minecraft, serves as a sandbox in which embodied agents (instead of human players) can build structures by placing and removing blocks in a discrete three-dimensional grid-based world. Figure 8.1 shows an example of one such structure built by an agent after hundreds of ticks of simulation.

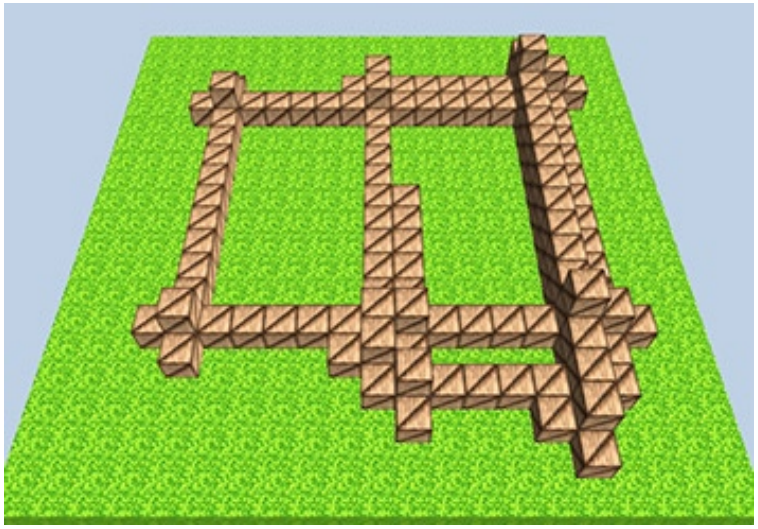


Figure 8.1: **The world of Voxelbuild.** In Voxelbuild, a simulated agent moves around and places blocks at discrete locations in an initially-empty world resulting in one of countless possible final structures whose impressiveness and complexity can be visually appreciated.

The Voxelbuild world is greatly simplified from that of Minecraft to facilitate faster evaluation and a smaller control network. Agents are two blocks tall and free to move around and build within their world subject to the following physical constraints: (1) agents are affected by gravity (i.e.

[1] Copyright (c) 2011 Mojang

they fall to the ground when there is empty space beneath them), (2) agents cannot walk through blocks, (3) agents may jump on top of blocks, but their jump height is limited to one block (jumping is achieved by moving in the direction of a single block at foot-level with at least two empty spaces above it), (4) blocks can only be placed adjacent to other blocks (however, there are no constraints on block removal and blocks are not subject to gravity, so it is possible to create "floating" structures by building upwards and then removing some of the blocks underneath)[2], and (5) block placement is limited to a small radius around the agent (here, the radius is 5, resulting in an $11 \times 11 \times 11$ area within which the agent can interact with the world). Subject to these constraints, building all but the simplest structures is a non-trivial task; especially difficult is building upwards because building above ground level is only possible by stacking blocks (which requires a level of precision in specifying block locations) and the only way to reach heights above the block placement radius is to stand on top of previously placed blocks (i.e. *scaffolding*).

In this initial experiment, for computational efficiency the world size is bounded to $21 \times 21 \times 12$ blocks. At the beginning of each evaluation, the environment is empty except for a single layer of "grassy" blocks at ground level. The agent is placed in the center of the world and allowed up to 10,000 ticks to perform combinations of six discrete actions: (1) turn left 90 degrees, (2) turn right 90 degrees, (3) move forward, (4) move backward, (5) place a "crate" block, or (6) remove a crate block (or grassy block). Importantly, a trial is terminated early if the agent requests an illegal move or enters a loop (e.g. moving back and forth or spinning in a circle). Thus in order to succeed in Voxelbuild, agents must first learn to avoid nonsensical moves such as walking through a wall or placing a block where one already exists. Because of early termination, actual trial lengths are generally much shorter than 10,000 ticks and vary widely according to how well each agent obeys the physical constraints of the world.

[2]Unlike in Minecraft, block placement and removal is not restricted by line-of-sight, although this restriction can be added to increase problem difficulty.

Agent Configuration

The discrete nature of both time and space in Voxelbuild simplifies the process of simulating and controlling agents through artificial means. In Voxelbuild, agents are controlled by evolved neural networks. On each *tick*, agents sense the world around them with an $11 \times 11 \times 11$ array of block sensors (1,331 sensors total), which extends in each direction as far as the maximum distance for block placement and removal (distance 5). Sensor values are determined according to the type of block present at the corresponding location (1 – crate or grassy block, 0 – world boundary block, -1 – no block). The controller network takes these block sensor values as input and outputs six *action selector* values where the highest value determines which action is taken by the agent. Ties between action selector values are resolved in order of the following priority: move forward, move backward, remove block, place block, turn left, turn right. When either the *remove block* or *place block* actions are selected, the location to place or remove a block is decided by an $11 \times 11 \times 11$ output array (ties are resolved deterministically with preference for closer locations). The exact network configuration including hidden layers is depicted in figure 8.2. Despite the large size of the input and output arrays, the number of connections in the network remains tractable by filtering connectivity through hidden layers of size $3 \times 3 \times 3$.

Experiment

The world of Voxelbuild is designed to facilitate experiments in open-ended creativity through the evolution of intelligent agents that construct diverse and increasingly complex structures. Of particular interest here is the phenomenon of *evolutionary transitions* wherein skills are acquired by the gene pool that enable the production of fundamentally different and more complex artifacts. The presence of transitions to higher orders of complexity is a hallmark of open-ended systems and suggests a potential for endless discovery. To study the prerequisites of such transitions, if

present, the task must be difficult enough that major transitions are observed in some, but not all, evolutionary runs. That way the developmental timeline of successful runs can be compared against unsuccessful runs to help identify how and why these transitions occur. Thus, building a structure in Voxelbuild is designed to be nontrivial – agents are given a great amount of freedom with their large radius of effect but are bound by the physical constraints of the world and therefore must learn how to move and act without violating its rules.

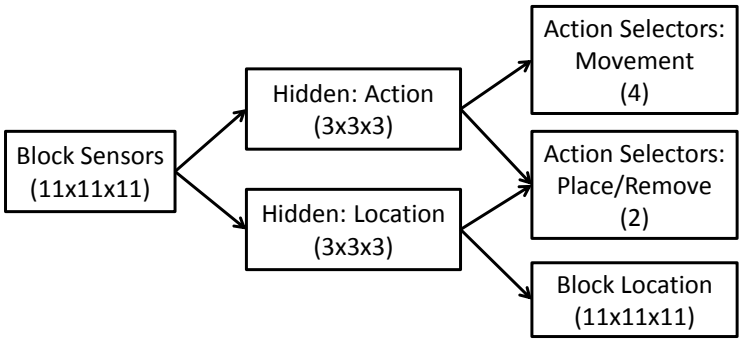


Figure 8.2: **Network configuration.** All layers shown as connected are fully connected: there are 108,027 connections in total. Connection weights are evolved with HyperNEAT. The neurons in each neuron group are spread evenly across three dimensions bounded between -1 and 1 on each axis.

While Voxelbuild supports the construction of an effectively endless array of unique structural artifacts, not all structures are possible without first acquiring certain fundamental skills, each of which might be considered a major evolutionary transition. During preliminary runs, it was observed that agents have a difficult time building vertically; more accurately, some runs find agents that learn vertical building and thus achieve better results, while agents in other runs continue building only at ground level, limiting their creations to only two dimensions. When considering the physical constraints of the world, two major evolutionary transitions in Voxelbuild are immediately conceivable, each of which opens the door to building structures of greater complexity. The first such transition is *vertical building* or block stacking. Before learning vertical building,

agents can only build effectively two-dimensional structures at ground-level. After agents learn to stack blocks at heights above ground level, new types of structures become possible that utilize all three dimensions. The second major evolutionary transition stems from agents' inability to place blocks outside of their small local radius. Faced with this restriction, it is impossible for agents to build structures taller than their block placement radius without first moving on top of a previously placed block. This process of placing a block and then moving on top of it is called *scaffolding* and represents a major evolutionary transition that enables agents to build structures of unlimited height. Because blocks must be placed adjacent to other blocks and agents cannot jump higher than one block at a time, moving in the z-dimension via scaffolding is an advanced behavior that requires careful coordination. Besides vertical building and scaffolding which are intuitively conceivable a priori, there may be still more major transitions in Voxelbuild that enable the building of different kinds of structures not possible in their pre-transition landscape.

The primary goal of this experiment is to demonstrate the potential for QD to act as a creative force in open-ended domains. To assess QD's ability to explore the vast space of structures possible in Voxelbuild, QD here is tasked to produce a diversity of structures that are *as large and tall as possible*. However, rather than evolving structures directly, QD evolves neural networks that control agents to move around the world placing blocks in a way that culminates in the final structure. In this way, the agents serve as instruments of creativity whose success is measured by the structures they are able to produce by the end of their trial. At first agents struggle to place even a single block. As evolution discovers controllers that more successfully navigate the rules of the Voxelbuild world, agents are able to build progressively larger and more intricate structures. Due to the high dimensionality of the behavior space (described in Section 8) preventing the application of a MAP-Elites grid, NSLC is selected as the QD algorithm to drive search.

An auxiliary goal of this experiment is to assess Voxelbuild's capacity for open-endedness by checking the evolutionary timeline for evolutionary transitions where specific innovations trigger

a cascade of behaviors (characterized by the final structures) fundamentally different and more complex than those observed before.

Investigating the presence of and circumstances surrounding such transitions requires performing several runs of evolution where some observe transitions and others do not. In this experiment, 10 runs of evolution are allocated the same amount of CPU time over a period of two weeks on a 160-core computing cluster. While each run is allocated the same amount of clock time, each run extends to a different number of evaluations due to variance in the average trial duration between runs (while trials are allowed up to 10,000 ticks, most terminate in under 200 ticks when the agent attempts an illegal action such as moving through a wall or placing a block where one already exists).

Experiment Parameters

The controller networks for Voxelbuild agents are evolved by HyperNEAT[3] extended with the multi-spatial substrate augmentation (Chapter 4) to realize the network configuration described in figure 8.2. Evolution proceeds with the following CPPN mutation rates: 5% add connection, 5% delete connection, 1% add neuron, and 1% delete neuron. The NSLC population size is 128 with a batch size of 32. The maximum size of the novelty archive is 512. On each epoch, the batch is added to both the population and the archive. The population is reduced back to capacity by deleting the individuals with the lowest Pareto-ranked fitness. Archive capacity is enforced by repeatedly drawing random tournaments of size 3 and deleting the participant with the lowest novelty score (in this way, novelty scores within the archive only need to be calculated for tournament participants). The QD collection holds 128 individuals and is dumped to file every 1,000 epochs (32,000 evaluations), allowing pre-transition collections to be compared to those after a transition.

[3]The HyperNEAT implementation here is a modified version of SharpNEAT 1.0 [46].

Fitness and Behavior Characterization

As a QD algorithm, NSLC requires both a *fitness function* to measure the quality of individuals and a *behavior characterization* that describes which aspects of an individual should be considered when calculating novelty. Fitness is calculated as follows, where b_{net} is the total number of blocks placed minus the number of blocks removed and h_{max} is the maximum height at which a block is placed:

$$\text{fitness} = b_{net} \times h_{max} \ .$$

This formulation rewards larger structures while emphasizing those that utilize the z-axis so that taller structures correspond to substantially higher scores. This approach makes it easy to identify when vertical building or scaffolding transitions occur but does not attempt to capture other possible aspects of structural quality. Behaviors are characterized by a vector of 5,292 binary values representing each location in the final state of the world, where a value of 1 means a block exists at that location and 0 means no block exists at that location.

Results

Figure 8.3 graphs the best fitness over the course of ten separate runs of evolution to illustrate when major evolutionary transitions occur, where transitions correspond to abrupt increases in fitness indicating that agents have learned vertical building. In two runs, vertical building is learned early (before the first QD collection dump at 32,000 evaluations). However, in two other runs, vertical building is learned later, allowing the state of the QD collection to be examined both before and after the transition occurs. As a testament to the difficulty of achieving this transition, six of the ten runs do not successfully transition in the given time frame. These "failed" runs are compared qualitatively to the state of successful runs before the transition to vertical building.

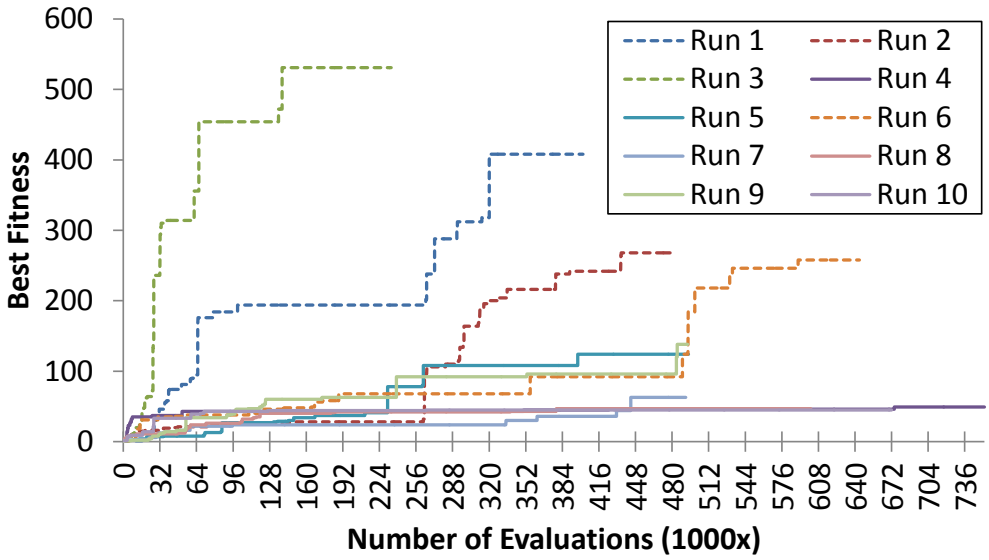


Figure 8.3: **Best fitness over time.** Graphing the best fitness over time for each run allows isolating when major transitions occur. High scores (above 200-250) correspond to achieving the vertical building transition. Out of the four successful runs (denoted by dashed lines), two runs transition early, while two other runs transition later. Tick marks on the x-axis indicate when the QD collection is dumped to file for analysis.

Examining the QD collections before and after major transitions are observed reveals a clear pattern of progression that is consistent across all runs.

Agents first learn to build *chaotic structures*, where blocks are placed haphazardly (figure 8.4). These structures are often quite small, suggesting that the lack of organization makes it difficult for agents to survive very long without performing an illegal move. Unsuccessful runs never progress beyond chaotic structures.

Runs that eventually succeed in learning vertical building (signifying a major evolutionary transition) first learn to place blocks in an organized manner (figure 8.5). In these structures, blocks are placed in straight lines or rectangles, with fewer gaps and less apparent randomness.

Four out of ten runs eventually gain the ability to build vertically. Structures at this point continue

to exhibit a high degree of organization, facilitating the construction of larger and more complex structures with features such as repeated motifs and some blocks placed above ground level (figure 8.6). Sometimes, agents employ primitive forms of scaffolding when placing blocks above ground level (e.g. agents build a large rectangular sheet at ground level, then stand on top of it to continue building on the next level up).

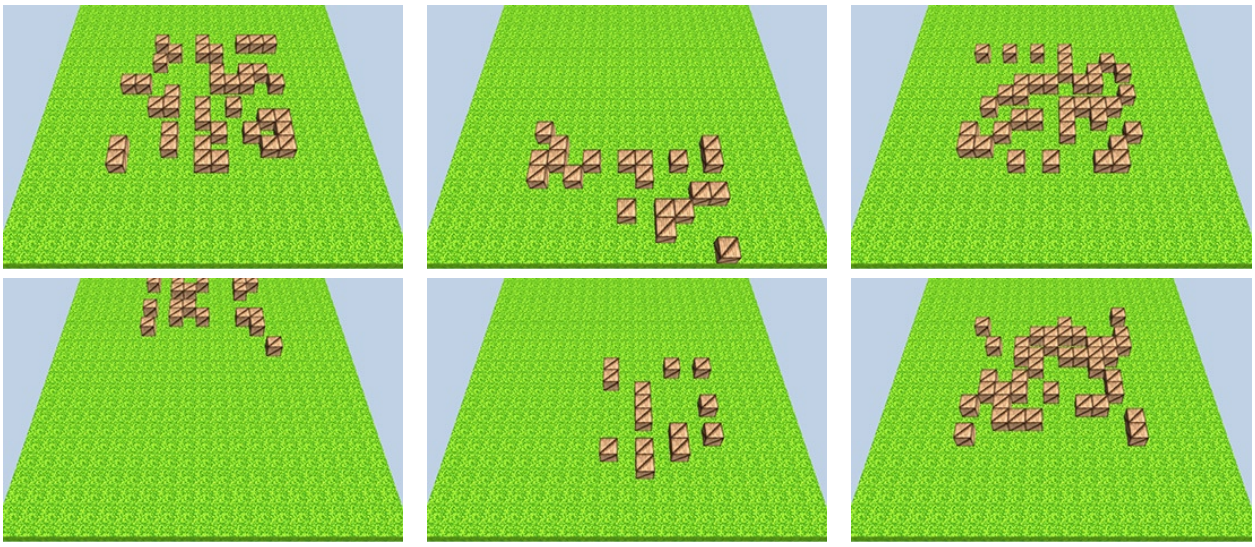


Figure 8.4: **Chaotic structures.** Chaotic structures are marked by haphazard, disorganized block placement and are the first types of structures that agents learn to build.

Discussion

This initial experiment reveals the first major evolutionary transitions in the world of Voxelbuild, the ability to place blocks vertically, preceded by an apparently necessary yet unanticipated preliminary transition: *the discovery of organization*. While vertical building is specific to Voxelbuild, its precursor, organized building, captures a principle that may generalize to other open-ended simulations and artificial life worlds: *organization precedes complexity*. Chaos in such simulations is sometimes celebrated as a sign of complexity (when we do not understand exactly what is going

on in a system, we may assume there are advanced strategies at play), but it is possible that in many cases chaos is actually a sign of evolutionary stagnation. In Voxelbuild, all runs begin with agents that place blocks haphazardly and no evolutionary trajectories are observed where chaotic building leads directly into vertical building, suggesting that organized behavior is an important *stepping stone* to more advanced behaviors. Similarly, deliberate and principled behavior, while sometimes less exciting to observe, may eventually lead to higher levels of complexity in other open-ended domains.

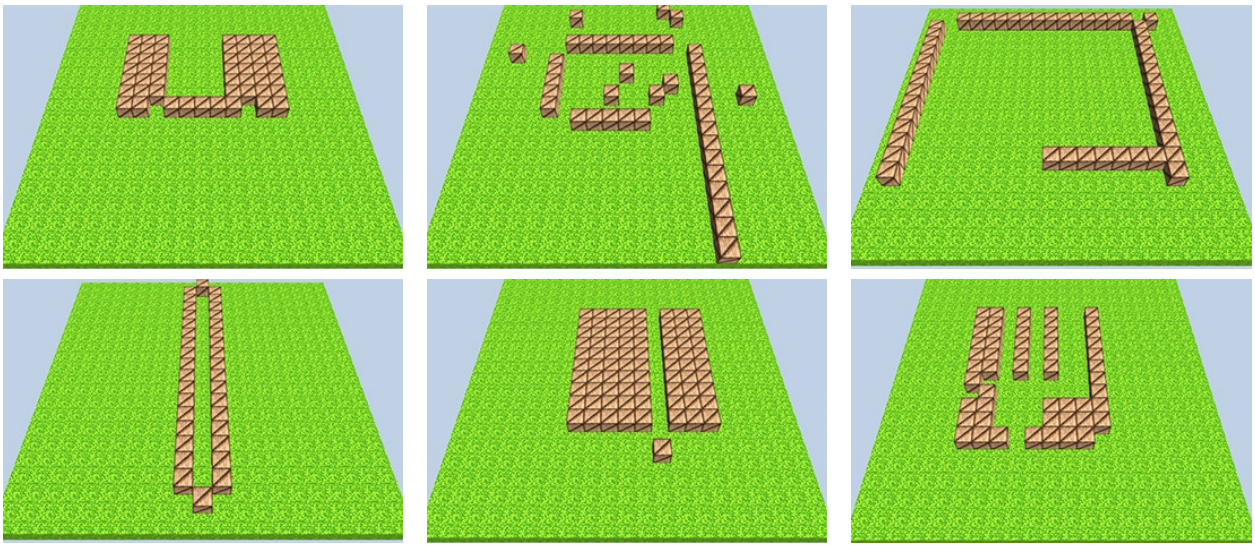


Figure 8.5: **Organized structures.** Primitive organization is marked by placing blocks in straight lines or rectangular shapes. This building strategy is the precursor for more advanced structures.

In Voxelbuild, agents must operate within the physical constraints of their world, which includes the stipulation that blocks must be placed adjacent to other blocks. Placing blocks at ground level easily satisfies this constraint because every location at height 1 is adjacent to the sheet of ground blocks at height 0. However, all positions at height 2 are initially illegal, requiring first placing a block nearby (usually underneath). Agents that place blocks seemingly at random (i.e. chaotic placement) have a difficult time building upwards because they have not learned how to satisfy the adjacency constraint. On the other hand, organized placement is characterized by contiguous lines

and rectangles, signifying that agents have learned to place blocks adjacent to each other at ground level and thus they also have an easier time satisfying adjacency in the vertical direction.

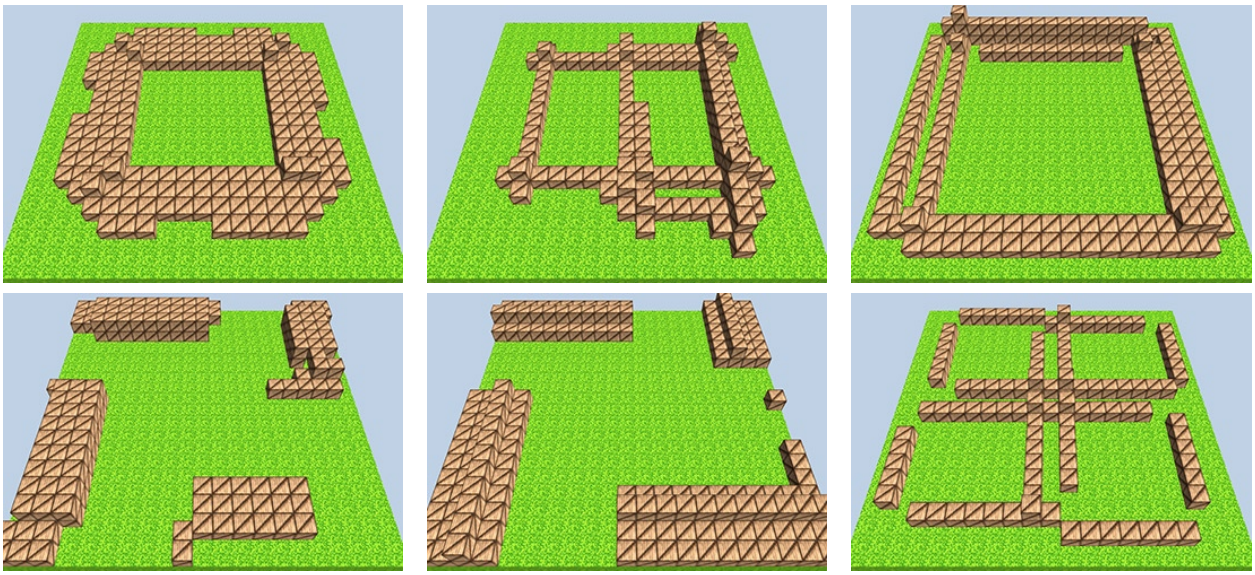


Figure 8.6: **Post-transition structures.** After learning organized block placement, larger and more complex structures become possible, including the ability to build vertically. In these examples, some blocks are placed at heights 2 and 3.

The principle of organization preceding complexity can be observed in nature. For example, early forms of life consisted of *prokaryotes* – single-celled organisms whose components are not membrane-bound but rather all float together in the cytoplasm. Later in Earth's evolutionary timeline, *eukaryotes* (organisms with cells whose specialized components are clearly separated into organelles by thin membranes) began to appear [98]. The separation of cellular components into functionally distinct organelles represents an advancement in organization that perhaps opened the door to multicellular organisms and in turn to the diversity of complex life that we observe today.

de Vladar et al. [29] characterize evolutionary innovations as "the population stepping into pre-existing but unoccupied dimensions of the embedding trait space" (drawing an interesting parallel with how QD may discover new regions of the behavior space) and further suggest that such in-

novations are heralded by "new ways of interacting with the environment." In Voxelbuild, the evolved behavior of constructing a series of straight lines and platforms (the latter of which sometimes functions as a primitive form of scaffolding) to build larger and more complex structures clearly constitutes a new way of interacting with the environment, thereby qualifying as an innovation in this sense. The rapid diversification and complexification of structures observed following this innovation signifies a major evolutionary transition that opens the door for new kinds of behaviors that were previously difficult or impossible. According to de Vladar et al. [29] "weak open-endedness" (an endless procession of novelty) is distinguished from "strong open-endedness" by a series of paradigm-shifting innovations. This initial experiment in Voxelbuild demonstrates at least the beginnings of such a timeline of evolutionary innovation, suggesting that Voxelbuild has some open-ended properties and thus represents a promising domain where perhaps QD can exhibit unbounded creativity.

As observed in Voxelbuild, the evolution and complexification of life on earth is often characterized by a stepwise (not gradual) pattern [34, 105]. In fact, de Vladar et al. [29] argues that such steps, each corresponding to major transitions, are impossible to define completely a priori because many innovations arise from *exaptations* (traits that are evolved for one purpose but later used for another purpose). In Voxelbuild, the appearance of primitive scaffolding is itself an exaptation because standing on top of previously-placed blocks is not evolved for any particular purpose but later on will enable the building of structures more than six blocks tall. The dependence of some evolutionary innovations on stepping stone behaviors that arise randomly with no purpose at all (and thus with no guarantee that their discovery will be "remembered" in the population) emphasizes the nondeterministic nature of the evolutionary process, making it impossible to guarantee any particular outcome. In sufficiently high-dimensional spaces, many runs of evolution may each take an entirely different path through the space of possible evolutionary trajectories. Given this lack of strict order, it is not surprising that only some runs exhibited the organized building style

that is prerequisite to the vertical building major transition.

In open-ended evolution, which strives for an effectively endless progression of meaningful complexification, major transitions represent evolutionary bottlenecks where progress may seem to stall. However, these bottlenecks are to be expected and may even be necessary for many of the most important innovations. In Voxelbuild, as on Earth, the road to more advanced behaviors and artifacts is marked not by incremental progress, but rather by sudden bursts of development following the discovery of important traits that introduce fundamentally new ways of interacting with the world. With this in mind, it is likely inappropriate to drive OEE solely by some concrete measure of fitness; while chasing ever-increasing fitness may be suitable for simple adaptation, it does little to encourage exaptations – the impetus for true innovation (in this context, exaptations are traits that appear in the population but are not rewarded by the predefined fitness measure). Instead, chasing a gradual series of adaptations towards a predefined target may actually take search in the wrong direction, where it can become trapped in an evolutionary dead end (a phenomenon commonly known in the evolutionary computation community as *deception*). In fact, the very presence of a fitness function in a way assumes that there is an end goal in mind. However, true OEE should continuously generate interesting new artifacts and behaviors beyond what can be conceived of a priori. Just as a silent observer of Earth four billion years ago would not have been able to imagine the eventual development of modern plant and animal life, open-ended evolution should not constrain the search process with preconceptions about what is possible or desirable, but rather should leave evolution free to explore and discover, not strictly pursuing what is regarded as "better" but instead pursuing what is different.

In this initial study into evolutionary transitions in Voxelbuild, some runs succeed in transitioning in the allotted time frame, while others do not. This raises an important question: *given more time, will unsuccessful runs eventually reach a major transition, or have they gone down an evolutionary trajectory where they will remain stuck forever in the space of simple behaviors?* Relatedly: *given*

more time, will successfully transitioned runs continue on to cross the threshold of still more major transitions, thus producing increasingly complex and interesting behaviors? To answer these questions, we must consider that evolutionary runtime is limited by available computation power. However, in this experiment as in many others, the bulk of computational power is spent running multiple rounds of evolution in parallel to be sure that at least some successful runs are observed, severely limiting the duration of each individual run. Studies like this one of the circumstances surrounding evolutionary transitions, especially of the factors that either promote or impede innovation, are therefore important because they may inspire new approaches that can more reliably achieve such milestones. Then it becomes more realistic to devote with confidence to a small handful of evolutionary runs or even a single run, allowing substantially deeper exploration into the hidden possibilities of open-ended domains such as Voxelbuild.

Conclusions

This chapter introduced a new domain called Voxelbuild for studying QD algorithms in the context of an open-ended problem where embodied agents move and build structures out of blocks. Initial experiments here reveal the presence of major evolutionary transitions where agents learn fundamentally new ways of interacting with the world, thereby unlocking more complex behaviors than were previously possible. Similar major transitions can be found in Earth's evolutionary timeline (most notably in the Cambrian explotion, where primitive multicellular life rapidly developed into the vast diversity of complex species observable today) and are considered by some to be an important hallmark of strong open-ended evolution.

Observing transitions in Voxelbuild gives evidence that it has some open-ended properties, making it a promising candidate for further study into how QD algorithms might behave in the context of a domain with an effectively unbounded variety of possible behaviors waiting to be discovered.

The hope is that QD in such domains might serve as a creative force, continually encountering and refining an endless stream of novel and increasingly complex behaviors.

CHAPTER 9: IMPROVING THE VOXELBUILD PLATFORM FOR LONG-TERM EVOLUTION

The preceding chapters presented the emerging field of quality diversity (QD) as an alternative ambition for evolutionary computation outside of its historical focus on pure optimization and featured a series of experiments designed to explore the factors that affect their behavior and performance. These experiments extensively compared existing published QD algorithms as well as several new variants introduced herein (e.g. ME-PGD, ME-NOV, and an array of multi-BC approaches) on a series of progressively more difficult maze navigation tasks.

Recognizing that maze navigation and many other published QD applications represent *closed* tasks where it is not only feasible but often even expected to explore and optimize the entire behavior space, a new domain for QD called Voxelbuild was introduced in Chapter 8 to pursue QD's potential to design and discover in an *open-ended* task where the behavior space instead contains unlimited possibilities. Because the concept of open-endedness involves more than an infinite behavior space, Voxelbuild was investigated for an important indicator of open-ended potential: the presence of *major evolutionary transitions* wherein key innovations trigger a rapid discovery of new forms of greater complexity that are fundamentally different from those possible before.

While these initial experiments presented in Chapter 8 do exhibit some major transitions, they are limited in their ability to explore deeply the evolutionary timelines possible in Voxelbuild both due to constraints on available computation time as well as the apparent inability to achieve the presupposed *scaffolding* transition that is necessary to build structures of unlimited height and thus progress beyond the current best results.

This chapter introduces a series of modifications to the encoding and search process in the Vox-

elbuild platform with the goal of achieving the scaffolding transition (thus unlocking a breadth of larger and more complex structures) as well as speeding up evaluations to enable a deeper exploration into the breadth of possibilities in the Voxelbuild "tree of life" in the same amount of computation time. One of the core goals of this dissertation is to study the behavior of QD algorithms in open-ended tasks over long time periods approaching the order of magnitude of the number of generations experienced by evolution on Earth in the transition from one species to another. By giving agents more tools to succeed and thus unlocking the ability to build in the z-dimension, the hope is that QD in Voxelbuild continues to innovate even over very long timescales, which would represent the first demonstration of QD as an effective driving force for open-ended discovery.

Reducing Dimensionality of the Voxelbuild Substrate

One major limiting factor in Voxelbuild, as in all search algorithms, is time. In the pursuit of better results, the simplest approach to try is to spend more time searching. While available computation time may be limited, it is possible to extend evolutionary time (the number of generations or evaluations conducted by the search) by speeding up evaluations.

In Voxelbuild, agents are controlled by large neural networks that must be fully activated on each tick of simulation to process sensory information and ultimately decide which action to take. In the configuration from Chapter 8, network activation represents the bulk of the time spent on each evaluation due to the large number of connections in the substrate. Recall that the input and output layers in the substrate contain groups of $11 \times 11 \times 11 = 1,331$ neurons (corresponding to a sensor radius of 5), each of which is fully connected to groups of $3 \times 3 \times 3 = 27$ neurons in the hidden layer (figure 9.1). One instance of this 1,331-to-27 connectivity pattern represents 35,937 connections and there are three such instances, together comprising 99.8% of the total number of connections.

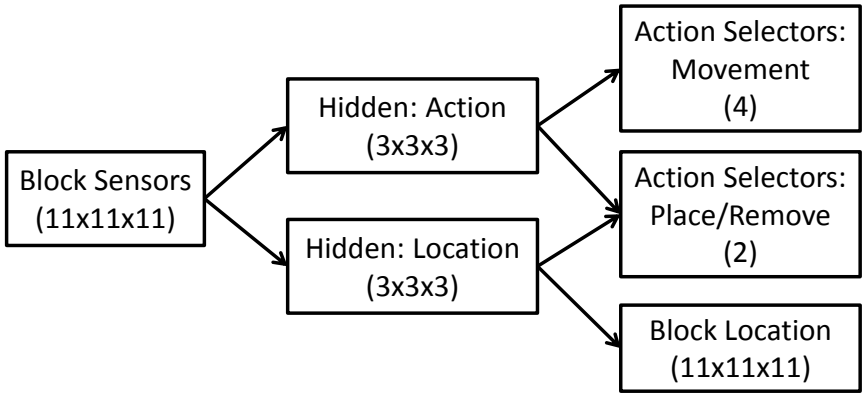


Figure 9.1: **Old network configuration.** In the substrate configuration from Chapter 8, input and output neuron groups containing 1,331 neurons are fully connected to hidden neuron groups containing 27 neurons, together comprising 107,811 of the substrate's 108,027 connections.

By reducing the agent's sensor and effector radius from the original 5 down to 3, 2, or even 1, the number of neurons in the input and output layers and thus the number of connections in the substrate can be substantially reduced. In a $7 \times 7 \times 7$ configuration (radius $= 3$) there are 3.9 times fewer connections; in a $5 \times 5 \times 5$ configuration (radius $= 2$) there are 10.6 times fewer connections; and finally, in a $3 \times 3 \times 3$ configuration (radius $= 1$) there are 49.3 times fewer connections. In the radius $= 1$ configuration employed in this chapter, the number of connections in the substrate is only 2,349 (figure 9.2).

Reducing the size of the control network in this way directly translates into faster evaluations (up to 40 or 50 times faster), allowing search to proceed much deeper into the evolutionary timeline of Voxelbuild in the same amount of time as the experiments presented in Chapter 8. If longer evolution time was the missing ingredient, this speed increase alone may be sufficient to achieve the scaffolding major transition necessary to potentiate structures of unlimited height. However, the speedup does not come without side-effects – modifying the size of the sensor array fundamentally changes how the agent interacts with the world and may influence the type and quality of structures that agents build during their lifetime. For example, with a small sight and block placement radius

(e.g. radius $= 1$), agents can only build relatively simple patterns while standing in one place; consequently, more complex building strategies can only emerge through frequent moving and re-positioning. Another side-effect is that the new configuration may improve the efficiency of information processing through the agent's control network because the input, hidden, and output layers now all operate with the same $3 \times 3 \times 3$ configuration (i.e. information is no longer forcibly lost as activation passes through the hidden layer).

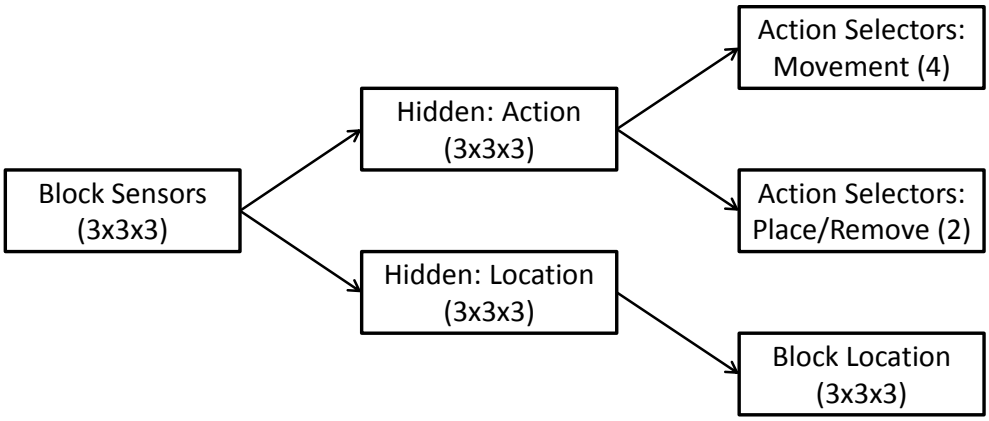
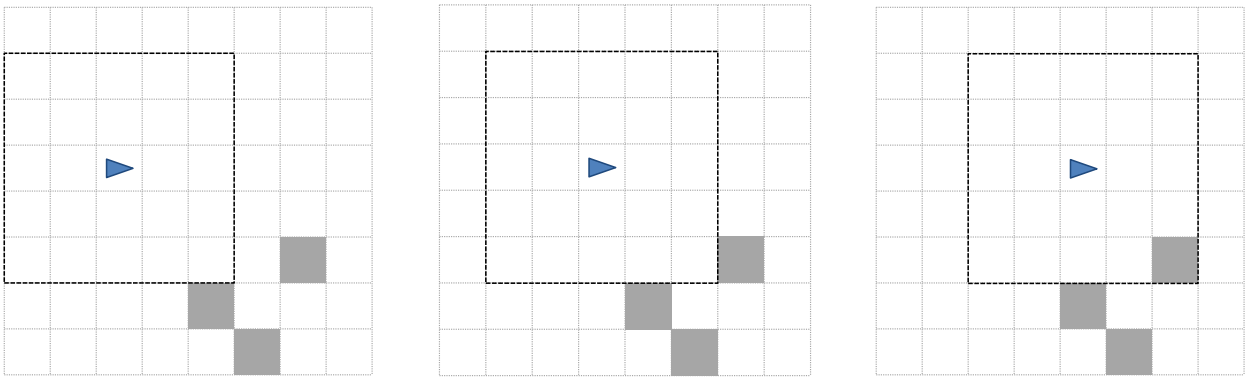


Figure 9.2: **Reduced network size.** A $3 \times 3 \times 3$ array of block sensors is connected to the outputs through two parallel $3 \times 3 \times 3$ hidden layers – one to process information about action selection and the other to process information about block placement. In this chapter, the size of the input and output layers (which corresponds to the maximum range agents can sense and place blocks) is reduced from 11 to 3 (radius 1). The total number of connections in this configuration is 2,349 (reduced from 108,027). An important change from the old substrate configuration is the removal of connections from the *location* hidden layer and the *place/remove* action selectors (the old configuration is depicted in figure 9.1). The presence of such connections in the old substrate may suggest an inappropriate geometric relationship between the hidden neurons and the two action selectors that are embedded at arbitrary locations in space. By removing these connections, the *location* hidden layer is free to "think" purely in terms of block locations, thus matching the information content of the input and output layers connected to it.

Information Desert Effect

In Voxelbuild, agent behavior is purely a reactive function of sensory experience on the current tick. When this sensory experience does not change from one tick to the next, the agent will request the same action, which can manifest as repetitive or trivial behavior patterns. Agents are placed into a world that is initially flat and empty (no blocks in sight). If an agent moves from this initial position without first placing a block, their sensory experience does not change and they will repeat the same move action over and over until reaching the world boundary where symmetry is finally broken and the control network has an opportunity to encode a different action response to this new sensory experience (seeing a wall of boundary blocks in the field of view). This pathology is called the *information desert effect* and is exacerbated by larger world size or a smaller field of view. An example of an agent moving through an information desert is shown in figure 9.3.



(a) Tick N (b) Tick $N + 1$ (c) Tick $N + 2$

Figure 9.3: **Information desert example.** In this example an agent with a sight radius of 2 begins with no blocks in its field of view (9.3a). After moving forward, the agent experiences no change in sensory information compared to the previous tick and can only repeat the same action as before: move forward (9.3b). One can imagine this process repeating for a very long time until eventually a block enters the agent's field of view, thus freeing it from the information desert (9.3c).

If the agent places a block upon reaching the world boundary, they are then able to localize and

afterwards are unlikely to move back into an information desert situation because moving around near the placed block now changes their sensory experience, creating a wide range of possible experiences that are each mapped to one of a potentially wide range of actions. Some such actions may include placing more blocks, further adding to the richness and information content of the environment.

However, reducing the sensor-effector radius increases the likelihood of moving back into an information desert even after escaping. In the case of a radius $= 1$ configuration, the agent can move back into an information desert after placing a block simply by stepping away from it. After moving only one block unit away, the agent immediately reverts to the action that their control network is encoded to select in the absence of sensory information (likely this is to continue moving forward until hitting a world boundary again). One can imagine trivial, repetitive, and time-consuming behaviors such as placing a single block and then moving across the entire world to the opposite world boundary before placing another block. The increased likelihood of re-entering a previously experienced sensory state (especially an information desert) as sensor radius decreases causes agents to exhibit increasingly repetitive behavioral patterns. In other words, reducing the connectivity in the substrate to speed up evaluations comes with the cost of lower complexity agent behaviors, which in turn may negatively affect the complexity of structures that can be produced.

Adding Homing Sensors

To meaningfully mitigate the information desert problem, agents needs access to a form of sensory information that changes as they move around the world even when the block pattern observed within their sensor radius remains the same. The core problem stems from the agents' inability to localize in the absence of a landmark. Without placing their own landmark on the first turn, agents must rely on the world boundary as a landmark (this is the most common strategy observed in the

results in Chapter 8). Reliance on the world boundary is a crutch that precludes the possibility of ever *removing* the world boundary in future iterations of Voxelbuild to truly provide endless potential for creativity. However, the more immediate concern in this chapter is that the lack of localization ability causes repetitive and uninteresting behavior patterns that continue from one end of the world to the other, severely limiting the possibility of changing between different building strategies throughout the building process.

Fortunately, localization can be easily served to the agent in the form of a "homing sensor" that provides the distance and direction to the spawn point. With such a sensor, the information desert effect disappears, and the control network gains the ability to specify different behaviors for otherwise identical block sensor states. With access to this new information, agents can alter their building strategy at any time, making their own decision about how long to repeat the same pattern before choosing a new one. With building patterns no longer anchored to the world boundary, more interesting structures should become possible, even when block sensor information is drastically reduced (as in the radius $= 1$ configuration in this chapter).

A simple homing sensor array is implemented as a $3 \times 3 \times 3$ grid of neurons where only one neuron is active at a time – the one that points in the relative direction of the spawn point. Distance is indicated by the strength of the activation signal where progressively smaller values are observed as the agent moves further away from the spawn point. More specifically, activation strength decreases linearly from 1.0 at the spawn point to 0.0 at the furthest corners of the world (the upper four corners of the bounding box). Note that because the spawn point is located at the bottommost level of the world, it is impossible for the upper nine homing sensor neurons to activate.

Integrating the new homing sensor into the substrate requires a separate hidden layer to process homing sensor information in addition to the hidden layer that processes block sensor information (figure 9.4). In this new substrate configuration, both types of sensors share the same $3 \times 3 \times 3$

layout as the hidden layers which in turn share the same $3 \times 3 \times 3$ layout as the block placement actuators. This consistency allows information to flow through the network without changing scale from one layer to the next, which may make effective control networks easier to evolve.

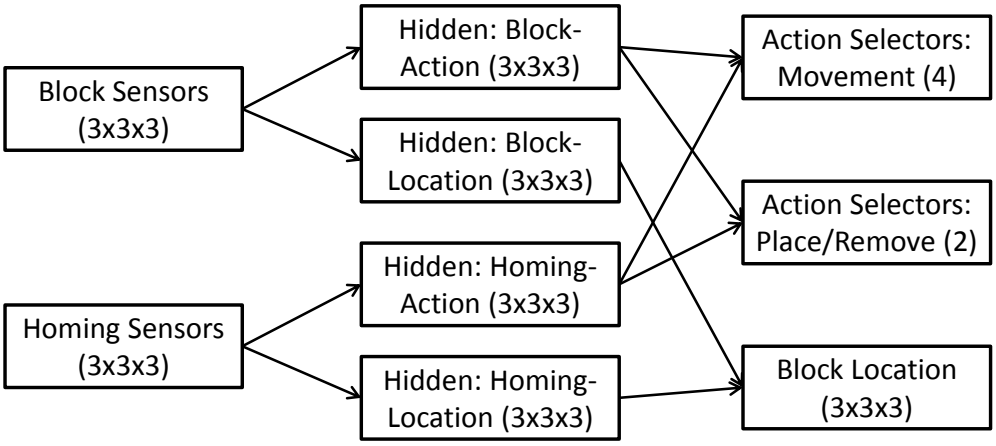


Figure 9.4: **Shallow substrate with homing sensor.** In this network configuration, a new input layer corresponds to a new $3\times3\times3$ array of homing sensors to facilitate agent localization. Because these sensors process a fundamentally different type of information than the block sensors, they are connected to two new hidden layers parallel to the two existing layers for processing block sensor information. These new hidden layers allow homing and block sensor information to be processed separately before being combined at the outputs.

New Fitness Function and Behavior Characterization

In its original formulation (Chapter 8), Voxelbuild scored final structures with a fitness function designed to highlight major transitions but that did not represent any intuitive notion of structure quality. Recall that the fitness function from Chapter 8 is $num\_blocks \times highest\_block$, which overemphasizes the importance of placing a single block as high as possible. Under this scheme, there is no direct reward for building a nontrivial vertical structure. Instead, the easiest way to achieve a high score is to place as many blocks as possible at ground level and then build a narrow tower upwards to multiply the score by the height (even a 1×1 tower is sufficient; e.g. placing

a single block on top of a full sheet at ground level effectively doubles the score). While this approach was ideal for highlighting the exact moment when vertical building is first discovered, it does not indicate when *nontrivial* vertical building appears (if ever).

Nontrivial vertical building is defined as placing *many* blocks above ground level. Higher scores should be assigned to structures that actually take up space in the z-direction. To illustrate with a real world example: a lighthouse would receive a high score while a flag pole would not. A new fitness function is introduced in this chapter that captures the spirit of nontrivial vertical building: *the sum of block values where each block is valued according to its height*.

Specifically, the new fitness function assigns each block a value equal to $height^2$ up to a maximum value of 100 (achieved at height $= 10$). This quadratic formulation puts extra emphasis on placing blocks far above ground level. Placing blocks at e.g. height $= 10$ is a non-trivial task for two reasons: (1) height $= 10$ is outside of the agent's block placement radius and therefore agents must employ many levels of *scaffolding* to reach such a height, (2) the adjacency constraint dictates that all blocks must be attached to other blocks and as a consequence constructing the necessary scaffolding requires a carefully coordinated sequence of movement and block placement actions. Under the regime of the new fitness function, the only way to achieve the highest scores is to achieve the scaffolding major transition (which has heretofore not been observed in Voxelbuild).

Behavior Characterization

The original Voxelbuild behavior characterization (BC) from Chapter 8 simply consists of a massive binary vector where each location in the final world state is assigned a value of 1 for "block" or 0 for "no block." This naive characterization is problematic for several reasons. First, it is not spatially aware. Shifting a structure over by 10 units is assigned the same behavioral distance as if it was shifted over by 100 units. Furthermore, this binary array characterization does not

capture any intuitive structural features such as size, shape, density, or orientation. Structures are considered very far apart in the behavior space as long as they contain blocks in different places. For example, imagine a chess board where black squares correspond to blocks and white squares correspond to empty space; such a structure is regarded as maximally different from an "inverse chess board" where instead *white* squares correspond to blocks. However, intuitively these two structures are almost identical: they have the same height, width, length, density, and alternating checkerboard motif. Finally, the high dimensionality of the binary vector characterization (5,292 values in the previous chapter and more values must be added if the world size is expanded) takes a substantial amount of time to compute and corresponds to a vast behavior space that may be difficult for evolution to intelligently explore because it is very easy to obtain high novelty scores with only trivial changes.

This chapter introduces a new low-dimensional BC that attempts to more closely match human intuition about structural similarity. The new characterization is composed of several different features that together attempt to describe a structure without specifying the exact location of each block. By focusing on intuitive structural features, QD is able to drive search towards structures that are diverse in a way that is appreciable to the human eye:

- Height
- Width
- Depth
- Average width (averaged over the z-dimension)
- Average depth (averaged over the z-dimension)
- Internal center of mass (x coordinate)

- Internal center of mass (y coordinate)
- Internal center of mass (z coordinate)
- Density
- *Upper density*
- *Thickness*

Height, **width**, and **depth** are world-relative (e.g. maximum width is achieved when the final structure contains blocks on opposing sides of the world boundary in the x-direction). **Average width** and **average depth** are also world-relative but averaged over the z-dimension (e.g. maximum average width is achieved if blocks appear on opposing sides of the world boundary in the x-direction at every layer of the structure in the z-direction; layers where no blocks are present are not counted). **Internal center of mass** and **density** are structure-relative (i.e. calculated according to the final structure's height, width, and depth boundaries rather than the world boundaries). **Upper density** is calculated similarly to density, except blocks at each level are weighted according to their height to give a measure of "top heaviness." **Thickness** is defined as the ratio between the final structure's average area (average width $\times$ average depth) across all z-layers and its maximum area (width $\times$ depth); thus maximum thickness is achieved by a structure with maximum width and maximum depth at all z-layers where blocks are present.

The resulting *structural features BC* is a vector of 11 values, substantially reduced from the 5,292-value binary array in the previous chapter. While its low dimensionality makes novelty score calculations trivially fast and does not scale up with world size, it is still far too large to apply MAP-Elites or QD-Grid collection at a fine enough granularity to adequately describe the space (e.g. discretizing each dimension into only 5 bins would result in a grid with over 48 million bins).

As with the binary array BC, this new structural features BC must be applied with NSLC and a grid-free QD collection mechanism.

Adding Substrate Depth

Thus far, all of the modifications introduced in this chapter have been designed to target specific shortcomings inferred from the initial Voxelbuild results presented in Chapter 8. Substrate connectivity was reduced to speed up evaluation time and provide a consistent $3 \times 3 \times 3$ scale across all network layers. A new homing sensor was added to eliminate a pathological "information desert" phenomenon that caused agents to build in repetitive patterns anchored to a world boundary. The fitness function and BC driving evolution were changed to more closely match human expectations of quality and diversity. However, one important variable in the Voxelbuild configuration that can only be optimized through empirical validation is *the structure of hidden layers within the control network* (i.e. how many hidden layers exist in the substrate between the inputs and outputs as well as how they are connected to each other). This section introduces an alternative substrate architecture inspired by recent successes in the field of deep learning [16, 58] – *increasing substrate depth.*

First-level hidden layers can express relationships between different groups of sensor neurons where the array of neurons in the hidden layer is a map of *features* that each synthesize the inputs in different ways. For example, one first-level feature might be potentiated by a combination of blocks to the left of the agent while another may be potentiated by blocks to the right of the agent. By inserting more hidden layers between existing hidden layers and the output layers, the network is given the opportunity to develop progressively *higher-level* features (relationships between *lower-level* features in the previous layer) to describe the observed world state. Combining the two features in the previous example allows the expression of a higher-level feature where

blocks appear on both the left and right of the agent without having to explicitly specify this combined feature in the lower-level layer. While this issue may on the surface seem like a trivial matter of adding *more* features rather than *more complex* features, consider that combining features in this way allows lower level features to be spatially organized without disrupting HyperNEAT's ability to express gradients of connectivity patterns by crowding the substrate with arbitrarily-placed "combination features." (Where would the example left-and-right feature be located if it was in the lower-level array? The answer may be "somewhere around the middle," but that is the case for most higher-level features and there simply is not enough room to express all such features.) Furthermore, higher-level features allow an additional layer of nonlinearities, increasing the expressive power of the network.

In the current **shallow substrate**, activation from the input layer feeds through a single hidden layer before arriving at the output layer. Importantly, the hidden layer is divided into independent neuron groups corresponding to different input and output modalities. The block sensors and homing sensors are each synthesized separately through their own dedicated hidden neurons, which are each further subdivided into two groups, one for each output modality (action selection and block location). The resulting shallow substrate configuration contains a single hidden layer divided into four independent groups, as depicted in figure 9.4).

An alternative **deep substrate** configuration is proposed here that adds a new hidden layer between the existing first-level hidden layer and the output layer. This new second-level layer provides an opportunity for information from each input modality to be synthesized before proceeding to the outputs. While input modalities are combined, the separation between output modalities is maintained, resulting in a second-level hidden layer composed of two independent groups. The deep substrate is depicted in figure 9.5.

The question at hand is whether the increased expressive power offered by the deep substrate's

additional hidden layer actually translates to better or fundamentally more complex results than the shallow substrate. Even if the *potential* for more advanced behaviors is increased, deeper networks are generally more difficult to train and thus this potential may not be realized. To resolve this uncertainty, both the shallow and deep configurations are included in the experiment described in the next chapter.

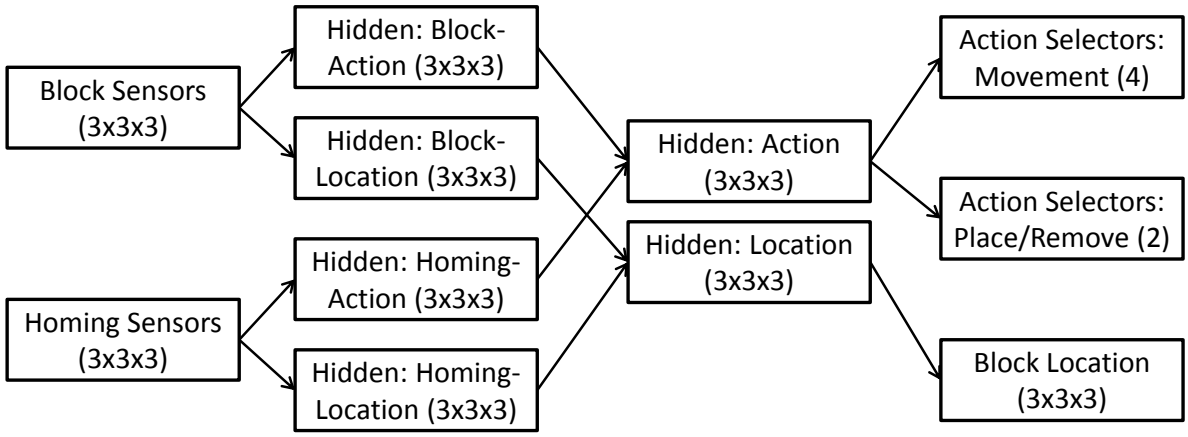


Figure 9.5: **Deep substrate with homing sensor.** In this network configuration, an additional hidden layer is placed between the first hidden layer and the outputs. While the first hidden layer processes block sensor and homing sensor information separately, the second hidden layer combines this information before it continues on to the outputs. The new hidden layer effectively allows the expression of additional nonlinearities in decision-making, possibly facilitating more advanced agent behaviors.

CHAPTER 10: EXPERIMENT: LONG-TERM INNOVATION IN VOXELBUILD

Informed by initial experiments, the previous chapter presented a series of modifications to the Voxelbuild platform designed to improve the speed and reliability of evolution and direct search towards results that more closely match human intuition about structural quality and structural diversity. These modifications also address shortcomings in the network architecture and give agents more tools to succeed in hopes of achieving the scaffolding transition, thus unlocking all dimensions of the world for building an effectively endless diversity of structures. With developmental roadblocks out of the way and the scaffolding transition achieved, the question then arises: *what happens when quality diversity in Voxelbuild is allowed to continue evolving for a very long time?*

Unlike optimization, which ultimately converges to some optimal performance, quality diversity search (because of its divergent nature) has no clear stopping point outside of exploring *all* possibilities in the domain to which it is applied. In an open-ended domain with endless possibilities, can QD continue to generate a diversity of increasingly complex novel forms forever? Running Voxelbuild over a long timescale presents unique opportunities to study the nature of open-endedness as well as the potential of quality diversity evolution as a creative tool. This chapter presents such an experiment, where evolution in the improved Voxelbuild platform is allowed to continue for several orders of magnitude longer than previously studied.

Exploring the idea of long-term evolution in Voxelbuild means confronting specific questions about how evolution behaves over time. For example, does the rate of discovery slow down over time or even stagnate altogether? Are discoveries gradual or episodic in nature? In other words, does evolution appear to get stuck exploring a particular theme for a long period of time, followed by a rapid burst of innovation signifying transcendence beyond some major transition yet unknown?

Are structures similar from one run to the next, or is each run fundamentally different? As evolution progresses, do structures follow similar patterns to those found earlier in the run, or are some motifs explored and then later abandoned? Structural features aside, do *agents* continue to learn new building strategies as evolution progresses? Specifically, do agents later in the evolutionary timeline eventually learn to employ many different block placement patterns throughout their trial as opposed to the single-strategy builders discovered early on?

Of particular interest in long-term Voxelbuild is the development of *complexity* over time. Truly open-ended discovery should not only continuously generate novel forms, but those forms should increase in complexity over time. In Voxelbuild, there are three major components to the complexity of each individual: (1) the complexity of agent control networks, (2) the complexity of generated structures, and finally (3) the complexity of agent behaviors over their lifetime.

Throughout the results presented in this chapter, the progression of structural and behavioral complexity is described by a series of snapshots of the QD collection sampled from several points throughout a typical evolutionary timeline. Structural complexity is visually evident and because final structures in Voxelbuild are not evolved directly but are actually the end product of thousands of ticks of an intelligent agent's sensor-action loop over the course of its lifetime, the complexity of agent behaviors directly corresponds to structural complexity. In other words, structures containing more blocks can only be achieved as agents obtain a better understanding of the "rules" of the world, enabling them to survive for more ticks before their trial is terminated (by attempting to perform an illegal move such as moving through a solid block or placing a block where one already exists). Additionally, more intricate structures result from agents learning more advanced behaviors observable through the emergence and combination of organized structural motifs.

Because agent control networks in Voxelbuild are encoded with HyperNEAT, they have a static topology but a variable pattern of connection weights. The weight patterns in early generations are

simple gradients, resulting from minimal CPPNs (compositional pattern producing networks) with no hidden nodes. The weight patterns in the control network (HyperNEAT substrate) grow more complex over time as more hidden nodes and connections are added to the genome (CPPN). Thus, the complexity of agent control networks can be measured by the average number of neurons and connections in the genome.

The next section describes how the long-running Voxelbuild experiments are configured.

Experiment

In the experiments described in this chapter, intelligent agents controlled by HyperNEAT-encoded neural networks are evolved with the NSLC (novelty search with local competition) QD algorithm. These experiments share the same configuration as the initial Voxelbuild experiments described in Chapter 8 modified by the changes to agent sensors, control network (substrate) structure, behavior characterization, and fitness function outlined in Chapter 9. These modifications drastically shorten evaluation time, allowing evolution to perform many more evaluations in the same amount of time, thus progressing deeper into the evolutionary timeline of each run to better explore the space of possibilities in Voxelbuild. Exploring deeper into evolutionary timelines allows the discovery of more advanced behaviors that build on and combine skills learned in earlier generations. In addition to speeding up evaluations, the experiments in this chapter have been allowed to continue significantly longer in terms of wall clock time than those in Chapter 8. Specifically, the experiments here are the result of 162 days of evolution (compared to 14 days in previous experiments) on the same compute cluster. Combined with faster evaluation times, the experiments here span roughly 300 times more evaluations than any previously reported Voxelbuild experiments.

Table 10.1: **Five treatments compared (two NSLC configurations and three controls).**

Treatment	Search Algorithm	Substrate	Number of Runs	Allotted Time
Shallow Substrate	NSLC	Shallow	12	162 days
Deep Substrate	NSLC	Deep	13	162 days
Fitness Only	Fitness-based	Shallow	5	50 days
Novelty Only	Novelty search	Shallow	5	50 days
Random Selection	Random selection	Shallow	5	50 days

Treatments Compared

The experiments in this chapter consist of five different configurations: two main configurations that differ only by the topology of the control network substrate and three controls that constitute an ablation experiment to show that both the quality and diversity components of NSLC are necessary to produce the results presented in this chapter. The two substrate configurations, previously described in Chapter 9, are repeated here in figures 10.1 and 10.2. A description of all five treatments is presented in table 10.1.

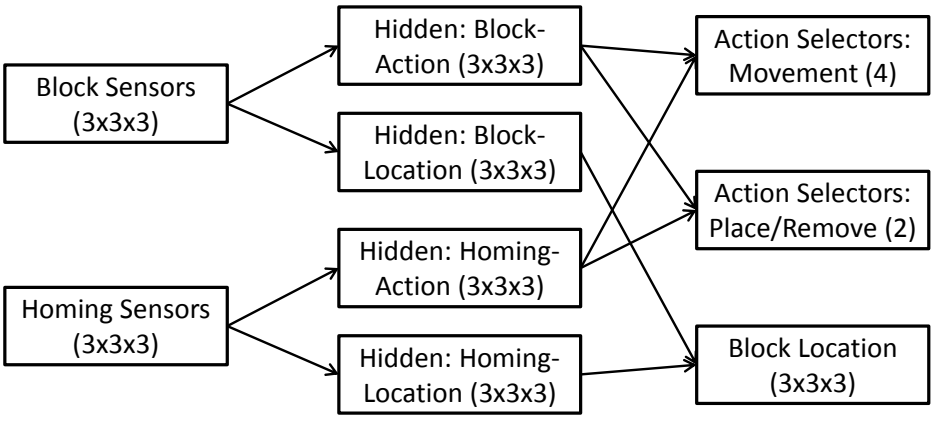


Figure 10.1: **Shallow substrate with homing sensor.** In this network configuration, agent sensors connect to a single hidden layer before proceeding to the outputs that determine action selection and, for the block placement and block removal actions, the location of the block to be placed or removed.

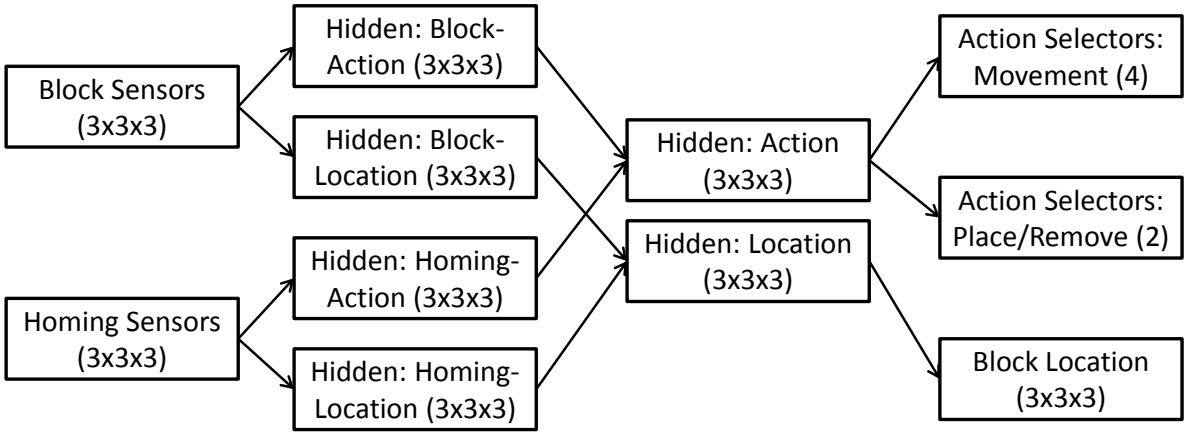


Figure 10.2: **Deep substrate with homing sensor.** In this network configuration, an additional hidden layer is placed between the first hidden layer and the outputs. While the first hidden layer processes block sensor and homing sensor information separately, the second hidden layer combines this information before it continues on to the outputs.

The **shallow substrate** configuration is easier to train and designed to explore the depth (in generations) of the space of possibilities in Voxelbuild while the **deep substrate** configuration is harder to train and slower to run (because it has a larger control network that takes longer to activate on each tick) but can potentially express more complex behaviors. By observing which substrate performs better (some amount of stagnation is expected), we can gain insight into the limits of this current iteration of the Voxelbuild platform and make more informed decisions about the most useful directions for future research. For example, stagnation and convergence (between runs) in the shallow substrate results would suggest that the open-ended potential of Voxelbuild is nearly exhausted. If stagnation is observed in the deep substrate at lower performance than attained by the shallow substrate, future research might aim to improve the evolutionary algorithm that drives search. If the deep substrate attains more advanced behaviors than any observed in the shallow substrate runs, that would suggest that open-endedness is limited by the agent's capacity to process information and therefore attention should be directed to expanding the structure of the control network. These types of insights are especially important in the field of open-ended evolution

(OEE) because the ultimate goal is to create systems that generate interesting results forever; thus, any advances in OEE research cause future generations of research in the field to consume progressively larger amounts of CPU time. When future OEE experiments continue to succeed even several years into their evolutionary timelines, it becomes vitally important to carefully determine where improvements should be made to the system.

The control runs in this chapter together represent an ablation experiment testing the necessity of both the quality and diversity components of NSLC to achieve the best possible results. If **fitness only** converges and stagnates at a low performing primitive behavior, we can conclude that Voxelbuild is a difficult task that requires novelty pressure to circumvent deception. If **novelty only** converges and stagnates at low-performing primitive behaviors, we can conclude that it is important to refine and perfect skills before they can successfully progress to higher orders of complexity. Finally, the **random selection** control in effect generates random yet increasingly varied weight patterns in the agent control network, establishing a baseline that illustrates how much success can be derived from the encoding alone without any intelligent training.

Experiment Parameters

For the long-term experiments in this chapter, QD is configured with parameters intended to allow evolution to continue indefinitely without breaking down or losing the most valuable discoveries. Table 10.2 contains parameter values shared by across all treatments. The fitness function and behavior characterization that drive the quality and diversity components of NSLC respectively are described in detail in the previous chapter. Evolution is steady-state, i.e. batches of 32 individuals are evaluated at a time and added to the comparatively larger population, archive, and QD collection according to the same replacement rules as the initial Voxelbuild experiments presented in Chapter 8. Replacement in the novelty archive and QD collection is governed by randomly-

Table 10.2: **Experiment parameters (shared by all treatments).**

Parameter	Value
Fitness function	$height^2$ (max 100)
Behavior characterization	Structural features
Population size	256
Archive size	1024
Novelty tournament size	8
QD collection size	128
QD collection tournament size	16
Evaluations between QD reports	320,000
Voxelbuild world size	$25 \times 25 \times 15$

drawn tournaments to save computation time at the expense of novelty score calculation accuracy. The imperfection of novelty calculations allows the occasional inclusion of less-novel individuals, preventing the archive and QD collection from stagnating over the course of evolution. As a result, QD collections generally contain individuals only from relatively recent generations. To ensure that interesting discoveries are not lost forever as evolution proceeds, the contents of the QD collection are periodically written to file. With this configuration, the longest run reported 790 QD collections (representing a total of 101,120 individuals saved to file in a single run). To illustrate the most advanced behaviors discovered in each run, the results presented in section 10 only contain individuals selected from the final QD collection (unless stated otherwise). However, the database of results generated by the experiments here is far richer than can feasibly be analyzed and reported by hand.

Results

This section presents results from the long-term Voxelbuild experiments described in the previous section. While the results here are reported after 162 days due to time constraints, the improved

Voxelbuild platform is designed to be able to continue indefinitely. Figures containing images of structures depict the final state of the world at the end of an agent's lifetime (generally thousands of ticks of a sensor-action loop, where one tick represents a movement or block placement action). Unless stated otherwise, such figures contain structures all selected from the same QD collection, where each collection contains a total of 128 individuals. Because of the tendency of tournament-style replacement to lose past discoveries over time, each QD collection consists entirely of individuals discovered relatively recently. Thus, the diversity observed in a single QD collection corresponds to the diversity of distinct structural styles actively being explored by NSLC at that point in the run.

Shallow Substrate

The evolutionary timelines revealed by the shallow substrate treatment explore more deeply into the space of possibilities in Voxelbuild than any other treatment. Figure 10.3 depicts the highest fitness found so far by each of the 12 shallow substrate runs over the course of evolution. Increases in fitness indicate that QD continues to generate progressively higher quality individuals even over very long timescales. The shallow substrate experiments in this chapter span an evolutionary timeline nearly 300 times longer on average than those in Chapter 8. That quality continues to increase even near the end of these runs suggests that quality diversity in Voxelbuild would continue to discover new and higher quality behaviors if evolution was allowed to continue even longer; this potential is especially evident in run 1 (which achieved the highest number of evaluations during the 162 days of allotted time) where fitness sharply increases during the final 2% of the run.

Figures 10.4 and 10.5 depict 28 structures selected from the final QD collection of run 2 after 185,280,000 evaluations. The structural diversity apparent in this collection demonstrates QD's ability to successfully explore the vast space of potential behaviors in Voxelbuild, pursuing many

different structural archetypes at the same time. While the structural diversity in this collection is substantial, most of the structures share a similar motif: "window-like" indentations on otherwise flat surfaces. This motif first appeared earlier in the evolutionary timeline and then became a core component of most of the structures discovered later in the timeline. This window motif corresponds to a specific type of building strategy that became a core component of all higher-order behaviors appearing later in the timeline.

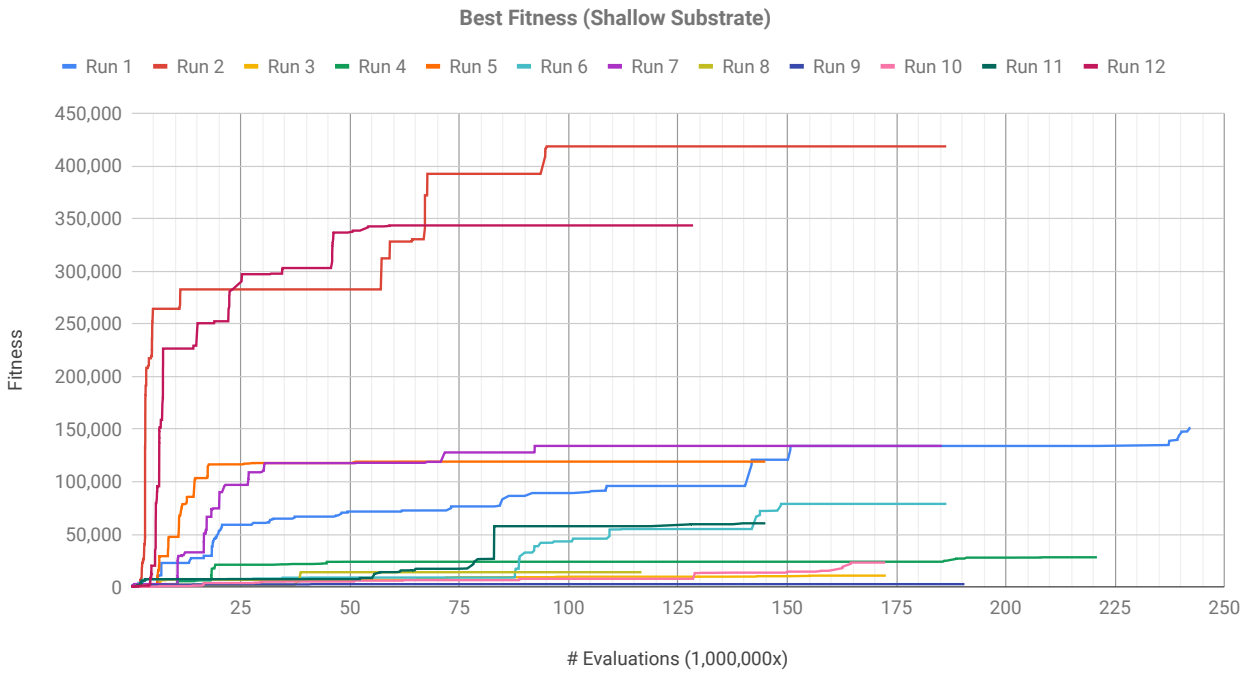


Figure 10.3: **Best fitness over time.** As evolution progresses, quality continues to increase even over very long timescales, suggesting that the potential for continued open-ended discovery in this iteration of the Voxelbuild platform has not yet been exhausted.

Figures 10.6 and 10.7 depict 24 structures selected from the final QD collection of run 12 after 128,320,000 evaluations. The structures depicted here, while similarly diverse, contain entirely different motifs than those found in run 2 (figures 10.4 and 10.5). The uniqueness of building styles between runs emphasizes that the space of possibilities in Voxelbuild is vast enough that evolution is able to follow a different trajectory in each timeline. The unique evolutionary trajectory

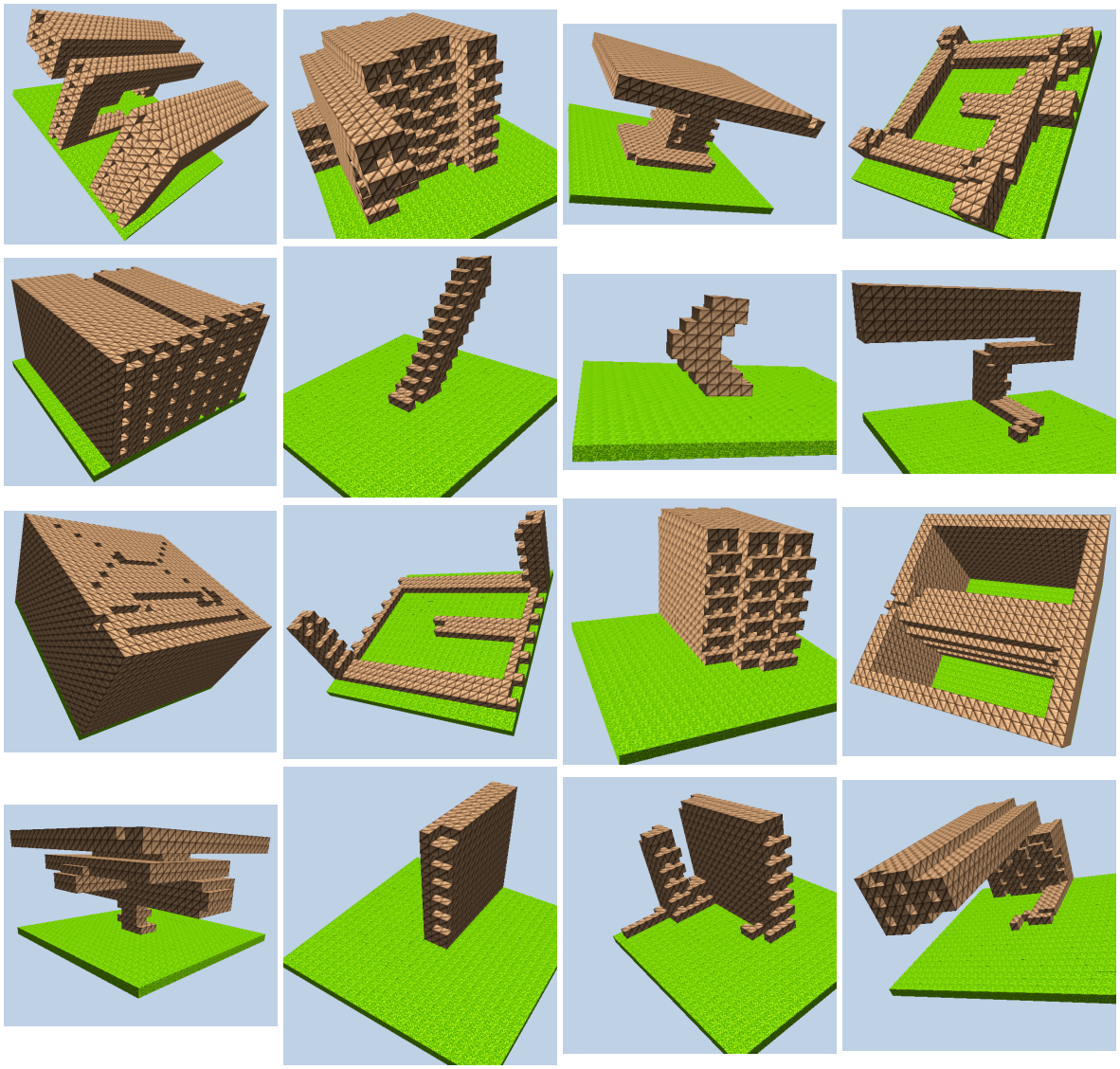


Figure 10.4: **Selected images from the end of run 2 (part 1).** Depicted are 16 out of a total of 28 structures selected from the final QD collection of *shallow substrate run 2*. The remaining 12 structures are depicted in figure 10.5.

in each run results from differences in random mutations that cause different skills to emerge early on that are later "locked in" once they are incorporated into higher-order, more advanced behaviors. This phenomenon, called *canalisation*, is also observed in natural evolution on Earth. In other words, QD continually acquires new skills and then combines them in different ways,

transitioning forward into more advanced behaviors as evolution progresses rather than getting stuck at a "richness ceiling" beyond which new discoveries can only be made laterally.

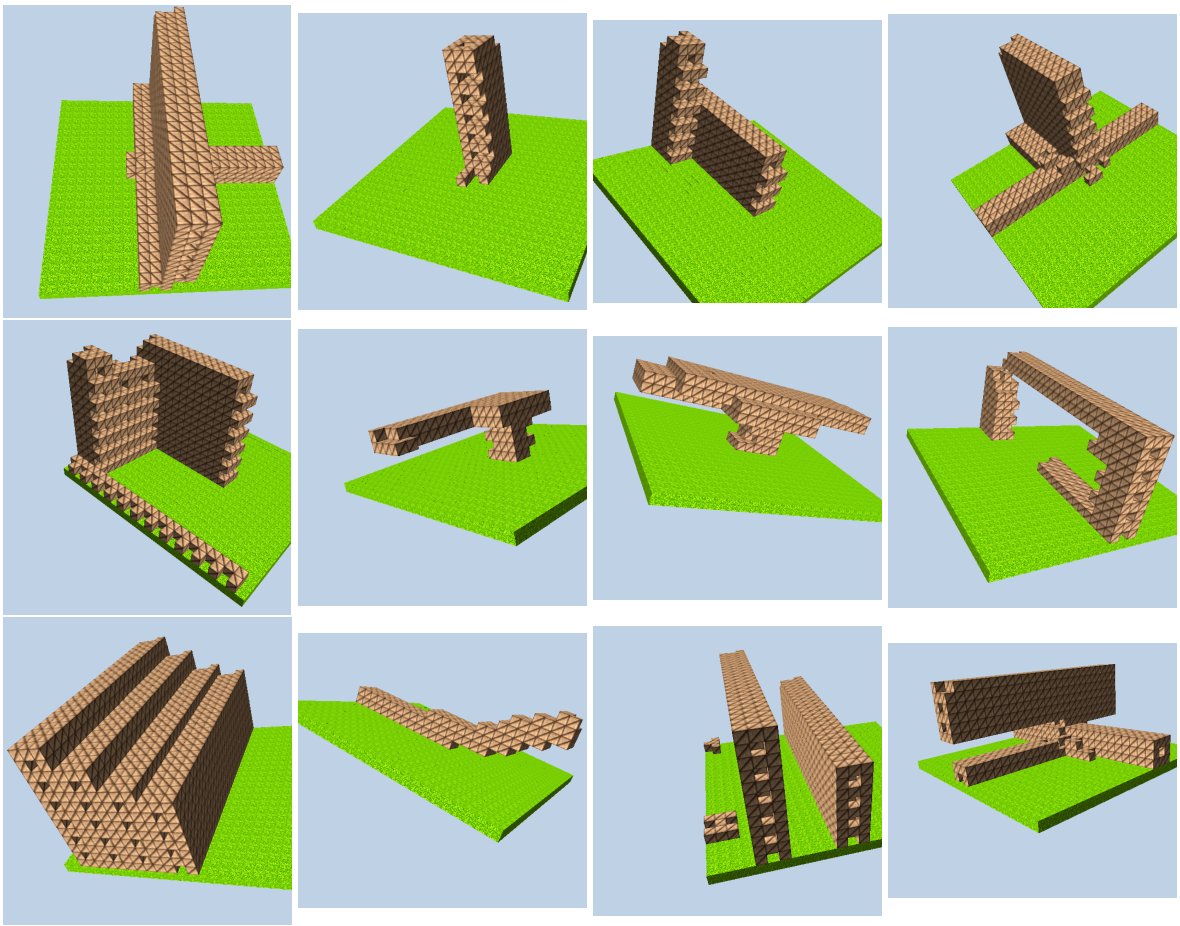


Figure 10.5: **Selected images from the end of run 2 (part 2).** Depicted are 12 out of a total of 28 structures selected from the final QD collection of *shallow substrate run 2*. This figure is a continuation of figure 10.4.

Figure 10.8 depicts a selection of 12 structures from the final QD collections of *shallow substrate runs 4 and 7* after 220,800,000 and 185,280,000 evaluations respectively. These structures showcase a common *scaffolding* strategy: "stair-building." The discovery of scaffolding represents a major evolutionary transition marking the appearance of a skill by which agents can move up z-levels by placing blocks and then moving on top of them through a coordinated sequence of

movement and placement actions. The first 6 images depict a "jagged" style of stairs appearing in run 4, distinct from the "smooth" stairs found in run 7. Stairs appear again in many different runs (but not all; some runs employ different types of scaffolding behaviors or even multiple types simultaneously). However, when the stair-like scaffolding technique appears, it generally has a distinct style unique to that particular run. That each run employs a unique style of the same type of behavior demonstrates that canalisation in Voxelbuild occurs even in subtle ways.

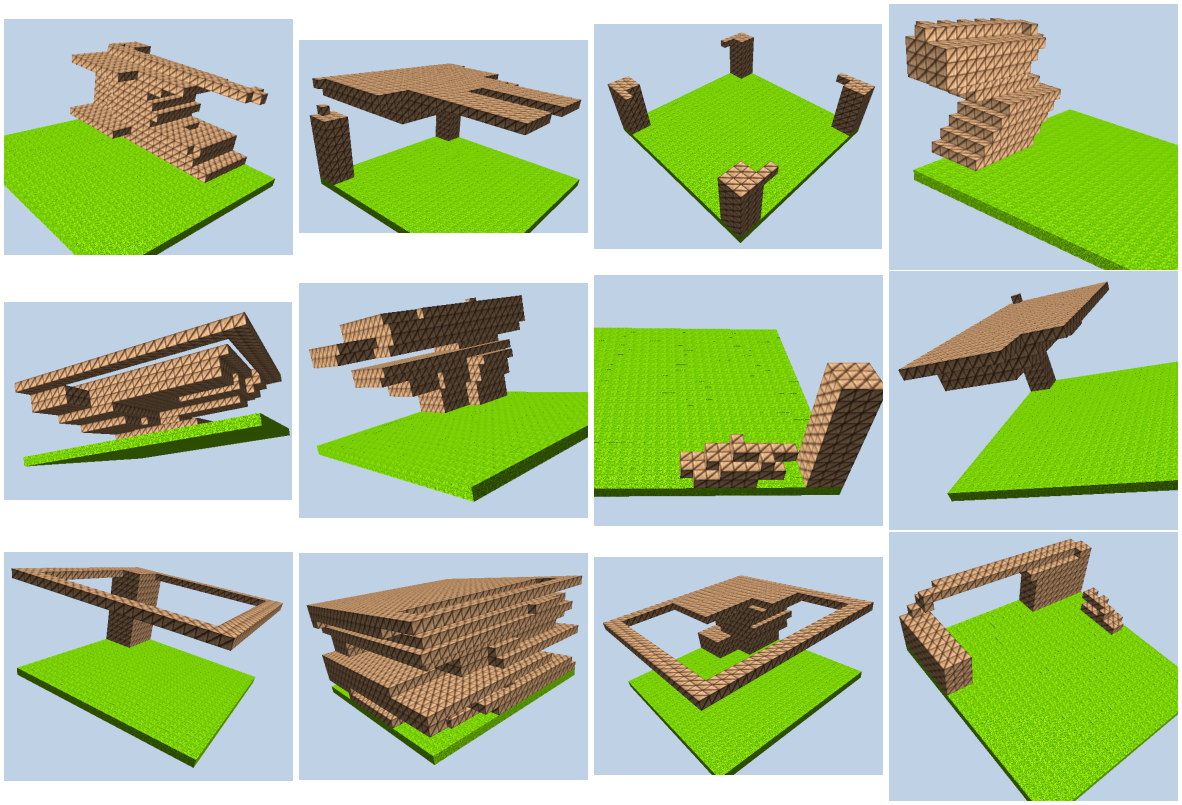


Figure 10.6: **Selected images from the end of run 12 (part 1).** Depicted are 12 out of a total of 24 structures selected from the final QD collection of *shallow substrate run 12*. The remaining 12 structures are depicted in figure 10.7.

Over the course of a single evolutionary timeline, new skills appear earlier in the timeline and are progressively refined and combined into more advanced higher-order behaviors later in the timeline. The gradual emergence of an advanced "tower altar" motif appearing in the final QD col-

lection of run 1 after 252,000,000 evaluations is illustrated by sampling the QD collection at various points along the evolutionary timeline of run 1. Figures 10.9 (16,000,000 evaluations), 10.10 (63,040,000 evaluations), 10.11 (126,400,000 evaluations), and 10.12 (252,000,000 evaluations) depict a series of images selected from each sampled collection, each showcasing the diversity of representations of the same motif simultaneously explored by QD as evolution progresses.

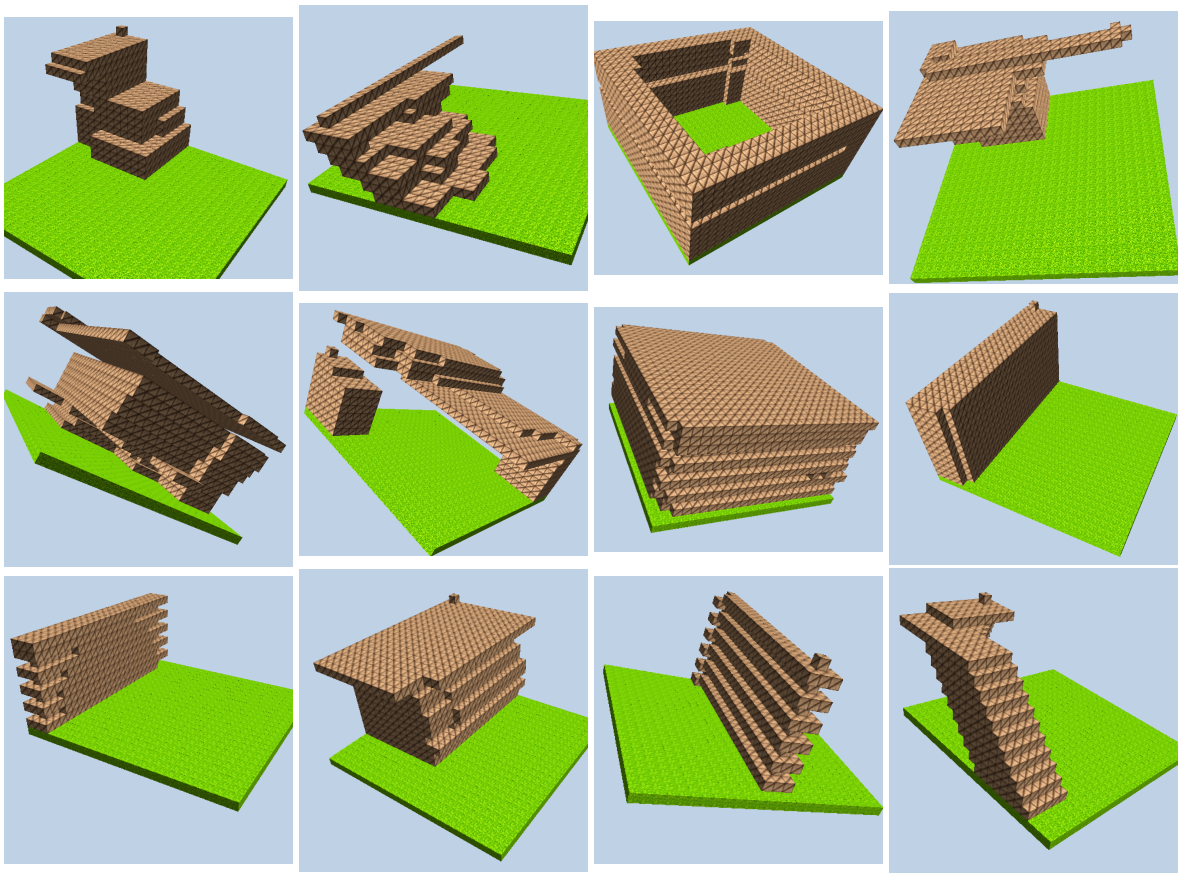


Figure 10.7: **Selected images from the end of run 12 (part 2).** Depicted are 12 out of a total of 24 structures selected from the final QD collection from *shallow substrate run 12*. This figure is a continuation of figure 10.6.

First, a "flat jagged tower" motif appears in the evolutionary timeline that ultimately becomes part of the set of skills required to construct the advanced "tower altar" structures explored by QD at the end of run 1. Figure 10.9 depicts a selection of 9 structures from the QD collection of run 1 after

16,000,000 evaluations exhibiting different versions of the flat jagged tower motif. All versions are one block wide and have jagged edges but vary in height and shape. In some cases, the tower appears multiple times.

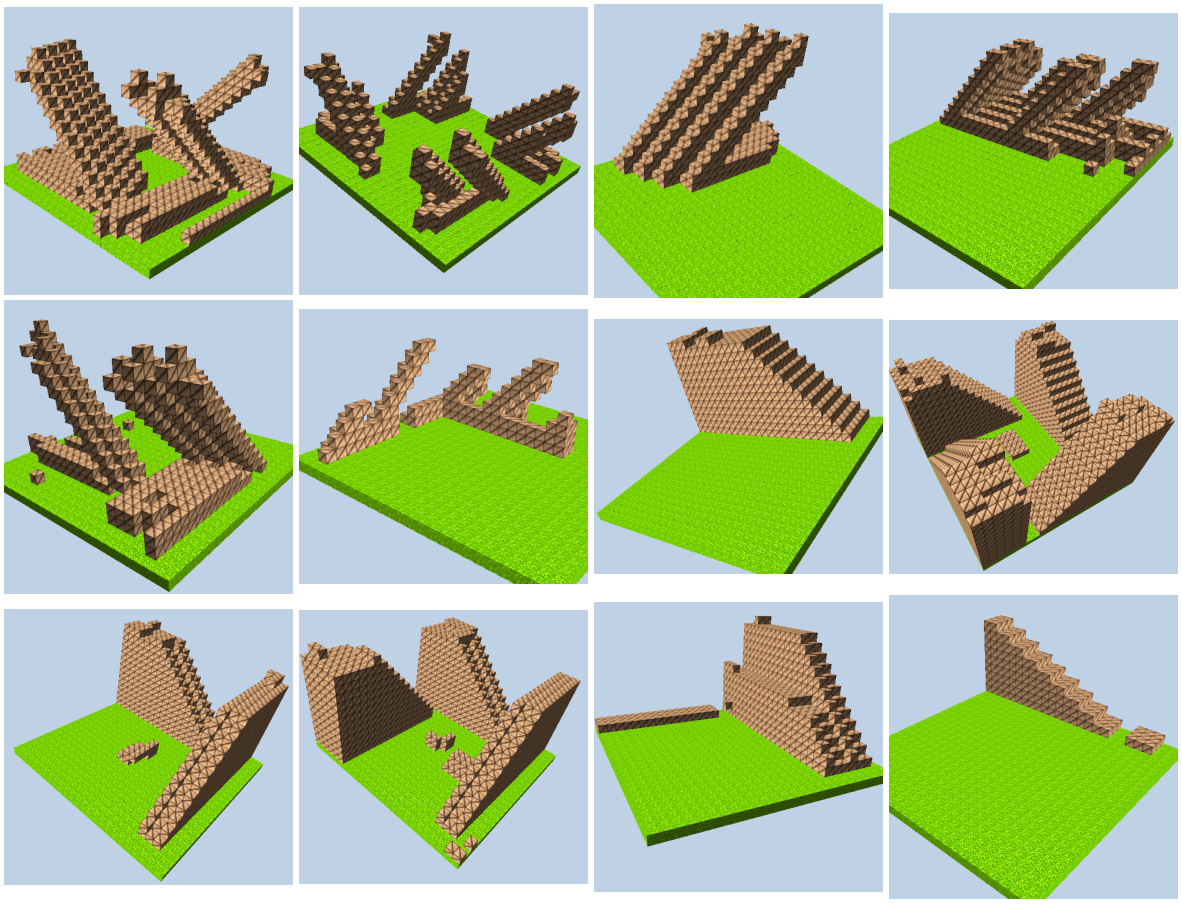


Figure 10.8: **Selected images from the end of runs 4 and 7.** Depicted are 12 structures selected from the final QD collections of *shallow substrate runs 4 and 7* illustrating a common "stair-like" scaffolding strategy.

Second, a new motif called "spaced flat sheets" appears in the evolutionary timeline that, combined with the earlier "flat jagged tower", eventually becomes part of the set of skills required to construct the "tower altar" structures appearing at the end of run 1. Figure 10.10 depicts a selection of 9 structures from the QD collection of run 1 after 63,040,000 evaluations exhibiting different

versions of the spaced flat sheets motif. These sheets always appear in groups or clusters along the edges of the world boundary and vary in terms of the number of groups (how many edges of the world boundary are blocked off) as well as the number of sheets in each group. The flat sheets are always one block wide and always appear with one or more blocks of empty space between them. This motif is likely an elaboration of the earlier "flat jagged tower" motif, where the outer edge of each sheet is flat due to the flat edge of the world boundary but the inner edge is often textured in more intricate ways than the original flat jagged towers discovered earlier. In general, the inner edge texture is often concave: wider at the top and bottom, sometimes appearing with long overhanging protrusions. At this point in the run, the types of sheets observed in the collection are messy and inconsistent in appearance even within a single image.

Third, the "spaced flat sheets" motif discovered earlier is refined, becoming smoother and more consistent in appearance. Figure 10.11 depicts a selection of 9 structures from the QD collection of run 1 after 126,400,000 evaluations (halfway through the evolutionary timeline) exhibiting different versions of the refined spaced flat sheets motif. The outer edges of each sheet are still flat against the world boundary, while the inner edges are both more consistent within each structure as well as more consistent between structures. The inner edges of each sheet are still concave but overhanging protrusions are now very rare. At this point in the timeline, the number of sheets in each structure has increased, often fully covering three or four edges of the world boundary. Additionally, the spacing between each sheet is now consistently one block wide.

The first image in figure 10.11 depicts an early appearance of the advanced "tower altar" motif, which combines the "flat jagged tower" and "spaced flat sheets" motifs, forming a ring of flat sheets around a single jagged tower placed in the center of the map. Constructing a tower altar structure requires agents to combine the comparatively more primitive skills required to build both the spaced flat sheets and jagged tower components. As this point in the run, the tower altar rarely appears in the QD collection; most versions of the encircling ring of flat sheets have an empty

space in the middle of the world or else only the beginnings of the jagged tower component one or two blocks in height.

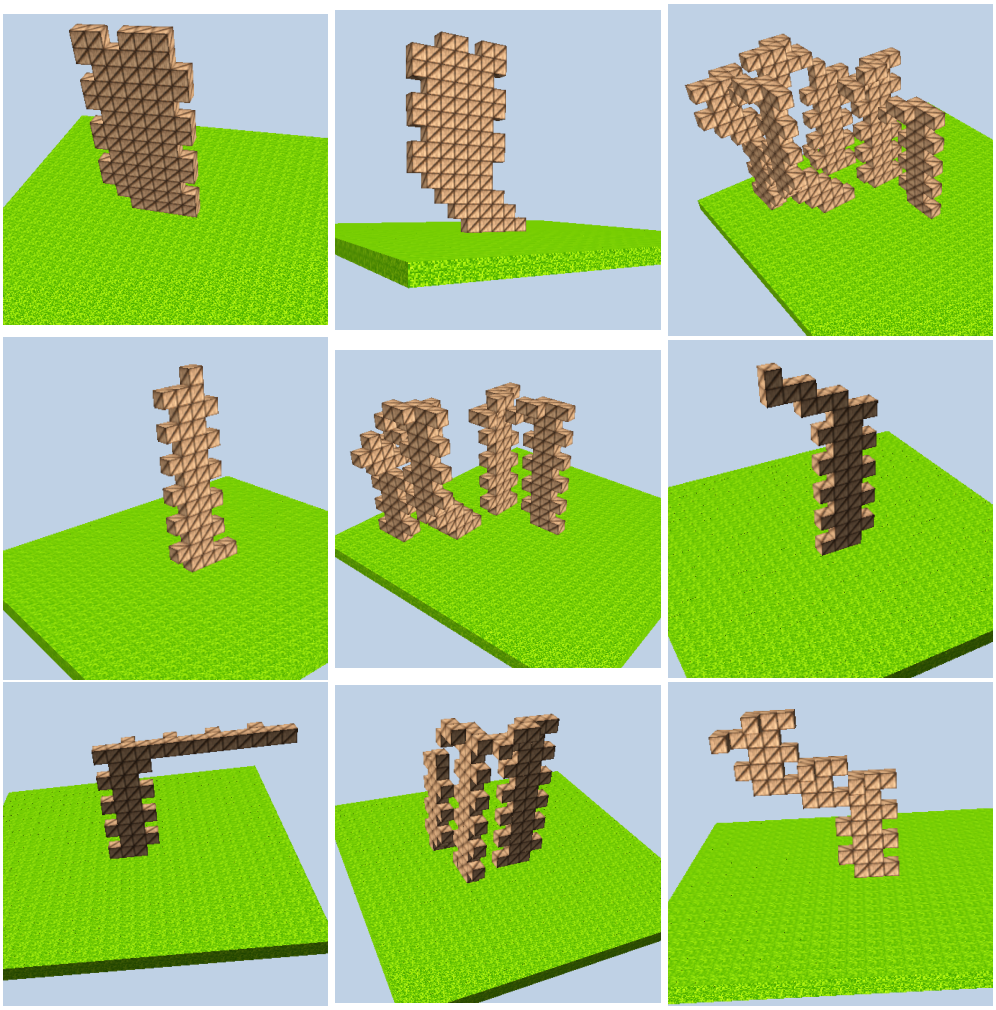


Figure 10.9: **Evolution of the "tower altar" in run 1: appearance of the "flat jagged tower".** Depicted is a selection of 9 structures from the QD collection of run 1 after 16,000,000 evaluations that display a diversity of flat tower-like objects constructed by Voxelbuild agents.

By the end of the evolutionary timeline observed throughout run 1, agents have learned to reliably combine the "flat jagged tower" motif with the "spaced flat sheets" motif after discovering and refining both skills earlier in the run, resulting in the new, higher-order "tower altar" motif. QD then successfully produces a diversity of tower altars following its initial discovery. Figure 10.12

depicts a selection of 12 different tower altar structures from the QD collection of at the end of run 1 after a total of 252,800,000 evaluations. In all variants, a number of flat sheets face inwards towards a single flat jagged tower. In this collection of tower altars, the flat jagged towers at the center of the map form consistent horizontal patterns, although the towers themselves vary in height. The spaced flat sheets surrounding the tower consistently form a distinct concave shape. The main source of diversity between tower altars is the number and placement of spaced flat sheets around the flat jagged tower in the center.

The consistency in appearance of the flat jagged towers and spaced flat sheets motifs across all variants of the tower altar is evidence of their canalisation in this evolutionary timeline. In other words, the agents capable of producing such a variety of tower altars descended from a single common ancestor with a robust enough representation of the lower-order flat jagged tower and spaced flat sheet skills within its control network to be able to successfully produce a diversity of distinct tower altar styles without disrupting the constituent skills and thus the general shape of each tower and sheet is the same in all variants. The appearance of the tower and sheet motifs are now difficult to change because they all share the same robust representation present in their common ancestor.

While agents in the evolutionary timeline of run 1 progressively developed skills ultimately culminating in the advanced "tower altar" motif, QD also explored a number of other distinct structural motifs throughout the run at the same time. Figure 10.13 depicts a selection of 16 structures sampled from the same four QD collections as the tower altar progression reported above. Early structural motifs such as "checkerboard patterns" likely influenced the textures present in more advanced structures later in the timeline. The final collection contains several unique patterns of "stacked horizontal sheets" similar to the vertical sheets that appear throughout the entire tower altar progression. The theme of "minimal ground support" appears throughout the entire timeline, consisting of increasingly long chains of single-block-wide spiral-like scaffolding that eventually

support much larger structures floating several z-levels above the ground by the end of the run. The minimal ground support style of scaffolding later developed into "boat-like objects" that first appeared in the QD collection after 63,040,000 evaluations. In the collections after 126,400,000 evaluations and 252,800,000 evaluations, the boat-like objects appear stacked in various ways, including a thicker style of spiral-like scaffolding. The final image in figure 10.13 represents perhaps the most impressive discovery in this timeline, combining nearly all of the skills outlined here into an advanced "floating stadium" structure; when viewed from below, the stadium is actually floating several z-levels off the ground, supported only by a single block wide chain of scaffolding.

The diversity of structural motifs discovered, subsequently refined, and ultimately combined into higher-order behaviors throughout the evolutionary timeline of a single run (run 1) provides clear evidence that QD in Voxelbuild is able to simultaneously explore several distinct evolutionary trajectories at the same time over millions of evaluations, similarly to how natural evolution on Earth supports an ever-widening tree of life containing countless independently developing species.

Genome Complexity

Figure 10.14 depicts the average number of connections in the CPPN genomes in the population at the end of each shallow substrate run. As evolution in Voxelbuild continues over a long period of time, the number of connections in the genome (CPPN) steadily increases (in a roughly linear fashion). While genomes begin with minimal complexity (no hidden nodes and only sparse connectivity between the inputs and outputs), by the end of each run the genome size has increased to around 1,000 to 2,000 connections on average. These genome sizes are among the largest NEAT genomes ever reported known to the author, hinting at the ability of such systems to sustain continued innovation even with steadily increasing genome size, hinting at a genuine trend of increasing complexity as evolution progresses. In fact, the indirect HyperNEAT encoding may even be an im-

portant source of canalisation in Voxelbuild because the underlying CPPN genome can complexify in a way that that makes critical parts of the control network difficult to change.

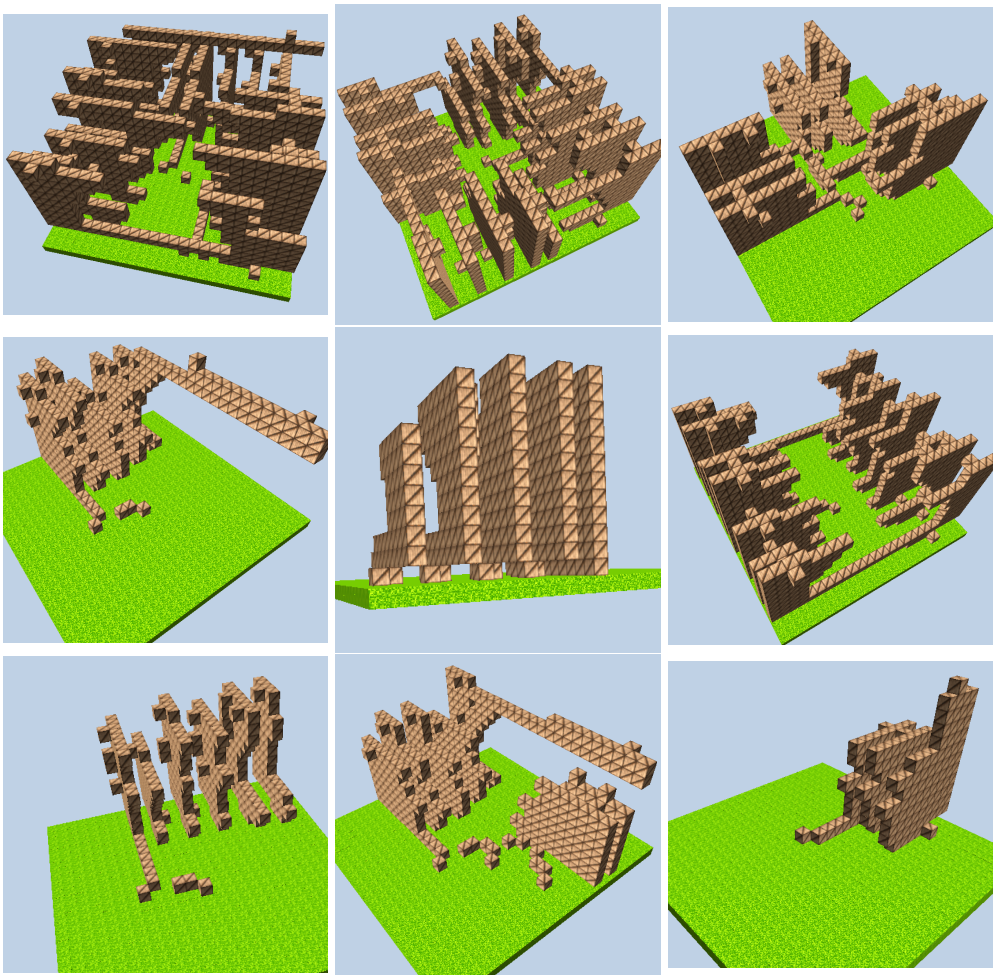


Figure 10.10: **Evolution of the "tower altar" in run 1: appearance of "spaced flat sheets".** Depicted is a selection of 9 structures from the QD collection of run 1 after 63,040,000 evaluations that display a diversity of parallel vertical sheets constructed by Voxelbuild agents.

Deep Substrate

The deep substrate configuration allows agent control networks to process information differently and potentially express more complex behaviors than those possible with the shallow substrate

by encoding a greater variety of lower-order skills to be combined into higher-order behaviors. However, the extra network structure comes at the expense of longer evaluation times because the control network takes longer to activate on each tick. Additionally, deeper networks are harder to train because there are more connections that must be trained between the sensors and action selectors as well as more opportunities for destructive mutations to occur.

Figure 10.15 depicts the highest fitness found so far by each of the 13 deep substrate runs over the course of evolution. Increases in fitness indicate that QD is continuing to generate progressively higher quality individuals even over very long timescales. Compared to the shallow substrate runs presented in the previous section, agents evolved with the deep substrate control network did not achieve fitness increases as often as those with a shallow network, confirming that the deep substrate is indeed harder to train.

No deep substrate runs achieved "very high scores" (e.g. over 300,000 fitness) while two shallow substrate runs achieved such scores. However, with the fitness function in this iteration of Voxelbuild, the highest scores do not necessarily correspond to "better" results. The highest scoring individuals discovered in two shallow substrate runs contain a number of very large and dense structures that extend to all world boundaries (including the upper boundary in the z-direction). Building such structures requires long sequences of straight and well-organized block placement patterns that do not necessarily correspond to more complex and interesting behaviors. In fact, structures that densely fill up the entire world with blocks cannot achieve as much diversity as those composed of intricate patterns such as those from deep substrate run 7 depicted in figure 10.16 (133,760,000 evaluations). Thus a trade-off is evident between intricacy (which supports diversity) and fitness.

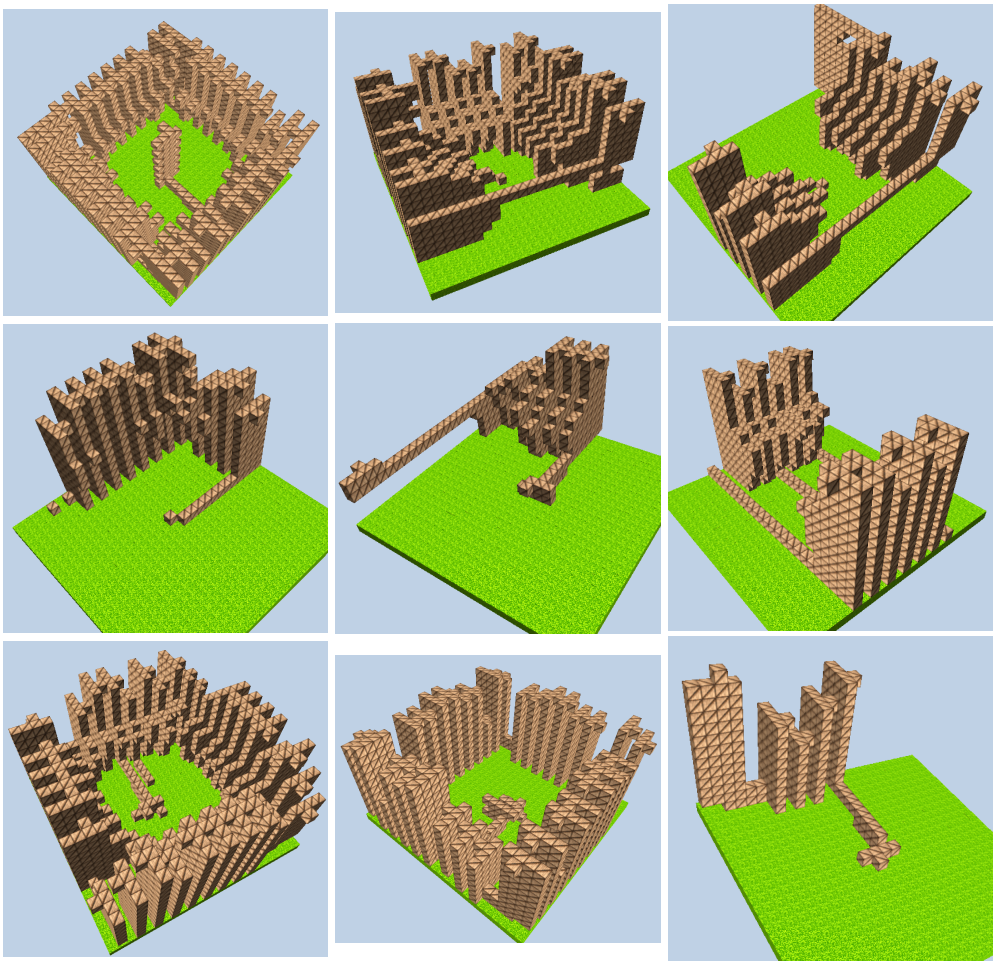


Figure 10.11: **Evolution of the "tower altar" motif in run 1: refined "spaced flat sheets" and first appearance of the "tower altar".** Depicted is a selection of 9 structures from the QD collection of run 1 after 126,400,000 evaluations showing a diversity of parallel vertical sheet arrangements.

Genome Complexity

Figure 10.17 depicts the average number of connections in the genomes (CPPN) in the population at the end of each deep substrate run. Just as in the shallow substrate experiment, the number of connections in the genome (CPPN) steadily increases in a roughly linear fashion over the course of evolution. Deep substrate CPPNs contain a slightly lower number of connections than those in the

shallow substrate runs perhaps due to the increased difficulty of making meaningful improvements to the more complex substrate.

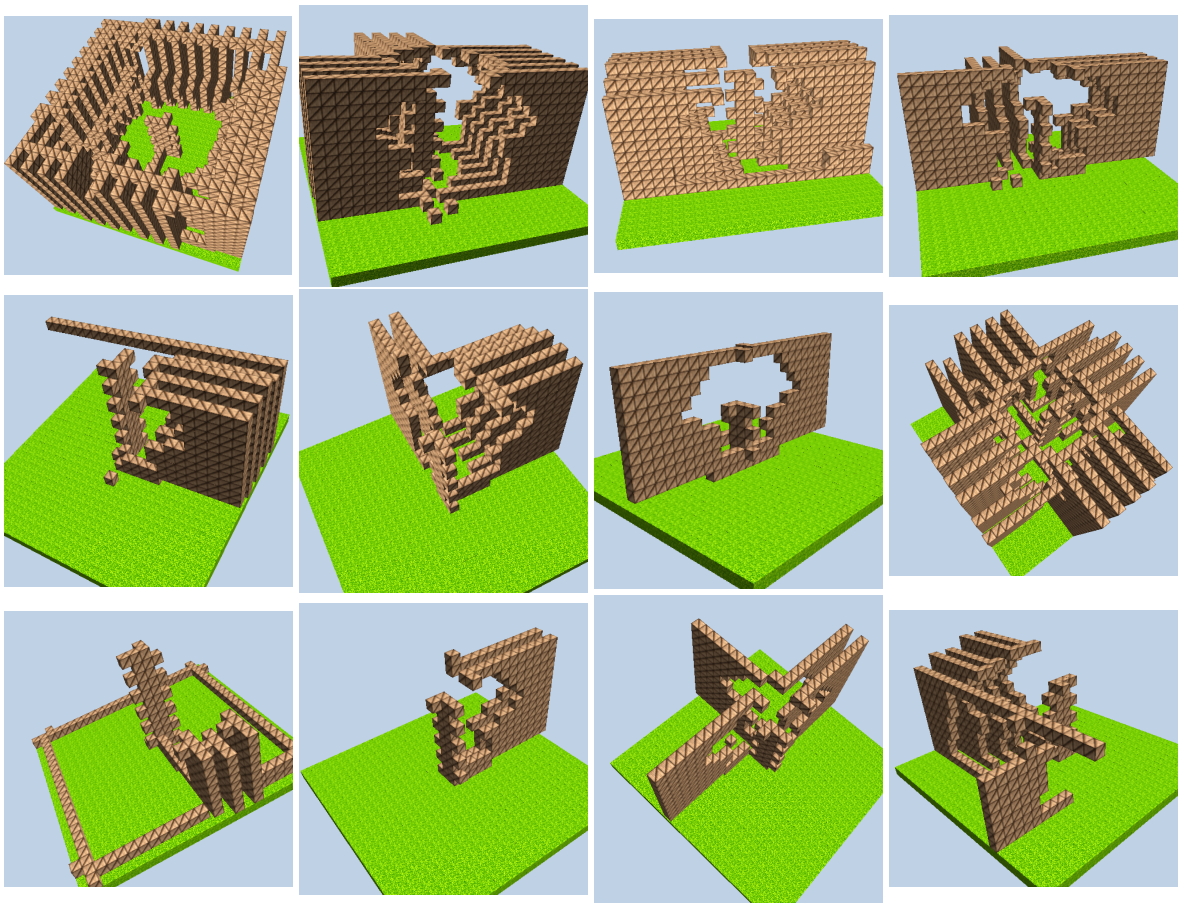


Figure 10.12: **Evolution of the "tower altar" motif in run 1: diversity of "tower altar" structures.** Depicted is a selection of 12 structures from the final QD collection of run 1 after 252,800,000 evaluations. A diverse array of tower altar variants all contain similarly-shaped towers and sheets.

Control: Fitness Only

In this control, the novelty component of NSLC has been removed and search is driven only by *global* fitness pressure, i.e. a basic traditional evolutionary algorithm. Figure 10.18 depicts the

highest fitness achieved by each of the five runs driven by fitness search. Performance in each run improves at first but then stagnates indefinitely at a much lower than QD-driven runs due to the well known problem called *deception* where fitness pressure inevitably drives search into an evolutionary dead end from which no further improvement is possible without pursuing a long series of lower-performing stepping stones. Specifically, fitness-only search in Voxelbuild is unable to progress beyond structures more than two blocks in height. Figure 10.19 depicts a selection of structures achieved by this control.

Control: Novelty Only

To observe how successfully Voxelbuild can be explored by novelty alone drives evolution, the local competition component of NSLC is removed and search is driven by novelty alone, i.e. standard novelty search. Figure 10.20 depicts the highest fitness achieved by each of the five runs driven by novelty search. Performance in four out of five runs stagnates at relatively trivial behaviors because the space of possible behaviors in Voxelbuild is vast enough that novelty search can continue to find new structures even at very long timescales without ever progressing to those requiring more advanced skills. In this way, novelty search in Voxelbuild acts as a sort of jack-of-all-trades that continuously learns new skills without practicing any of them. Some runs discover a stair-like or tower-like scaffolding strategy. Run 3 was able to achieve relatively high scores by stumbling upon a wide version of a stair-like scaffolding strategy. Figure 10.21 depicts a selection of structures achieved by this control across all five runs. Within each run, structural diversity is noticeably higher than in the fitness-driven control but structures never advance beyond the initial scaffolding pattern due to the lack of fitness pressure.

Control: Random Selection

In this control, search is driven by purely random selection. The purpose of this control is to demonstrate the best behaviors that can be achieved by the encoding alone without any intelligent training of the connection weights in the substrate. In all cases, fitness is unable to progress beyond 256 even over an extremely long timescale (figure 10.22). Figure 10.23 depicts a selection of the best behaviors achieved by essentially random control networks.

Discussion

The interpretation of a system like Voxelbuild is clearly challenging. Both qualitative and quantitative factors enter into our understanding of its capabilities and dynamics, and many further questions will likely be asked. Nevertheless, it is still appropriate to highlight that the long-term QD evolution experiments with Voxelbuild in this chapter are highly unique. The author knows of no precedent of an artificial evolutionary system demonstrating continued innovation for hundreds of millions of evaluations. Furthermore, the results presented here suggest that innovation driven by QD in Voxelbuild may have continued had the experiment been allowed to run even longer. Not only are these among the longest QD experiments ever run, but they are the first even to hint at the idea that QD can sustain continued discovery for such an unprecedented duration. Though it is true that the nature of innovation in these experiments is still open to scrutiny, continual qualitative changes coupled with increasing fitness are salient evidence of a novel long-term dynamic. They are also fuel for further research and discovery in the field in the future with such massive run lengths. That the controls also fail to exhibit the same dynamics further hints at the unique capacity of QD to sustain such a process in perpetuity, or at least for a very long time.

Successfully applying QD to open-ended creative tasks (such as Voxelbuild) with a vast or even

endless space of possibilities inevitably generates an enormous amount of results as evolution progresses over very long timescales. Ultimately the goal of such systems is to continue to generate interesting new artifacts forever, effectively producing an endless amount of results. To view, appreciate, and make use of the discoveries generated by QD here, as well as in other open-ended creative tasks in the future, it is critically important to design result-filtering mechanisms that carefully select only discoveries worth looking at to make analysis tractable. In many domains it may be difficult to design such result-filtering mechanisms a priori because interesting artifacts will likely appear during the run that vary in ways that are not specifically selected for by the filtering mechanism designed before the run began.

Furthermore, and perhaps more importantly, our concept of "interestingness" is subjective, hard to describe, and differs depending on who is viewing or using the results. In the iteration of the Voxelbuild platform described in this chapter specifically, the structural features (such as height, width, and density) used to identify which results should be reported to the user were not on their own predictive of which results were actually the most interesting to look at. The most interesting results in Voxelbuild include structural motifs such as checkerboard patterns, spirals, "stadiums," and other objects resembling real-life artifacts (particularly interesting is that intelligent agents here are able to construct such human-recognizable structures even with no concept of them). Although the QD collection mechanism employed here to filter results could be improved, perhaps a better solution in future work is to *crowdsource* the task of sorting through artifacts generated by such an open-ended system.

The results produced by the long-running Voxelbuild experiments here not only exhibit continuous generation of novel behaviors, but also progressively higher-order more complex behaviors by combining and building on lower-order constituent skills discovered earlier in each evolutionary timeline. With each transition into higher-order behaviors, it potentially becomes more difficult to achieve further transitions because doing so requires targeting control network perturbations

caused by random mutation to parts of the control network in a way that does not disrupt lower-order constituent skills. Depending on the encoding employed by such a system, as artifacts transition into more advanced behaviors the genome may become increasingly brittle if the opportunity for destructive mutations does indeed increase. Thus, in designing such systems in future work, it may be important to consider if the encoding and underlying evolutionary process is sufficiently able to protect against such brittleness on its own (perhaps by incorporating specific canalisation mechanisms observed in natural biology). While the results here demonstrate canalisation at several levels of granularity (preserving small distinct textural patterns as well as preserving critical building block skills required to transition into more advanced behaviors), achieving truly endless open-ended discovery may require specific improvements to the encoding and underlying evolutionary algorithm. In this iteration of Voxelbuild, evolutionary canalisation may be limited by the limited structure available in the predefined HyperNEAT substrates, ultimately restricting the ability of agent control networks to contain robust representations of skills essential to further development. However, as the comparison between the shallow and deep substrates demonstrate, networks with more structure are harder to train, so future work may try to address this problem through a dynamically-expanding network structure or through improvements to the training algorithm allowing networks with larger structure to efficiently improve over time.

While the results presented in this chapter continue to achieve open-ended discovery even over very long timescales confirming that Voxelbuild contains a rich space of possibilities, it can easily be expanded by increasing the size of the world (perhaps removing world boundaries altogether) or adding more ways to agents to sense and interact with the world (including the introduction of multiple agents at the same time). The success of the experiments here confirm that QD can act as a powerful driver for open-ended evolution, suggesting that it may also perform effectively in other OEE worlds, perhaps uncovering new discoveries that have yet to be achieved in those domains.

Conclusion

This chapter contained experiments applying QD to an improved iteration of the Voxelbuild platform where evolution is allowed to continue for an unprecedented number of generations, orders of magnitude longer than the initial Voxelbuild experiments in Chapter 8. As demonstrated by the results, agents continue to develop more advanced behaviors and new structural motifs even up to the end of each run, suggesting that Voxelbuild may contain even more potential for open-ended discovery, representing an important step towards OEE's ultimate goal of systems that endlessly generate not only novel but fundamentally more advanced behaviors similar to those observed in natural evolution on Earth.

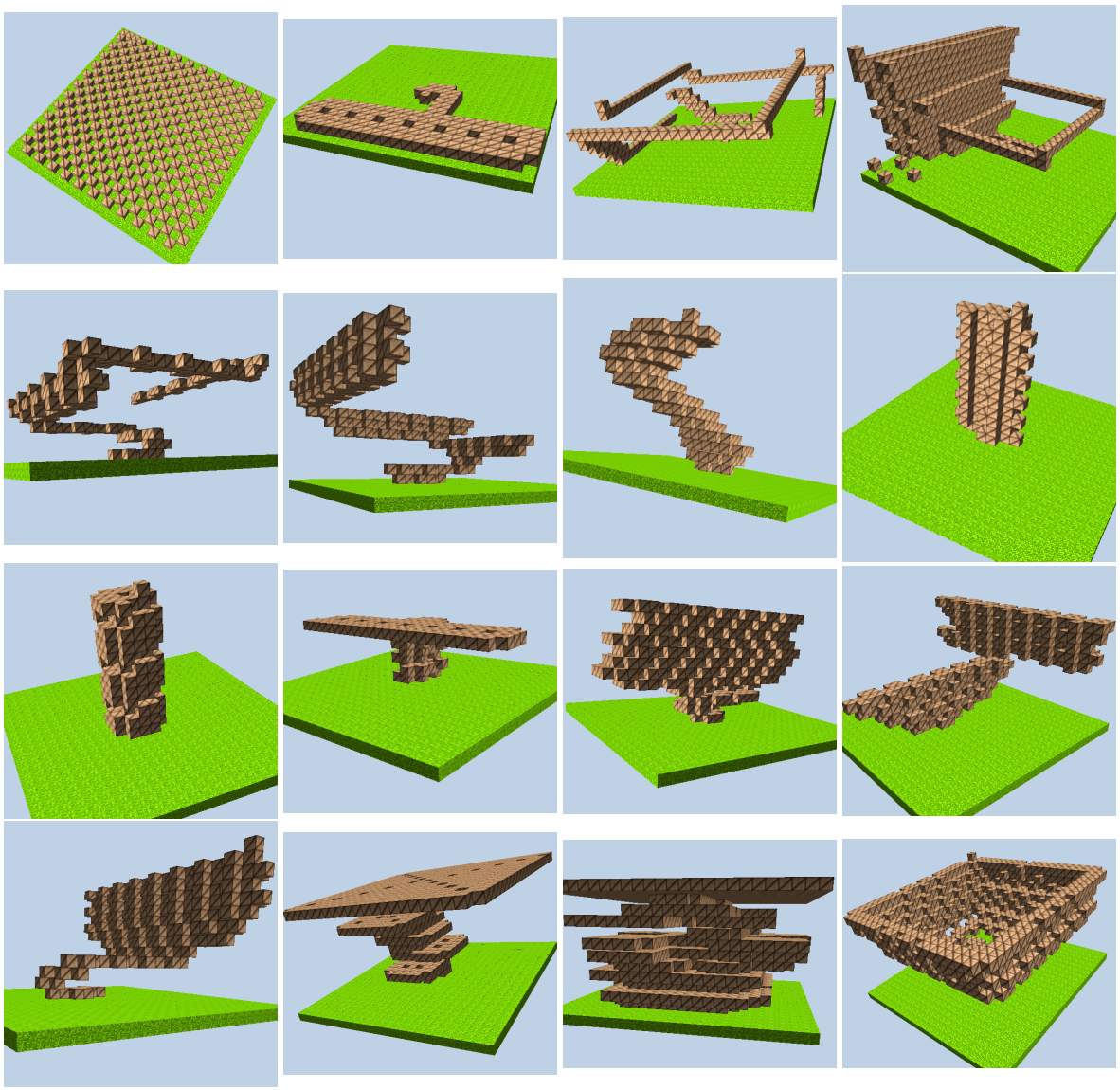


Figure 10.13: **Other structural motifs expressed in run 1.** Depicted is a selection of 16 images sampled from the same QD collections as the tower altar progression: after 16,000,000 evaluations, after 63,040,000 evaluations, after 126,400,000 evaluations, and after 252,000,000 evaluations.

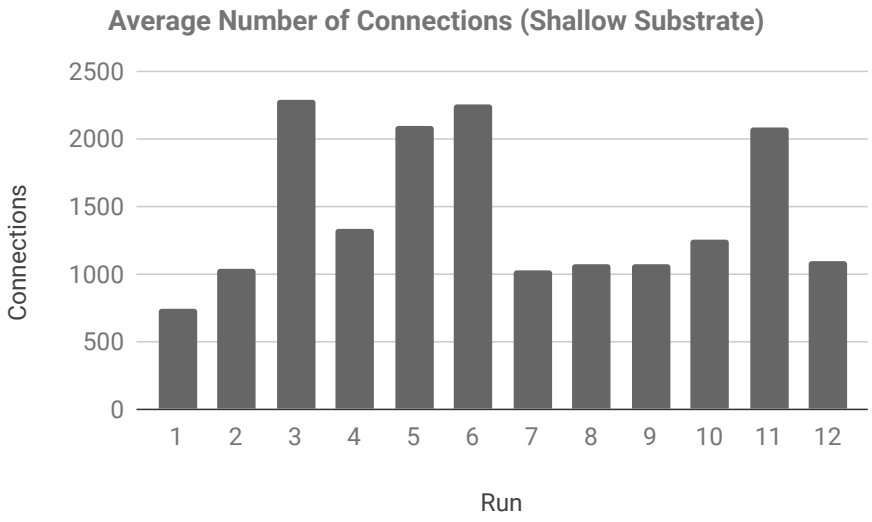


Figure 10.14: **Average genome (CPPN) complexity at the end of each run.** Depicted is the average number of connections in genomes in the population at the end of each run.

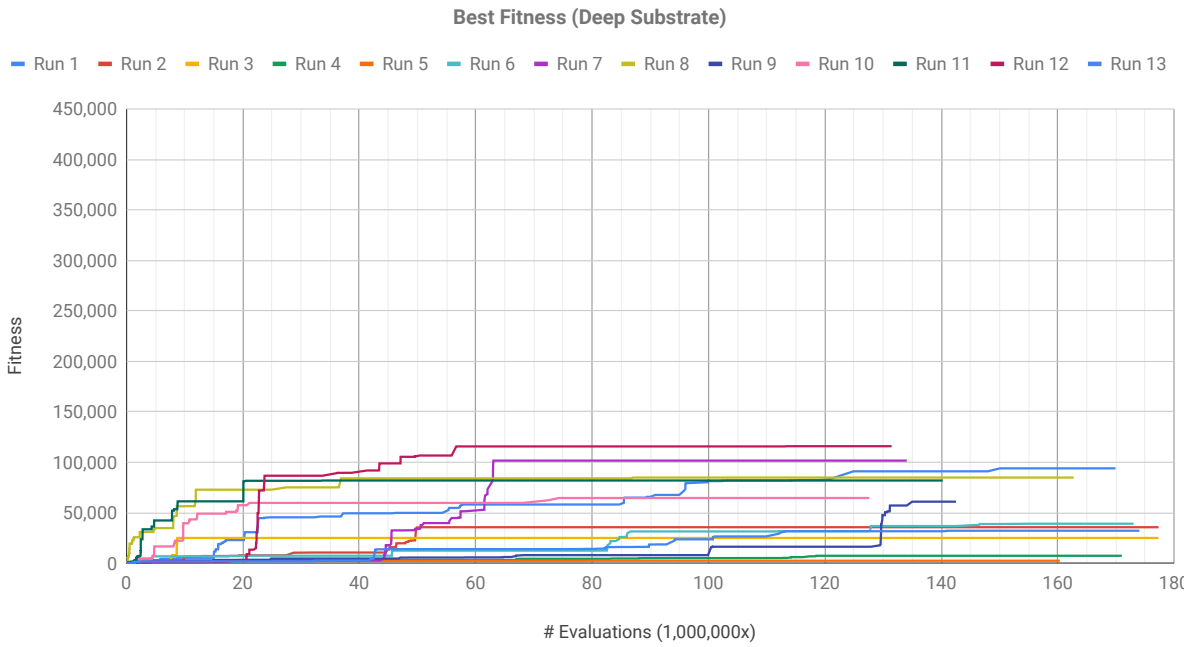


Figure 10.15: **Best fitness over time.** As evolution progresses, quality continues to increase even over very long timescales, suggesting that the potential for continued open-ended discovery in this iteration of the Voxelbuild platform has not yet been exhausted.

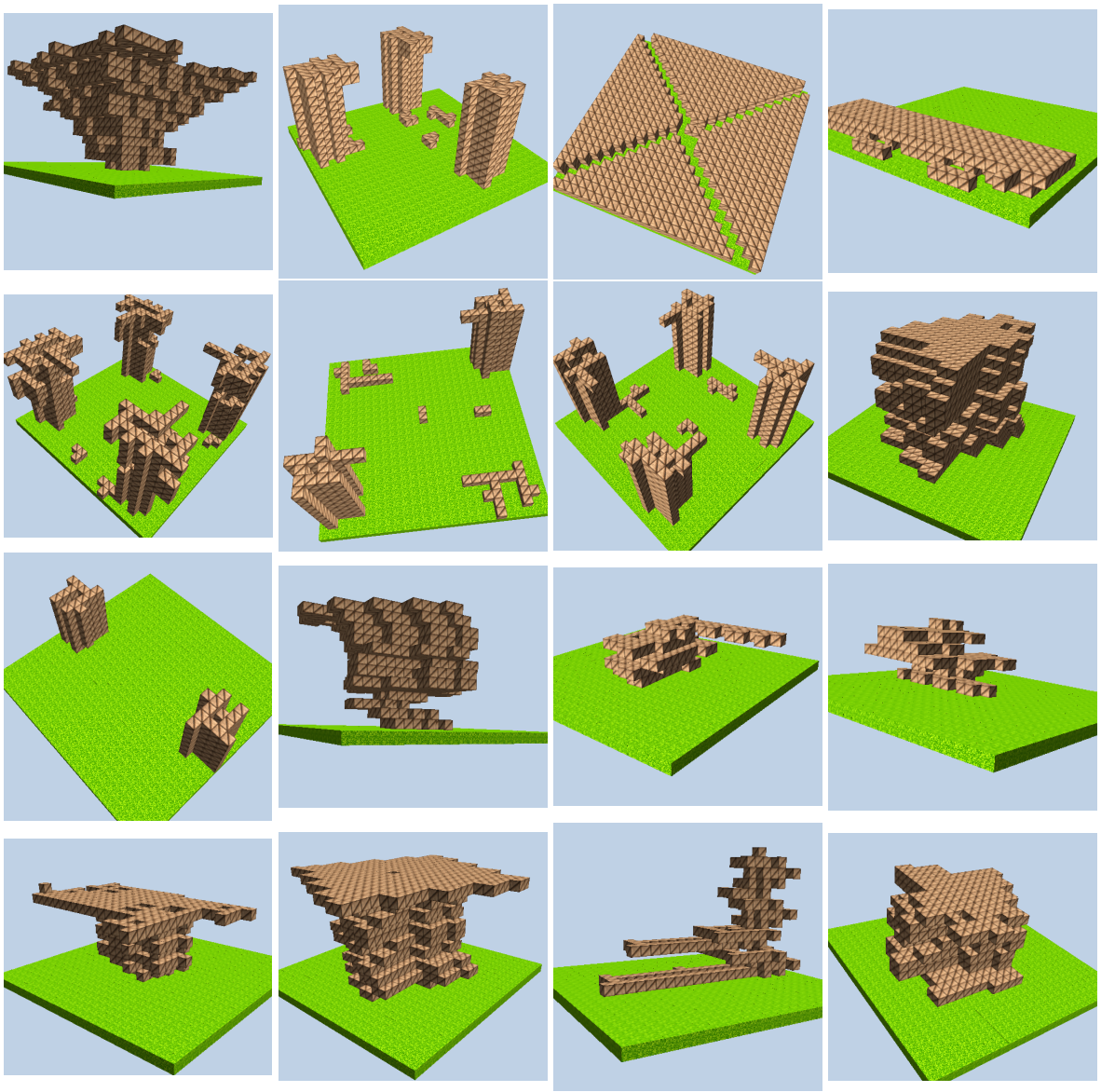


Figure 10.16: **Selected images from the end of run 7.** Depicted are 16 structures selected from the final QD collection of *deep substrate run 7* after 133,760,000 evaluations. The types of structures discovered by this run are visually more intricate than any of those found in the shallow substrate runs. This intricacy may be made possible by the additional hidden layer that combines information from the block location sensors before proceeding to the outputs.

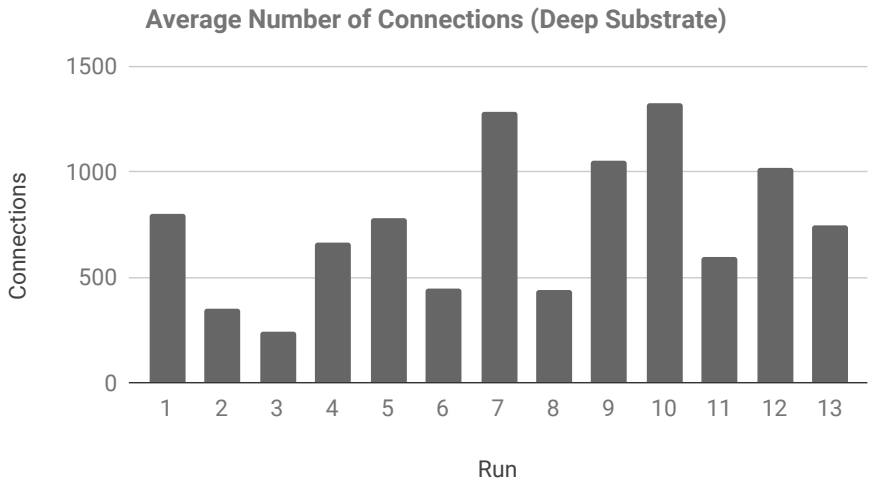


Figure 10.17: **Average genome (CPPN) complexity at the end of each run.** Depicted is the average number of connections in genomes in the population at the end of each run.

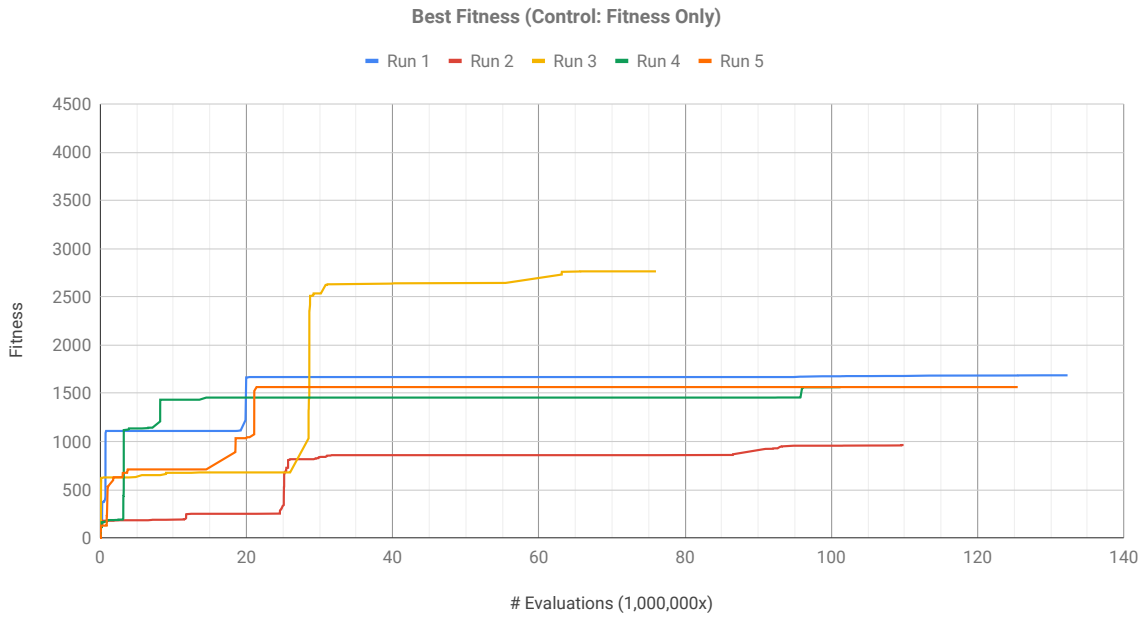


Figure 10.18: **Fitness-only control best fitness over time.** When fitness alone drives progress in Voxelbuild, the ability of evolution to advance is severely stunted. Top performance levels in this graph are orders of magnitude below QD-based runs.

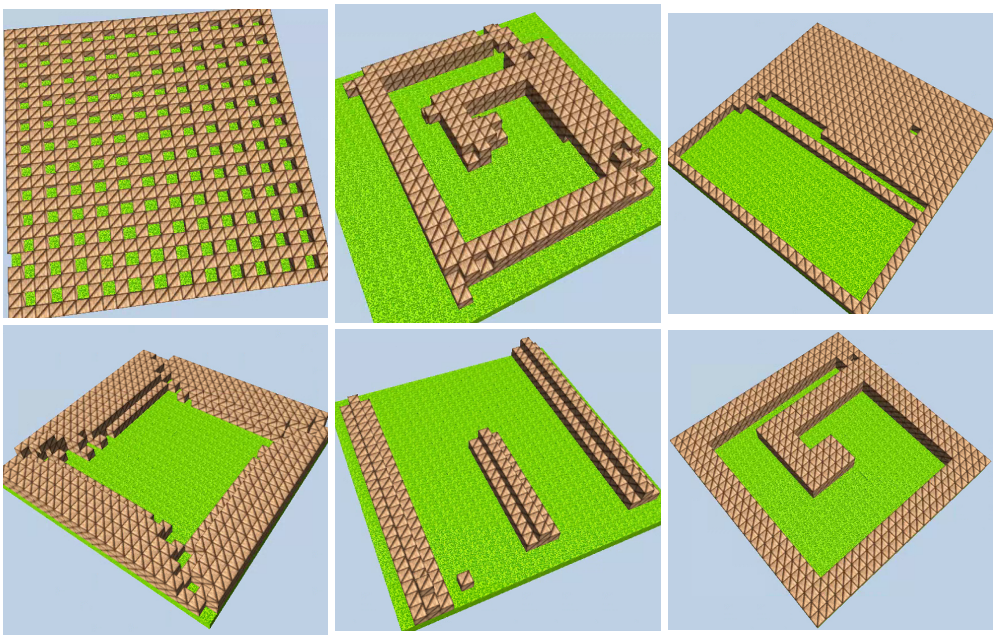


Figure 10.19: **Fitness-only control selected images.** This selection represents some of the most interesting structures produced across all five fitness-driven control runs. Agents never progress beyond building structures more than two blocks tall. Within each run, structural diversity is noticeably lower than runs that include a drive for novelty.

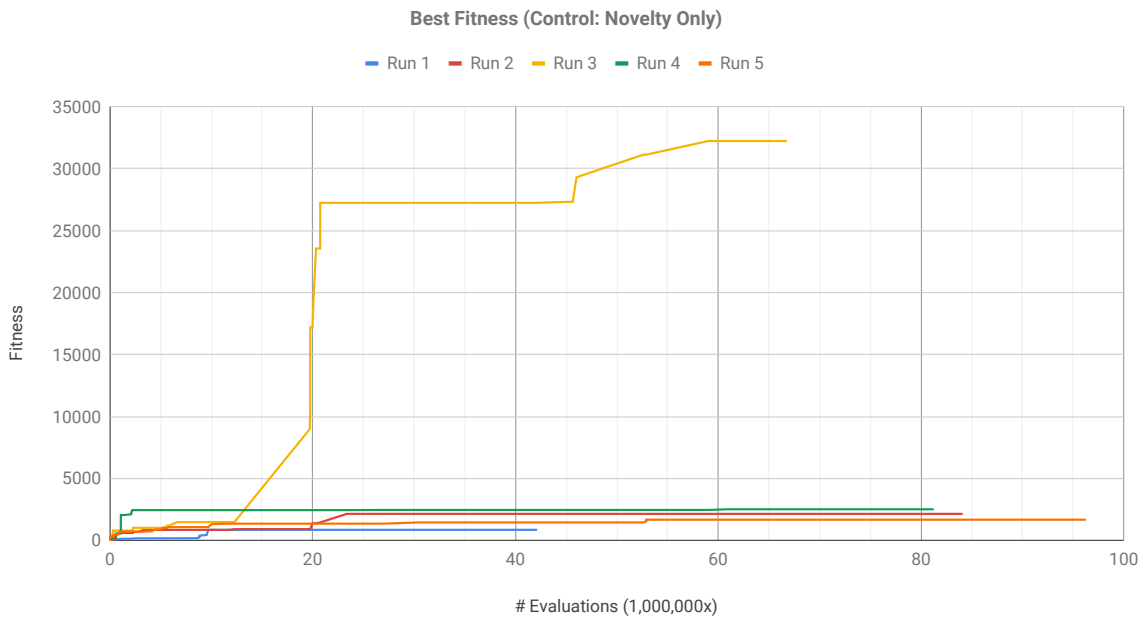


Figure 10.20: **Best fitness over time.** Novelty alone usually leads to stagnation because it does not refine any of the behaviors it discovers to reach their maximum potential, although on occasion it can stumble luckily upon a behavior that is intrinsically high-scoring without significant effort to improve it.

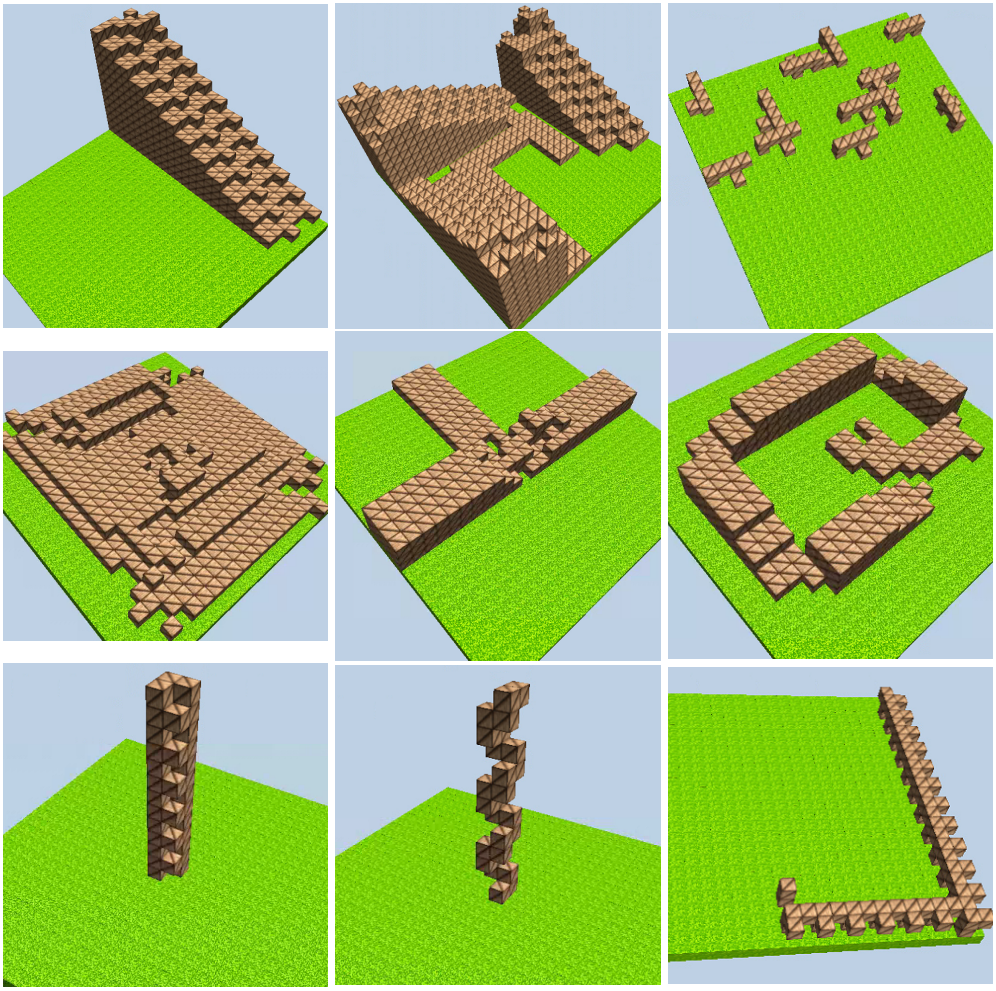


Figure 10.21: **Novelty-only control selected images.** This selection represents some of the most interesting structures produced across all five novelty-driven control runs.

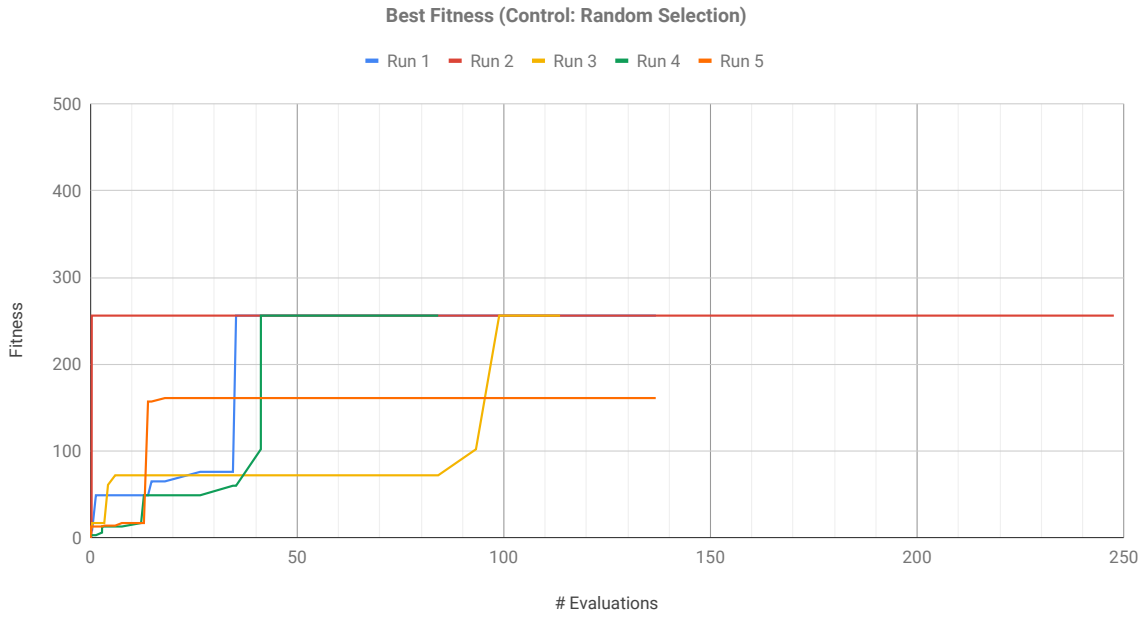


Figure 10.22: **Random-selection control best fitness over time.** Without principled selection of any kind, in effect almost zero progress is possible, confirming that the encoding on its own does not explain the discoveries of QD-based search in Voxelbuild.

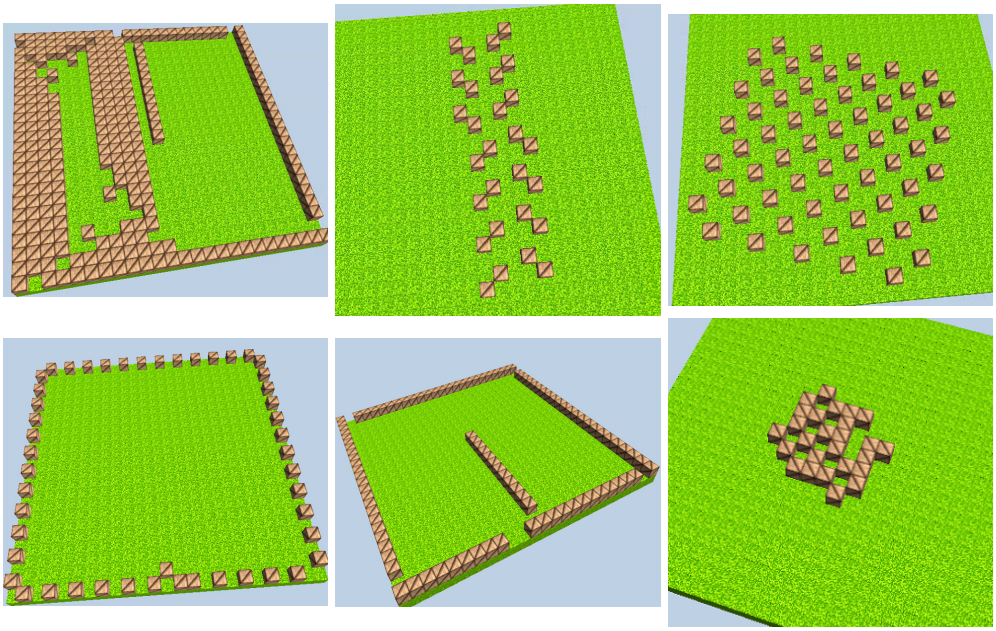


Figure 10.23: **Random-selection-only control selected images.** This selection is representative of the most interesting structures produced across all five random-selection control runs. The first image depicts the highest-scoring structure that appears in almost all runs; agents build this type of structure by following an outward spiral pattern that eventually terminates at the inner portion of the spiral on the left-hand side of the structure.

CHAPTER 11: DISCUSSION

This work as a whole seeks to achieve a deeper understanding of an emerging class of evolutionary algorithms called *quality diversity* and then demonstrate that these new algorithms can serve as an engine for open-ended discovery.

This goal is addressed through a comprehensive study of QD spanning many different algorithms (both existing and novel approaches), behavior characterizations, and maze-navigation tasks. First, an investigation into the relationship between the two major components of quality diversity – the behavior characterization and the fitness function – results in the discovery of a concept called *alignment* that affects search performance and may help inform the design of behavior characterizations in future applications of QD. The study continues by pushing quality diversity to its limit with a series of increasingly difficult tasks, demonstrating that at sufficient task difficulty, search driven by an unaligned behavior characterization breaks down (regardless of algorithm) and cannot solve the problem while *aligned* characterizations still can.

Throughout this work, two novel modifications to the standard MAP-Elites algorithm are introduced that in some cases can improve performance: (1) *MAP-Elites + Novelty (MENOV)* and (2) *MAP-Elites + Passive Genetic Diversity (MEPGD)*. Furthermore, the problem of alignment is addressed through a new class of QD algorithms capable of searching with two behavior characterizations at the same time: achieving good performance with an aligned characterization while satisfying the user's desired notion of diversity with an unaligned characterization. Four such algorithms are presented and subsequently tested on difficult problems where their performance generally surpasses that of their single-characterization counterparts: (1) *Multi-BC Novelty Search (NS-NS)*, (2) *Multi-BC Novelty Search with Local Competition (NS-NSLC)*, (3) *Multi-BC MAP-Elites (ME-ME)*, and (4) *Multi-BC MAP-Elites + Novelty (MENOV-MENOV)*.

Equipped with new knowledge about how to best configure and apply QD algorithms, this work then introduces a new visually-appreciable creative building task called Voxelbuild and illustrates the presence of open-ended properties in this new domain by consistently achieving two *major evolutionary transitions* with NSLC. The application of QD to Voxelbuild is supported by a new extension to the HyperNEAT neuroevolution method called the *multi-spatial substrate* that allows for the successful evolution of large neural controllers with many unrelated sensor or effector modalities (such as those in Voxelbuild).

Voxelbuild is then refined to achieve better results faster and more reliably by (1) optimizing the HyperNEAT substrate, (2) providing the agent with new sensory information allowing it to localize in an otherwise barren landscape, and (3) modifying the behavior characterization and fitness function to better reflect human intuition about structural quality and diversity. It then explores the behavior of quality diversity algorithms over long evolutionary timescales by tracking the development of novelty and complexity in runs spanning millions of generations, several orders of magnitude longer than previous experiments.

The core idea explored by Voxelbuild is that quality diversity's aptitude for illuminating the fitness potential across a fixed, tractable behavior space might translate into a powerful tool for open-ended discovery when the behavior space is vast or unbounded. Even when it is impossible to visit all parts of the behavior space, quality diversity remembers promising individuals from all parts that it *does* visit, increasing the likelihood of finding stepping stones to higher order behaviors. Such stepping stones represent an opportunity for evolution to access entirely new parts of the behavior space by building upon old skills, with each step climbing the ladder of complexity as individuals on the search frontier become increasingly advanced.

When not traversing into higher orders of complexity, QD otherwise wanders around the space favoring the direction of behaviors it has never seen before (especially when driven by an algorithm

with explicit novelty pressure) or else that improve upon existing discoveries. In the context of a boundless space of possibilities, the hope is that this process continues indefinitely, with each run over time constructing a unique evolutionary timeline in a way comparable to that observed in the history of natural evolution on Earth. The experimental results support that indeed such continual innovation can persist over very long timescales.

If open-endedness can be described as *the continual production of increasingly complex novel forms,* then a successful realization of the proposed Voxelbuild system implies that QD should be considered as a possible component of any artificial OEE (open-ended evolution) system whose goal is to achieve open-endedness because QD searches for the requisite properties directly: (1) novelty is explicitly rewarded and (2) meaningful complexity increases are inevitable when the novelty available at lower complexity levels is eventually exhausted.

Much of the OEE community focuses on identifying and implementing some ensemble of lower-level mechanisms observed in nature in the hope of achieving open-endedness as an emergent property of the complex interactions between the chosen mechanisms. At the highest level of abstraction is often a "survival of the fittest" and limited-resources style reward structure, while some practitioners simulate lower level biological processes all the way down to the cellular level.

Instead, QD sidesteps the issue altogether and replaces such mechanisms with a selection criteria at the highest level that favors candidates that are different than or else better than their closest neighbors. The result is a search process designed to find and collect niches across the search space where each niche represents a unique take on solving the problem. In a closed domain this effectively illuminates all of the best solutions while in an open domain the hope is that QD discovers new types of solutions forever.

QD's importance thus goes beyond an effective engine to drive open-ended evolution. In QD, diversity is not merely an aftereffect – instead it is the primary target of search. The nature of this

diversity is determined a priori (by defining an appropriate behavior characterization) to best suit the needs of the user. That QD excels at generating the exact type of diversity asked of it makes it ideal as a design tool especially when the desired dimensions of variation are not directly coded but rather result from complex interactions between the solution parameters and the environment. A QD practitioner, for example, might be interested in finding wheeled robots that drive fast in a wide variety of ways. Given an encoding that specifies the brain and morphology of the robot, applying QD with a behavior characterization representing ways of driving and a quality metric representing driving speed allows an evolutionary search process to collect and optimize a variety of unique driving patterns without knowing them a priori.

The research presented in Chapter 10, by producing a wide variety of high-quality Voxelbuild structures in a single run even when evolution only has access to adjust agent neural controllers, effectively demonstrates QD's potential as a design tool and offers more broadly a mechanism suitable for driving open-ended discovery. These new applications stand to elevate evolution above simple optimization by exploiting its unique capacity to explore many promising directions at the same time.

This work demonstrates that quality diversity search can successfully drive an evolutionary timeline full of nature-like discovery, thus offering the alife community a tool to help achieve and study open-endedness while simultaneously suggesting through example to the engineering community that an intentionally divergent form of evolution can aide in design problems by finding in a single run a continuous stream of different candidate solutions according to an arbitrary notion of diversity.

CHAPTER 12: CONCLUSIONS

This dissertation presented a series of experiments evolving neural controllers for maze-navigating robots with a class of evolutionary search techniques called quality diversity that discover many different ways to drive through the maze in a single run. These experiments (1) represent the first extensive study comparing different approaches to QD and inspired a number of new multi-BC approaches introduced for the first time here that allow researchers to apply QD to unaligned notions of diversity even on difficult problems. This dissertation also (2) introduced and empirically validated a new way to encode networks in HyperNEAT that allows the successful evolution of controllers for robots with many sensory and effector modalities. In an effort to expand the applications of QD to open-ended creative problems, this dissertation (3) introduced a new experimental platform called Voxelbuild where QD continually evolves robots that stack blocks in new and more complex ways. Finally, (4) a series of improvements to the basic Voxelbuild platform enabled a capstone experiment testing QD's ability to continue to innovate even over very long timescales. Together, these contributions help advance the nascent field of quality diversity towards realizing its potential to harness the toolkit of natural evolution not only as an automated problem-solver but now also as an endless source of creativity.

LIST OF REFERENCES

[1] P. J. Angeline, G. M. Saunders, and J. B. Pollack. An evolutionary algorithm that constructs recurrent neural networks. 2000. In press.

[2] T. Bäck, U. Hammel, and H.-P. Schwefel. Evolutionary computation: Comments on the history and current state. *IEEE Transactions on Evolutionary computation*, 1(1):3–17, 1997.

[3] M. Bedau. The arrow of complexity hypothesis (abstract). In S. Bullock, J. Noble, R. Watson, and M. Bedau, editors, *Proceedings of the Eleventh International Conference on Artificial Life (Alife XI)*, page 750, Cambridge, MA, 2008. MIT Press. URL `http://www.alifexi.org/papers/ALIFExi-abstracts-0010.pdf`.

[4] M. A. Bedau and N. H. Packard. Measurement of evolutionary activity, teleology, and life. In C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, editors, *Proc. of Artificial Life II*, pages 431–461, Redwood City, CA, 1991. Addison-Wesley.

[5] M. A. Bedau, E. Snyder, and N. H. Packard. A comparison of evolutionary activity in artificial evolving systems and the biosphere. In *Proceedings of the Fourth European Conference on Artificial Life*, pages 125–134, Cambridge, MA, 1997. MIT Press.

[6] M. A. Bedau, E. Snyder, and N. H. Packard. A classification of long-term evolutionary dynamics. In *Artificial life VI*, pages 228–237. Cambridge: MIT Press, 1998.

[7] M. A. Bedau, J. S. McCaskill, N. H. Packard, S. Rasmussen, C. Adami, D. G. Green, T. Ikegami, K. Kaneko, and T. S. Ray. Open problems in artificial life. *Artificial life*, 6 (4):363–376, 2000.

[8] M. A. Bedau, J. S. McCaskill, N. H. Packard, S. Rasmussen, C. Adami, D. G. Green,

T. Ikegami, K. Kaneko, and T. S. Ray. Open problems in artificial life. *Artificial Life*, 6 (4):363–376, 2000.

[9] P. J. Bentley and S. Kumar. Three ways to grow designs: A comparison of embryogenies for an evolutionary design problem. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 1999)*, pages 35–43, 1999.

[10] C. M. Bishop. *Pattern Recognition and Machine Learning*, volume 1. Springer, New York,NY, 2006.

[11] M. Blum. On the size of machines. *Information and control*, 11(3):257–265, 1967.

[12] M. Boden. *Mind as machine: a history of cognitive science*. Oxford University Press, USA, 2006.

[13] J. C. Bongard. Evolving modular genetic regulatory networks. In *Proceedings of the 2002 Congress on Evolutionary Computation*, 2002.

[14] J. C. Bongard and R. Pfeifer. Evolving complete agents using artificial ontogeny. *Morphofunctional Machines: The New Species (Designing Embodied Intelligence)*, pages 237–258, 2003.

[15] A. Channon. Improving and still passing the ALife test: Component-normalised activity statistics classify evolution in geb as unbounded. In *Proc. of Artificial Life VIII*, pages 173–181, Cambridge, MA, 2003. MIT Press.

[16] D. C. Cireşan, U. Meier, L. M. Gambardella, and J. Schmidhuber. Deep, big, simple neural nets for handwritten digit recognition. *Neural computation*, 22(12):3207–3220, 2010.

[17] D. Cliff, I. Harvey, and P. Husbands. Explorations in evolutionary robotics. *Adaptive Behavior*, 2:73–110, 1993.

[18] J. Clune, B. E. Beckmann, C. Ofria, and R. T. Pennock. Evolving coordinated quadruped gaits with the HyperNEAT generative encoding. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2009) Special Session on Evolutionary Robotics*, Piscataway, NJ, USA, 2009. IEEE Press.

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

[20] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[21] A. Cully and Y. Demiris. Quality and diversity optimization: A unifying modular framework. *IEEE Transactions on Evolutionary Computation*, 22(2):245–259, 2018.

[22] A. Cully and J.-B. Mouret. Behavioral repertoire learning in robotics. In *Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation (GECCO '13)*, pages 175–182, New York, NY, USA, 2013. ACM.

[23] A. Cully, J. Clune, and J. Mouret. Robots that can adapt like natural animals. *ArXiv e-prints*, abs/1407.3501, 2014. URL `http://arxiv.org/abs/1407.3501`.

[24] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret. Robots that can adapt like animals. *Nature*, 521(7553):503–507, 2015.

[25] D. B. D'Ambrosio, J. Lehman, S. Risi, and K. O. Stanley. Evolving policy geometry for scalable multiagent learning. In *Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, pages 731–738. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

[26] D. B. D'Ambrosio, J. Lehman, S. Risi, and K. O. Stanley. Task switching in multirobot learning in multiagent learning through indirect encoding. In *Proceedings of the Interna-*

tional Conference on Intelligent Robots and Systems (IROS 2011), Piscataway, NJ, 2011. IEEE.

[27] D. Dasgupta and D. McGregor. Designing application-specific neural networks using the structured genetic algorithm. In *Proceedings of the International Conference on Combinations of Genetic Algorithms and Neural Networks*, pages 87–96, 1992.

[28] K. A. De Jong. *Evolutionary Computation: A Unified Perspective*. MIT Press, Cambridge, MA, 2002.

[29] H. P. de Vladar, M. Santos, and E. Szathmáry. Grand views of evolution. *Trends in Ecology & Evolution*, 2017.

[30] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[31] S. Doncieux, N. Bredeche, J.-B. Mouret, and A. G. Eiben. Evolutionary robotics: What, why, and where to. *Frontiers in Robotics and AI*, 2(4), 2015. ISSN 2296-9144. doi: 10.3389/frobt.2015.00004. URL `http://www.frontiersin.org/evolutionary_robotics/10.3389/frobt.2015.00004/abstract`.

[32] J. Doucette and M. I. Heywood. Novelty-based fitness: An evaluation under the santa fe trail. In *Proceedings of the European Conference on Genetic Programming (EuroGP-2010*, pages 50–61. Springer, 2010.

[33] J. Drchal, O. Kapral, J. Koutník, and M. Šnorek. Combining multiple inputs in HyperNEAT mobile agent controller. In *Artificial Neural Networks–ICANN 2009*, pages 775–783. Springer, 2009.

[34] N. Eldredge and S. J. Gould. Punctuated equilibria: an alternative to phyletic gradualism. *Models in paleobiology*, 82:115, 1972.

[35] D. Floreano and F. Mondada. Automatic creation of an autonomous agent: Genetic evolution of a neural-network driven robot. In *From Animals to Animats III: Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, pages 421–430. MIT Press, 1994.

[36] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence Through Simulated Evolution*. John Wiley & Sons, 1966.

[37] J. Gauci and K. O. Stanley. Autonomous evolution of topographic regularities in artificial neural networks. *Neural Computation*, 22(7):1860–1898, 2010.

[38] M. Gell-Mann and S. Lloyd. Information measures, effective complexity, and total information. *Complexity*, 2(1):44–52, 1996.

[39] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. 1989.

[40] D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. pages 148–154, 1987. ISBN 1-55860-066-3. URL `http://www.mpi-sb.mpg.de/services/library/proceedings/contents/icga87.html`.

[41] H. J. Goldsby and B. H. Cheng. Automatically discovering properties that specify the latent behavior of uml models. In *Model Driven Engineering Languages and Systems*, pages 316–330. Springer, 2010.

[42] J. Gomes and A. L. Christensen. Generic behaviour similarity measures for evolutionary swarm robotics. In *Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation (GECCO '13)*, pages 199–206, New York, NY, USA, 2013. ACM.

[43] J. Gomes, P. Urbano, and A. L. Christensen. Evolution of swarm robotics systems with novelty search. *Swarm Intelligence*, pages 1–30, 2013.

[44] J. Gomes, P. Mariano, and A. L. Christensen. Devising effective novelty search algorithms: A comprehensive empirical study. In *Proceedings of the 17th Annual Conference on Genetic and Evolutionary Computation*, pages 943–950. ACM, 2015.

[45] L. Graening, N. Aulig, and M. Olhofer. Towards directed open-ended search by a novelty guided evolution strategy. In R. Schaefer, C. Cotta, J. Kołodziej, and G. Rudolph, editors, *Parallel Problem Solving from Nature – PPSN XI*, volume 6239 of *Lecture Notes in Computer Science*, pages 71–80. Springer, 2010. ISBN 978-3-642-15870-4.

[46] C. Green. SharpNEAT homepage. `http://sharpneat.sourceforge.net/`, 2003–2006.

[47] I. Harvey. *The Artificial Evolution of Adaptive Behavior*. PhD thesis, School of Cognitive and Computing Sciences, University of Sussex, Sussex, 1993. URL `http://www.cogs.susx.ac.uk/users/inmanh/inman_thesis.html`.

[48] J. H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. University of Michigan Press, Ann Arbor, MI, 1975.

[49] G. S. Hornby and J. B. Pollack. The advantages of generative grammatical encodings for physical design. In *Proceedings of the 2002 Congress on Evolutionary Computation*, 2001. URL `http://demo.cs.brandeis.edu/papers/long.html#hornby_cec01`.

[50] G. S. Hornby and J. B. Pollack. Creating high-level components with a generative representation for body-brain evolution. *Artificial Life*, 8(3), 2002.

[51] D. Hubel and T. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, 160(1):106, 1962.

[52] I. M. Johnstone and A. Y. Lu. On consistency and sparsity for principal components analysis in high dimensions. *Journal of the American Statistical Association*, 104(486):682–693, 2009.

[53] E. R. Kandel, J. H. Schwartz, and T. M. Jessell. *Principles of Neural Science*. McGraw-Hill, New York, fourth edition, 2000.

[54] M. Kirschner and J. Gerhart. Evolvability. *Proceedings of the National Academy of Sciences*, 95(15):8420–8427, 1998.

[55] S. Kistemaker and S. Whiteson. Critical factors in the performance of novelty search. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, GECCO '11, pages 965–972, 2011. ISBN 978-1-4503-0557-0. doi: 10.1145/2001576.2001708. URL `http://doi.acm.org/10.1145/2001576.2001708`.

[56] E. V. Koonin. The biological big bang model for the major transitions in evolution. *Biology Direct*, 2(1):21, 2007.

[57] P. Krcah. Solving deceptive tasks in robot body-brain co-evolution by searching for behavioral novelty. In *10th International Conference on Intelligent Systems Design and Applications (ISDA)*, pages 284–289. IEEE, 2010.

[58] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

[59] J. Lehman and K. O. Stanley. Exploiting open-endedness to solve problems through the search for novelty. In S. Bullock, J. Noble, R. Watson, and M. Bedau, editors, *Proceedings of the Eleventh International Conference on Artificial Life (Alife XI)*, Cambridge, MA, 2008. MIT Press.

[60] J. Lehman and K. O. Stanley. Revising the evolutionary computation abstraction: Minimal criteria novelty search. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation (GECCO '10)*, pages 103–110, New York, NY, USA, 2010. ACM.

[61] J. Lehman and K. O. Stanley. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary Computation*, 19(2):189–223, 2011.

[62] J. Lehman and K. O. Stanley. Evolving a diversity of virtual creatures through novelty search and local competition. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation (GECCO '11)*, pages 211–218. ACM, 2011.

[63] J. Lehman and K. O. Stanley. Evolvability is inevitable: Increasing evolvability without the pressure to adapt. *PLoS ONE*, 8(4):e62186, 2013.

[64] J. Lehman, S. Risi, and J. Clune. Creative generation of 3d objects with deep learning and innovation engines. In *Proceedings of the 7th International Conference on Computational Creativity*, 2016.

[65] R. E. Lenski, C. Ofria, R. T. Pennock, and C. Adami. The evolutionary origin of complex features. *Nature*, 423(6936):139–144, 2003.

[66] X. Li, A. Engelbrecht, and M. G. Epitropakis. Benchmark functions for cec2013 special session and competition on niching methods for multimodal function optimization. *RMIT University, Evolutionary Computation and Machine Learning Group, Australia, Tech. Rep*, 2013.

[67] A. Liapis, H. P. Martínez, J. Togelius, and G. N. Yannakakis. Transforming exploratory creativity with delenox. In *Proceedings of the Fourth International Conference on Computational Creativity*, 2013.

[68] A. Liapis, G. N. Yannakakis, and J. Togelius. Enhancements to constrained novelty search: two-population novelty search for generating game content. In *Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference*, GECCO '13, pages 343–350, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1963-8. doi: 10.1145/2463372.2463416. URL `http://doi.acm.org/10.1145/2463372.2463416`.

[69] D. G. Loyola and M. Coldewey-Egbers. Multi-sensor data merging with stacked neural networks for the creation of satellite long-term climate data records. *EURASIP Journal on Advances in Signal Processing*, 2012(1):1–10, 2012.

[70] S. W. Mahfoud. *Niching Methods for Genetic Algorithms*. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL, May 1995.

[71] C. R. Marshall. Explaining the cambrian "explosion" of animals. *Annu. Rev. Earth Planet. Sci.*, 34:355–384, 2006.

[72] Y. Martinez, E. Naredo, L. Trujillo, and E. Galvan-Lopez. Searching for novel regression functions. In *2013 IEEE Congress on Evolutionary Computation (CEC)*, pages 16–23. IEEE, 2013.

[73] T. Miconi and A. Channon. A virtual creatures model for studies in artificial evolution. In *The 2005 IEEE Congress on Evolutionary Computation*, volume 1, pages 565–572. IEEE, 2005.

[74] J. Miller. Evolving a self-repairing, self-regulating, french flag organism. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2004)*, pages 129–139. Springer, 2004.

[75] T. M. Mitchell. *Machine Learning*. McGraw-Hill, New York, NY, 1997.

[76] G. Morse, S. Risi, C. R. Snyder, and K. O. Stanley. Single-unit pattern generators for quadruped locomotion. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation (GECCO '13)*, pages 719–726. ACM, 2013.

[77] J.-B. Mouret. Novelty-based multiobjectivization. In *New Horizons in Evolutionary Robotics*, pages 139–154. Springer, Berlin Heidelberg, 2011.

[78] J.-B. Mouret and J. Clune. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*, 2015.

[79] J.-B. Mouret and S. Doncieux. Overcoming the bootstrap problem in evolutionary robotics using behavioral diversity. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC-2009)*, pages 1161–1168. IEEE, 2009.

[80] J.-B. Mouret and S. Doncieux. Encouraging behavioral diversity in evolutionary robotics: An empirical study. *Evolutionary Computation*, 20(1):91–133, 2012.

[81] E. Naredo and L. Trujillo. Searching for novel clustering programs. In *Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference*, GECCO '13, pages 1093–1100, 2013. ISBN 978-1-4503-1963-8. doi: 10.1145/2463372.2463505. URL `http://doi.acm.org/10.1145/2463372.2463505`.

[82] A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. *ArXiv e-prints*, abs/1412.1897, 2014. URL `http://arxiv.org/abs/1412.1897`.

[83] A. Nguyen, J. Yosinski, and J. Clune. Innovation engines: Automated creativity and improved stochastic optimization via deep learning. In *Proceedings of the 2015 Conference on Genetic and Evolutionary Computation*, GECCO '15, New York, NY, USA, 2015. ACM.

[84] A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '15)*. IEEE, 2015.

[85] A. Nguyen, J. Yosinski, and J. Clune. Innovation engines: Automated creativity and improved stochastic optimization via deep learning. In *Proceedings of the 17th Annual Conference on Genetic and Evolutionary Computation (GECCO '15)*, New York, NY, USA, 2015. ACM.

[86] S. Nolfi. Evolving non-trivial behavior on real robots: A garbage collecting robot. *Robotics and Autonomous Systems*, 22:187–198, 1997.

[87] S. Nolfi and D. Floreano. *Evolutionary Robotics*. MIT Press, Cambridge, 2000.

[88] C. Ofria and C. O. Wilke. Avida: A software platform for research in computational evolutionary biology. *Artificial life*, 10(2):191–229, 2004.

[89] J. K. Pugh and K. O. Stanley. Evolving multimodal controllers with HyperNEAT. In *Proceeding of the fifteenth annual conference on genetic and evolutionary computation*, GECCO '13, pages 735–742, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1963-8. doi: 10.1145/2463372.2463459. URL `http://doi.acm.org/10.1145/2463372.2463459`.

[90] J. K. Pugh, L. B. Soros, P. A. Szerlip, and K. O. Stanley. Confronting the challenge of quality diversity. In *Proceedings of the 17th Annual Conference on Genetic and Evolutionary Computation (GECCO '15)*, New York, NY, USA, 2015. ACM.

[91] J. K. Pugh, L. Soros, and K. O. Stanley. Searching for quality diversity when diversity is unaligned with quality. In *International Conference on Parallel Problem Solving from Nature*, pages 880–889. Springer, 2016.

[92] J. K. Pugh, L. B. Soros, and K. O. Stanley. Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI*, 3:40, 2016.

[93] J. K. Pugh, L. B. Soros, and K. O. Stanley. An extended study of quality diversity algorithms. In *Proceedings of the 2016 Genetic and Evolutionary Computation Conference Companion*, pages 19–20. ACM, 2016.

[94] J. K. Pugh, L. B. Soros, R. Frota, K. Negy, and K. O. Stanley. Major evolutionary transitions in the voxelbuild virtual sandbox game. In *Artificial Life Conference Proceedings 14*, pages 553–560. MIT Press One Rogers Street, Cambridge, MA 02142-1209 USA journals-info@ mit , 2017.

[95] J. C. F. Pujol and R. Poli. Evolving the topology and the weights of neural networks using a dual representation. *Applied Intelligence Journal*, 8(1):73–84, January 1998. Special Issue on Evolutionary Learning.

[96] N. J. Radcliffe. Genetic set recombination and its application to neural network topology optimization. *Neural computing and applications*, 1(1):67–90, 1993.

[97] T. S. Ray. An approach to the synthesis of life. In *Proc. of Artificial Life II*, pages 371–408. Addison-Wesley, 1992.

[98] M. Ridley. *Evolution, 3rd Edition*. 2004.

[99] S. Risi and K. O. Stanley. An enhanced hypercube-based encoding for evolving the placement, densty and connectivity of neurons. *Artificial Life*, 18(4):331–363, 2012.

[100] S. Risi, S. D. Vanderbleek, C. E. Hughes, and K. O. Stanley. How novelty search escapes the deceptive trap of learning to learn. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2009)*, New York, NY, USA, 2009. ACM Press.

[101] S. Risi, C. Hughes, and K. Stanley. Evolving plastic neural networks with novelty search. *Adaptive Behavior*, 18(6):470–491, 2011.

[102] S. Risi, , and K. O. Stanley. Confronting the challenge of learning a flexible neural controller for a diversity of morphologies. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2013)*, New York, NY, USA, 2013. ACM.

[103] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In *Parallel Distributed Processing, Volume 1*, pages 318–362. MIT Press, Cambridge, MA, 1986.

[104] J. Schrum and R. Miikkulainen. Evolving multimodal networks for multitask games. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(2):94–111, 2012.

[105] P. Schuster. Major transitions in evolution and in technology: What they have in common and where they differ. *Complexity*, 21(4):7–13, 2016. ISSN 1099-0526. doi: 10.1002/cplx.21773. URL `http://dx.doi.org/10.1002/cplx.21773`.

[106] H.-P. P. Schwefel. *Evolution and optimum seeking: the sixth generation*. John Wiley & Sons, Inc., 1993.

[107] H. A. Simon. *Models of man: Social and rational – Mathematical Essays on Rational Human Behavior in a Social Setting*. Wiley, 1957.

[108] K. Sims. Evolving 3d morphology and behavior by competition. *Artificial Life*, 1(4):353–372, 1994.

[109] R. V. Solé and S. Valverde. Before the endless forms: embodied model of transition from single cells to aggregates to ecosystem engineering. *PLoS ONE*, 8(4):e59664, 2013.

[110] A. Soltoggio and B. Jones. Novelty of behaviour as a basis for the neuro-evolution of operant reward learning. In *Proceedings of the 11th Annual conference on Genetic and*

evolutionary computation, GECCO '09, pages 169–176, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-325-9. doi: 10.1145/1569901.1569925. URL http://doi.acm.org/10.1145/1569901.1569925.

[111] L. Spector, J. Klein, and M. Feinstein. Division blocks and the open-ended evolution of development, form, and behavior. In *Proc. of the 9th annual conf. on Genetic and evolutionary computation*, pages 316–323. ACM, 2007.

[112] R. K. Standish. Open-ended artificial evolution. *International Journal of Computational Intelligence and Applications*, 3(02):167–175, 2003.

[113] K. O. Stanley. Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines Special Issue on Developmental Systems*, 8(2):131–162, 2007.

[114] K. O. Stanley. HyperNEAT user's page. http://eplex.cs.ucf.edu/hyperNEATpage/, 2009–2013.

[115] K. O. Stanley. Why evolutionary robotics will matter. In S. Doncieux, N. Bredeche, and J.-B. Mouret, editors, *New horizons in evolutionary robotics*, pages 37–41. Springer, 2011.

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

[117] K. O. Stanley and R. Miikkulainen. A taxonomy for artificial embryogeny. *Artificial Life*, 9 (2):93–130, 2003.

[118] K. O. Stanley and R. Miikkulainen. Competitive coevolution through evolutionary complexification. *Journal of Artificial Intelligence Research (JAIR)*, 21:63–100, 2004.

[119] K. O. Stanley, B. D. Bryant, and R. Miikkulainen. Real-time neuroevolution in the NERO video game. *IEEE Transactions on Evolutionary Computation Special Issue on Evolutionary Computation and Games*, 9(6):653–668, 2005.

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

[121] E. Szathmáry and J. Maynard Smith. The major evolutionary transitions. *Nature*, 374(6519): 227–232, 1995.

[122] P. Szerlip and K. O. Stanley. Indirectly encoded sodarace for artificial life. In *Proceedings of the European Conference on Artificial Life (ECAL-2013)*, volume 12, pages 218–225, 2013.

[123] P. A. Szerlip, G. Morse, J. K. Pugh, and K. O. Stanley. Unsupervised feature learning through divergent discriminative feature accumulation. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-2015)*, Menlo Park, CA, 2015. AAAI Press.

[124] T. Taylor. Requirements for open-ended evolution in natural and artificial systems. *CoRR*, abs/1507.07403, 2015. URL `http://arxiv.org/abs/1507.07403`.

[125] T. Taylor, M. Bedau, A. Channon, D. Ackley, W. Banzhaf, G. Beslon, E. Dolson, T. Froese, S. Hickinbotham, T. Ikegami, et al. Open-ended evolution: perspectives from the oee workshop in york. *Artificial Life*, 2016.

[126] L. Trujillo, G. Olague, E. Lutton, and F. de Vega. Discovering Several Robot Behaviors through Speciation. *Applications of Evolutionary Computing: Evoworkshops 2008: Evocomnet, Evofin, Evohot, Evoiasp, Evomusart, Evonum, Evostoc, and Evotranslog*, page 164, 2008.

[127] L. Trujillo, G. Olague, E. Lutton, F. F. De Vega, L. Dozal, and E. Clemente. Speciation in behavioral space for evolutionary robotics. *Journal of Intelligent & Robotic Systems*, 64 (3-4):323–351, 2011.

[128] S. Udin and J. Fawcett. Formation of topographic maps. *Annual Review of Neuroscience*, 11(1):289–327, 1988.

[129] R. Velez and J. Clune. Novelty search creates robots with general skills for exploration. In *Proceedings of the 2014 Conference on Genetic and Evolutionary Computation*, GECCO '14, pages 737–744, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2662-9. doi: 10.1145/2576768.2598225. URL `http://doi.acm.org/10.1145/2576768.2598225`.

[130] P. Verbancsics and K. O. Stanley. Constraining connectivity to encourage modularity in hyperneat. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2011)*, pages 1483–1490. ACM, 2011.

[131] B. G. Woolley and K. O. Stanley. On the deleterious effects of a priori objectives on evolution and representation. In *GECCO '11: Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 957–964, Dublin, Ireland, 12-16 July 2011. ACM. ISBN 978-1-4503-0557-0. doi: doi:10.1145/2001576.2001707.

[132] X. Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, 1999.