

Electronic Theses and Dissertations, 2020-

2020

Open-ended Search through Minimal Criterion Coevolution

Jonathan Brant
University of Central Florida

 Part of the [Computer Sciences Commons](#)
Find similar works at: <https://stars.library.ucf.edu/etd2020>
University of Central Florida Libraries <http://library.ucf.edu>

This Doctoral Dissertation (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Electronic Theses and Dissertations, 2020- by an authorized administrator of STARS. For more information, please contact STARS@ucf.edu.

STARS Citation

Brant, Jonathan, "Open-ended Search through Minimal Criterion Coevolution" (2020). *Electronic Theses and Dissertations, 2020-*. 21.
<https://stars.library.ucf.edu/etd2020/21>



OPEN-ENDED SEARCH THROUGH MINIMAL CRITERION COEVOLUTION

by

JONATHAN C. BRANT
M.S. University of Central Florida, 2011
B.S. University of Central Florida, 2008

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
2020

Major Professor: Kenneth O. Stanley

© 2020 Jonathan C. Brant

ABSTRACT

Search processes guided by objectives are ubiquitous in machine learning. They iteratively reward artifacts based on their proximity to an optimization target, and terminate upon solution space convergence. Some recent studies take a different approach, capitalizing on the disconnect between mainstream methods in artificial intelligence and the field’s biological inspirations. Natural evolution has an unparalleled propensity for generating well-adapted artifacts, but these artifacts are decidedly non-convergent. This new class of *non-objective* algorithms induce a *divergent* search by rewarding solutions according to their novelty with respect to prior discoveries. While the diversity of resulting innovations exhibit marked parallels to natural evolution, the methods by which search is driven remain unnatural. In particular, nature has no need to characterize and enforce novelty; rather, it is guided by a single, simple constraint: survive long enough to reproduce. The key insight is that such a constraint, called the *minimal criterion*, can be harnessed in a coevolutionary context where two populations interact, finding novel ways to satisfy their reproductive constraint with respect to each other. Among the contributions of this dissertation, this approach, called *minimal criterion coevolution* (MCC), is the primary (1). MCC is initially demonstrated in a maze domain (2) where it evolves increasingly complex mazes and solutions. An enhancement to the initial domain (3) is then introduced, allowing mazes to expand unboundedly and validating MCC’s propensity for open-ended discovery. A more natural method of diversity preservation through resource limitation (4) is introduced and shown to maintain population diversity without comparing genetic distance. Finally, MCC is demonstrated in an evolutionary robotics domain (5) where it coevolves increasingly complex bodies with brain controllers to achieve principled locomotion. The overall benefit of these contributions is a novel, general, algorithmic framework for the continual production of open-ended dynamics without the need for a characterization of behavioral novelty.

To my wife, Amanda.

ACKNOWLEDGMENTS

First and foremost, I would like to extend my utmost gratitude to my advisor, Dr. Kenneth O. Stanley, who invested countless hours in thought provoking discussion and instructional critique. His patient guidance shaped my ability to clearly communicate scientific ideas, both verbally and in writing.

An especially significant dose of appreciation is due to my wife, Amanda Brant, who so graciously chose to remain in that capacity throughout the duration of my studies. Her flexibility and loving support was instrumental in the preservation of my sanity. Special thanks are also due to my parents, Charles and Angela Brant, who encouraged my obsessive interest in computers from a young age.

I would also like to thank Dr. Annie S. Wu, who through her evolutionary computation classes and Evolutionary Computation Lab (ECLab) meetings, fostered in me an insatiable passion for the field, setting me on a trajectory that ultimately culminated in this dissertation.

Thanks also to my committee members, Dr. Ivan Garibay, Dr. Eric Hoffman and Dr. Haiyan (Nancy) Hu. Their helpful suggestions and oversight was instrumental in improving the quality of this dissertation.

I am especially grateful to past and present Lockheed Martin coworkers, most notably Dr. Gregory Harrison, Adam Kalicki, Juan Gomez, Kevin Krause, Kris Siegmundt and Matthew Milas, who authored flattering letters of recommendation and secured company sponsorship to cover tuition expenses.

Finally, I would like to extend thanks to former members of the Evolutionary Complexity Research Group (Eplex), including Dr. Navid Kardan, Dr. Gregory Morse, Dr. Justin Pugh and Dr.

Lisa Soros. Their thoughtful feedback and critique was instrumental in identifying new research directions and improving my presentation ability.

TABLE OF CONTENTS

LIST OF FIGURES	x
CHAPTER 1: INTRODUCTION	1
CHAPTER 2: BACKGROUND	9
2.1 Evolutionary Computation	9
2.2 Diversity Preservation	10
2.3 Open-endedness	12
2.4 Coevolution	13
2.5 Neuroevolution of Augmenting Topologies	15
2.6 Compositional Pattern Producing Networks	16
2.7 Non-objective Search	18
2.8 Quality Diversity	20
CHAPTER 3: APPROACH: MINIMAL CRITERION COEVOLUTION	23
CHAPTER 4: INITIAL MAZE DOMAIN	29
4.1 Experiment	29

4.1.1	Maze Evolution	32
4.1.2	Maze Domain Minimal Criterion	34
4.1.3	Experimental Configurations	35
4.1.4	Experimental Parameters	37
4.2	Results	38
4.3	Implications	44
CHAPTER 5: ENHANCED MAZE DOMAIN		46
5.1	Maze Encoding	46
5.1.1	Maze Expansion	47
5.1.2	Path Evolution	48
5.2	Experiment	52
5.3	Results	54
5.4	Implications	61
CHAPTER 6: RESOURCE LIMITATION		63
6.1	Diversity Preservation through Limited Resources	63
6.2	Experiment	65
6.3	Results	68

6.4	Implications	75
CHAPTER 7: BODY-BRAIN COEVOLUTION		78
7.1	Evolutionary Robotics Domain	78
7.2	Experiment	79
7.3	Results	87
7.4	Implications	102
CHAPTER 8: DISCUSSION		104
CHAPTER 9: CONCLUSION		108
APPENDIX A: MAZE GENERATION ALGORITHM		110
APPENDIX B: EXPERIMENTAL PARAMETERS		121
LIST OF REFERENCES		125

LIST OF FIGURES

4.1	Neural network architecture and sensor array of the maze navigating agent . .	31
4.2	Initial maze domain evolution process	34
4.3	Agent trajectories from the MCC control experiment (initial maze domain) . .	39
4.4	Agent trajectories from the MCC speciated variant (initial maze domain) . . .	40
4.5	Diversity of agent trajectories collected over full run (initial maze domain) . .	42
4.6	Average proportion of mazes solved per agent over Evolution (initial maze domain)	43
4.7	Speciated maze complexity trend (initial maze domain)	44
5.1	Comparison between the initial and enhanced maze encodings	48
5.2	Enhanced maze domain evolution process	51
5.3	Agent trajectories discovered within a single run (enhanced maze domain) . .	55
5.4	Maze expansion trend (enhanced maze domain)	56
5.5	Solution path complexification trend (enhanced maze domain)	57
5.6	Agent NN size trend (enhanced maze domain)	58
5.7	Solution path deception trend (enhanced maze domain)	59
5.8	Maze population viability trend (enhanced maze domain)	60

5.9	Average proportion of mazes solved per agent over Evolution (enhanced maze domain)	61
6.1	Solution path diversity of speciation and resource limitation variants	70
6.2	Maze size trend of speciation and resource limitation variants	71
6.3	Maze size distribution of speciation and resource limitation variants	72
6.4	Agent NN size trend of speciation and resource limitation variants	73
6.5	Maze deceptive junctures trend of speciation and resource limitation variants .	74
6.6	Mazes and agent trajectories discovered by MCC with resource limitation . .	75
7.1	Body CPPN architecture	83
7.2	Brain CPPN and NN architecture	85
7.3	Body size trend	88
7.4	Distribution of body sizes	89
7.5	Ambulation distance comparison with co-optimization	91
7.6	Ambulation distance by body size	92
7.7	Trajectory diversity comparison with co-optimization	94
7.8	End-point distribution comparison	95
7.9	Trajectory diversity by body size	96

7.10	Morphological diversity trend	98
7.11	Morphological diversity by body size	99
7.12	Morphological diversity trend for body size $11 \times 11 \times 11$	100
7.13	Sample of evolved bodies	101

CHAPTER 1: INTRODUCTION

A delicate tension has persisted historically between the practically oriented pursuits of evolutionary computation (EC) and the more abstract questions of artificial life (alife) that in recent years may be starting to fade. EC has traditionally been leveraged as merely another optimization method, focused on pragmatic problem solving in real-world domains, and often ignoring more philosophical issues in alife like the pursuit of open-endedness (i.e. investigating and decomposing the processes that continue to generate interesting artifacts indefinitely) [150, 151, 155, 162]. Historically, open-ended evolution like that seen in nature has been studied in the context of Earth-like domains with organism-like occupants and ecosystems [113, 124], a pursuit far removed from evolving targeted solutions to problems like walking or maze-navigation. However, this dichotomy has begun to close with the recent rise of non-objective algorithms that began with the introduction of novelty search [76, 78] and later extended to include aspects of interestingness [88], impressiveness [81], surprise [58–60] and curiosity [156, 157].

Instead of converging on a synthetic objective, this new class of EC algorithms focuses on divergence, i.e. pushing search outward from previously-visited locations, a dynamic that aligns more closely with the seemingly unbounded process of open-ended discovery in nature. A key insight is that these algorithms can be implemented in a more general purpose framework, outside the restrictive and abstract confines of alife. Recently, non-objective algorithms have been extended to incorporate a more objective fitness-like component, balancing divergence with adaptive pressure to induce a process termed *quality diversity* (QD) [123], which simultaneously produces artifacts that are both diverse and of objectively high quality, all within a single run. QD algorithms such as novelty search with local competition (NSLC) [23, 52, 79], and the Multi-dimensional Archive of Phenotypic Elites (MAP-Elites) [14, 24, 37, 46, 111] have started to bridge the gap between practical, general purpose methods and theoretical alife research into the nature of open-endedness.

However, despite the progress in applying non-objective algorithms to more concrete problems, there remains a disconnect between QD algorithms as implemented, and open-ended processes observed in nature. In particular, most QD algorithms published to date require defining a *behavior characterization* (BC), which is a descriptor of the phenotypic behavior or physical traits of individuals in the search space. The BC is often used as a drop-in replacement for an objective function, rewarding individuals to the degree that they exhibit novel behaviors, and inducing a divergent search dynamic through behavior space [36]. Nature lacks any such formalism for describing and comparing individual behaviors, yet remains unencumbered in its ability to continually produce novel artifacts. Moreover, while the descriptor space of the BC may be vast, it is of a fixed dimensionality and therefore intrinsically finite. For example, if an individual’s behavior is characterized in terms of its size and average velocity, then once the full spectrum of sizes and velocities have been sampled, novelty is subsequently diminished and pressure toward continued divergence is by necessity reduced. While recent work has suggested that BCs may be able to expand over time [88, 99], facilitating an unbounded process of expansion remains a notable challenge. Finally, most QD algorithms require a behavioral *archive* that persists descriptions of past behaviors and provides a reference point for the comparison of new behaviors. In the original novelty search algorithm as well as in NSLC and its variants, the archive chronicles points in the behavior space that have been previously visited, while MAP-Elites embeds previously visited points directly into the behavior space map. Nature requires no such archival mechanism, yet still manages to avoid evolutionary regression [11, 56].

Nature is therefore a source of inspiration and a benchmark against which to compare processes of open-ended discovery that lack such artificially imposed constructs. QD algorithms to date rely on quantifying and rewarding novelty as a mechanism to facilitate divergent search, which leads to the need for BCs and behavioral archives; however, a key insight in this dissertation is that divergence could alternatively be achieved through a process of genetic *drift* [82, 86], subject to a minimal

criterion (MC) as the sole constraint for reproduction. While there is some precedent of the MC being used to facilitate an evolutionary process [51, 77, 92, 93, 95], it is difficult to determine how to craft the MC a priori, and even more unclear how to dynamically vary the MC to induce an open-ended process that continually produces novel artifacts. The hypothesis of the present work is that *coevolution* may help to prevent evolutionary stagnation by forcing the two populations to satisfy their MC with respect to each other, while both continue to drift within the confines of their fixed reproductive criterion.

In this dissertation, these ideas are incorporated into an algorithm called *minimal criterion coevolution* (MCC), and are first evaluated in a maze navigation domain where a population of mazes are coevolved with maze-navigating agents (controlled by artificial neural networks). The MC for this domain requires that each maze be solved by at least one agent and each agent is able to solve at least one maze; any maze or agent that meets their respective constraint is permitted to reproduce. The results demonstrate that MCC is able to produce many different solutions (agent trajectories) to a range of different mazes within a single run, even in the absence of a behavioral descriptor or an archive.

While these results are encouraging and hint at the open-ended capacity of MCC, they stop short of revealing the full extent of its capabilities. In the initial maze domain, MCC rapidly exhausts the representational capacity of the mazes, driving them to maximum density and complexity long before the end of any given run. This raises the interesting question of what maze and agent pairings MCC would be able to discover if mazes were not fixed in size, but were allowed to expand unboundedly throughout evolution, creating increasingly difficult challenges.

Thus we introduce such an enhanced maze domain to provide insight into what happens when *both populations* in MCC are allowed to evolve and complexify without bound. An important facet of this experiment is that there are few simple benchmarks that enable evaluating methods like MCC

aimed at open-ended discovery, and accordingly also no baselines with which to compare any future such approach. By introducing an enhanced maze encoding that allows realistic mazes to increase in size and complexity indefinitely, and by quantifying how MCC behaves in such an unbounded domain, a new benchmark is established that can serve as a foundation for inspiring and measuring future progress in the field. It also offers a window into one of the first experiments of its type, where open-endedness is pursued in an unbounded domain that is not an alive world. The results show MCC exploiting the unbounded domain to produce larger and more complex mazes throughout each run, while maintaining a breadth of maze sizes and path configurations, each with effective solutions.

The maze navigation experiments provide compelling evidence of MCC’s ability to produce open-ended, coevolutionary divergence by imposing a simple reproductive criterion. The intuitive appeal of MCC is rooted in its simplicity as compared to other QD algorithms; however, MCC’s original method of diversity preservation remains ad-hoc and without natural precedent. Like many other evolutionary algorithms (EAs), MCC as described so far uses speciation to group individuals based on genetic similarity [28, 94, 118, 152]. A compatibility function is used to determine whether individuals belong in the same species, thereby requiring the algorithm designer to specify the dimensions by which to characterize genetic distance. Though speciation is a well-established ecological phenomena, its implementation in MCC is contrived and determining how to meaningfully encode and compare similarity imposes a priori structure on a system that is intended to discover novel and unanticipated artifacts with minimal evolutionary bias.

Nature, however, does not explicitly measure and compare genetic similarity to maintain biodiversity; instead, diversity is a byproduct of finite, yet essential resources that constrain niche carrying capacities and induce an *adaptive radiation* [138, 163, 164] that encourages divergence to new environmental niches. Based on this insight, an alternative (and much simpler) form of diversity preservation through resource limitation is evaluated. As with prior MCC experiments, the maze

domain is used to demonstrate a coevolutionary complexification between mazes and agents; however, rather than segregating the two populations by genetic similarity, a resource limit is imposed on mazes such that a given maze can only be used (i.e. successfully navigated) a finite number of times for satisfying an agent’s MC. The results demonstrate that enforcing resource limitation boosts population diversity by producing mazes with highly variable solution paths, while also accelerating evolution by exposing agents to a broader array of new and challenging environments. Resource limitation ensures that evolution remains unencumbered by potentially biased, a priori assumptions by alleviating the need to explicitly characterize and compare solution structure, while also facilitating the application of MCC to new domains for which such characterizations may be non-trivial or computationally-expensive.

While the maze domain is a simple and convenient experimental test bed, MCC has a far wider range of more compelling and potentially real-world applications. To evaluate the extensibility of MCC beyond that of benchmark domains, it is applied to an evolutionary robotics (ER) domain wherein morphology (body) and control (brain) are coevolved. In most ER research, body designs are crafted by hand and controllers are evolved to effectively ambulate a fixed morphology [24, 47, 87]. Recent non-objective search research demonstrated the ability to discover a diversity of voxel-based morphologies, but with simple, static control mechanisms based on a fixed pattern of per-voxel, volumetric actuation [61, 98]. In a notable exception to the aforementioned one-sided optimization methods, Cheney et al. [17] introduced a fitness-based approach for co-optimizing morphology with control in a voxel body locomotion domain. This work provides original inspiration for the body-brain coevolution domain introduced in this dissertation, and the results of MCC are compared to that of co-optimization, demonstrating that MCC can be an effective choice for complex coevolutionary tasks.

In the body-brain coevolution domain, three-dimensional voxel-based robot bodies that consist of active (i.e. muscle) and passive elements are coevolved with artificial neural networks (NNs)

that control expansion and contraction of each active voxel. To meet their MC, NN controllers must ambulate the body a minimum distance within a fixed simulation time, while bodies must be successfully ambulated by at least one NN. The composition of voxel bodies can vary with regard to voxel placement (i.e. the voxel lattice may be only partially full) and active vs. passive voxel ratios. Like the enhanced maze domain where mazes expand to complicate navigation, bodies can grow along all three dimensions, forcing controllers to account for altered physical dynamics.

The results show that MCC discovers a diversity of morphologies that vary in both size and voxel composition, along with NN controllers that produce successful ambulation. Voxel bodies continue to expand throughout evolution, signifying an ongoing coevolutionary adaptation to increasingly complex challenges. These experiments constitute the first application of an open-ended coevolutionary search process to a non-trivial task of practical interest, thereby suggesting the generality and extensibility of MCC to a variety of interesting new problems and domains.

MCC constitutes the first method of producing an open-ended coevolutionary search dynamic within a broadly-applicable algorithmic framework. In its final incarnation, MCC requires no measure of fitness, no ranking and no measure of genetic, phenotypic or behavioral diversity, making it one of the simplest algorithms in EC, yet one that is able to continually produce unbounded diversity and complexity – a trait exhibited by no other algorithm to date. At the same time, the intent of the maze domain is to offer an experimental test-bed to the EC and alife communities that can be harnessed for scientific inquiry into the nature of open-endedness and the conditions that enable unbounded discovery, while also providing a practical method for approaching open-ended design problems that extend on the MCC framework. The body-brain coevolution domain demonstrates that MCC is capable of scaling beyond simple benchmarks and tackling difficult, real-world tasks at the forefront of research and development.

In summary, the hypothesis of this dissertation is that genetic drift can be harnessed to facilitate

divergence in two coevolving populations, while an MC on reproduction channels drift along desirable and non-trivial evolutionary trajectories as both populations interact and increase in complexity to satisfy a continuously shifting challenge to maintain viability. This hypothesis is supported by the following key contributions:

1. A new algorithm called minimal criterion coevolution (MCC) is introduced. MCC coevolves two populations, each subject to an MC of reproductive viability. Both populations are driven to continually diverge and complexify as they simultaneously satisfy a mutual minimal criterion (MC) of reproductive viability.
2. MCC is validated in an evolving maze domain, where maze solution paths are made more difficult by adding and shifting walls, and agents controlled by artificial neural networks are evolved to solve increasingly more difficult mazes.
3. An enhanced maze domain is introduced, where maze boundaries are expanded and the solution path is directly-evolved, producing more challenging navigation environments with larger mazes and convoluted solution paths, and effective agent solutions.
4. Resource limitation is introduced as an alternative form of diversity preservation in MCC, thereby avoiding speciation-induced bias that results from characterizing and comparing an a priori formalization of genetic distance.
5. MCC is demonstrated in a difficult body-brain coevolution task, where it discovers multiple diverse morphologies along with artificial neural networks capable of controlling each

The next chapter of this dissertation introduces the field of EC along with common diversity preservation techniques, and provides historical context around the pursuit of open-endedness. Two evolutionary methods on which MCC relies, coevolution and neuroevolution, are discussed, followed

by an overview of compositional pattern producing networks (which are used in the body-brain co-evolution experiments) and concluding with a brief survey of quality diversity algorithms, and their connection to non-objective and open-ended search. Chapter 3 then details the main contribution of this dissertation, the MCC algorithm, and describes its implementation.

The initial maze domain encoding along with experiments and results is described in chapter 4, while chapter 5 describes the enhanced maze domain and demonstrates MCC's ability to exploit larger and more complex mazes. Chapter 6 outlines how population diversity can be maintained through resource limitation, comparing the results to speciation in the enhanced maze domain. MCC is applied to a challenging body-brain coevolution task in chapter 7 where it discovers novel morphologies and complex control strategies. Chapter 8 then discusses the contributions of MCC and the experimental domains, along with future research directions, and finally chapter 9 concludes with the implications of the work presented in this dissertation.

CHAPTER 2: BACKGROUND

This chapter provides a brief overview of the field of evolutionary computation (EC) and some common diversity preservation methods used within EC, followed by an introduction to the concept of open-endedness and the pursuit of open-ended evolution within the artificial life community. Two EC methods that are of particular relevance to MCC, coevolution and the NEAT algorithm, are then reviewed, followed by an overview of compositional pattern producing networks (CPPNs), which are used to generate regular body morphologies and controllers in the experiments in chapter 7. The chapter concludes with an introduction to non-objective search algorithms followed by a discussion of an emerging field inspired by both EC, open-ended evolution and non-objective search known as quality diversity.

2.1 Evolutionary Computation

Evolutionary biologists have long employed computational frameworks to model and investigate various theories of natural evolution [43, 63, 115]. In 1958, however, Richard Friedberg proposed a novel approach for harnessing computational abstractions of evolution to automatically develop, or *evolve* computer programs [44]. This idea was validated in 1962 when Hans-Joachim Bremermann demonstrated how a simulated evolutionary process could be applied to numerical optimization problems [9, 10]. The latter half of the 1960s and early 1970s heralded the birth of the field of EC, with three foundational EAs: evolution strategies (ES) [125, 140], evolutionary programming (EP) [40, 41] and genetic algorithms (GAs) [9, 66] being introduced within a decade of each other.

As both the name and the history of the field suggest, EC draws inspiration from natural evolution. It is well-suited to problems that exhibit large search spaces and require efficient methods of

exploration and exploitation. EAs maintain a population of candidate solutions that are encoded as genotypes (usually represented as a string of bits or real numbers) and subjected to a series of variational operations, including mutation and crossover. Mutation randomly perturbs one or more genes in the genotype, while crossover combines genes from two parent genotypes. Genotypes are decoded into phenotypes, which are domain-specific functional representations that can be evaluated with respect to some objective measure of performance, known as the fitness function. Individuals undergo a pruning process called selection (analogous to natural selection in Darwinian evolution) wherein only the highest performers with respect to the fitness function are permitted to survive and produce offspring, which are modified and evaluated in like manner [168].

This process of reproduction, variation and selection is intended to iteratively converge on a globally optimal solution; however, such an expectation is often overly optimistic in search spaces that are multi-modal or exhibit deceptive characteristics (i.e. fitness gradients that appear promising but ultimately dead-end in sub-optimal areas of the search space) [31]. A fundamental shortcoming of objective-driven optimization is that the stepping stones toward a globally optimal solution tend to elude the fitness function in deceptive domains because they may lead search through areas that initially appear sub-optimal [80, 83].

2.2 Diversity Preservation

Most evolutionary algorithms (EAs) model search as a *convergent* process that selects only the most fit individuals for reproduction, iteratively honing-in on high-fitness areas of the search space [132]. As a population-based search heuristic, however, EAs rely on maintaining a diverse sample of candidate solutions to avoid convergence to areas of the search space that are only locally optimal. Diversity preservation techniques are loosely based on niching in natural evolution, where organisms are segregated into distinct species and competition occurs primarily within the assigned

niche rather than at a population level [134, 167]. In EC, such methods are employed to manage the conflicting goals of exploration and exploitation, ensuring that multiple evolutionary lineages are maintained to avoid naively committing to a single evolutionary path without adequately vetting the alternatives [94].

Conventionally, similarity is assessed at the genotype-level where individuals are grouped based on age [67, 139], fitness [68] or structural similarity [50, 94]. Hornby [67] introduced a method called Age-Layered Population Structure (ALPS) which characterizes genotypes by the age of their genetic material. Randomly-generated individuals are injected into the population at regular intervals to boost population diversity, but only compete with genomes of a similar age. The intuition is that younger genomes may be more promising in the long run but have had less time to adapt than older individuals, putting them at a selective disadvantage. By preventing direct competition, ALPS maintains higher population diversity throughout the course of a run, thereby reducing the probability of early convergence.

Hutter and Legg [68] proposed a diversity preservation approach called fitness uniform selection scheme (FUSS) that increases selection pressure in sparsely populated fitness regions. FUSS preferentially selects for individuals that exhibit a minority fitness score over those that have a higher, but over-represented fitness, thereby encouraging genetic diversity by maintaining a range of performance profiles.

The Neuroevolution of Augmenting Topologies (NEAT) method [152] (introduced in section 2.5) measures structural similarity, speciating genotypes by quantifying shared evolutionary lineage between each topological component (nodes and connections) – a form of explicit fitness sharing [50]. As with ALPS, the intent of speciation in NEAT is to protect recent innovations by restricting competition to occur only between those of similar evolutionary descent, giving less mature individuals time to adapt within their niche before competing against the population at large.

Speciation and other niching methods encourage diversity in the genotype space, but without regard for whether genetic disparity translates into meaningful phenotypic differences. However, recent QD research has found that multiple distinct genotypes may collapse into the same phenotypic expression, suggesting that augmenting genetic speciation with an explicit search for behavioral novelty may be more effective at preserving *functional* diversity in deceptive or ambitious domains [78].

Nature, however, does not explicitly measure genotypic or behavioral similarity. Instead, diversity is a byproduct of finite, yet essential resources [85, 134], an insight motivates the present investigation of diversity preservation through resource limitation.

2.3 Open-endedness

The concept of open-endedness has its roots in *alife* – a discipline concerned with analyzing and understanding natural systems and the process by which individuals within those systems interact and evolve [6, 124, 161]. Natural evolution is often viewed as an unguided process, yet one that continually produces novel and increasingly complex artifacts, a characteristic referred to as *open-ended evolution* [91, 133, 146]. Just as artificial general intelligence (AGI) is considered the pinnacle of AI [2, 18], formalizing and reproducing (in a simulation) the dynamics that quantifiably lead to an open-ended evolutionary process is a significant objective of *alife* [146, 155].

While there is a distinct lack of consensus regarding precise methodologies for defining and measuring the characteristics of an open-ended evolutionary process, it is generally agreed that such a process produces novel, functional and increasingly complex forms in perpetuity [1, 86]. Artificial life researchers often develop *alife* worlds [13, 57, 100, 101, 113, 122, 124, 143, 172] as a convenient, visually-oriented method for exploring the effects of various conditions and constraints

imposed upon simulated evolutionary processes. The intent of these artificial worlds is to better understand and inform hypotheses about the ingredients required to generate and sustain life, often leveraging diversity of life on earth (which is generally considered an open-ended system [97, 136]) as inspiration.

Soros and Stanley [144] recently proposed a set of such ingredients hypothesized to be necessary for the generation of an open-ended evolutionary process, and created an alife world, dubbed Chromaria, to test them. Central among these conditions is the enforcement of a minimal criterion that places a lower bound on individual complexity and prevents the population from falling into a degenerate state of uninteresting and trivial behaviors. Additionally, individuals must interact to satisfy the MC, which allows its difficulty to vary organically over time. On Earth, the MC is self-replication, which requires the development and persistence of a reproductive apparatus and the ability to survive long enough to employ it. These are non-trivial capabilities that require a similarly advanced level of developmental sophistication. Unlike prior alife worlds, Chromaria does not attempt to preferentially select for individuals exhibiting higher fitness or merit scores; rather, everyone who satisfies the MC is allowed to reproduce. Interestingly, constraining evolution through the MC rather than a fitness score helps prevent evolutionary convergence, maintains population complexity (through an interaction requirement) and promotes an open-ended dynamic. The present work demonstrates how such an open-ended process can be encapsulated into a general-purpose algorithm and applied outside of an alife domain.

2.4 Coevolution

In traditional EC, the fitness function is typically defined as a global, extrinsic measure of performance; however, in some domains such an absolute performance metric may be prohibitively expensive to compute or impossible to formalize. The field of coevolution addresses these short-

comings by defining fitness as a relative measure based on interaction between individuals within a single population, or between two separate populations [12, 117]. Coevolution is traditionally divided into two methodologies: competitive coevolution and cooperative coevolution. Competitive coevolution pits self-interested individuals against each other, and meets out reward based on the extent to which one individual outcompetes the other (i.e. the fitness scores for competing individuals are inversely related) [131]. A canonical example of competitive coevolution is the predator-prey scenario in which a predator attempts to capture and devour their prey while the prey attempts to escape from the predator – two conflicting objectives where one individual is rewarded at the expense of the other [109]. By contrast, in cooperative coevolution, individuals work together to achieve an overarching objective. The problem space is divided into subcomponents with individuals, or “species”, specializing in each and working collaboratively to achieve a larger goal [119, 169, 170].

An interesting property of coevolution is its theoretical ability to produce a never-ending arms race wherein subjective notions of performance and competitive or cooperative interactions allow individuals leeway to evolve increasingly diverse and complex behaviors [38, 109, 154], a property that is itself similar to the dynamics that would be observed in an open-ended system. In practice, however, this dynamic has proven difficult to sustain. Instead of producing creative interactions or diverse behaviors in perpetuity, coevolutionary algorithms eventually converge to mediocre stable states, wherein sub-optimal strategies are discovered and individuals lack a fitness-based incentive to further evolve [38].

Moran and Pollack [102] recently showed that mixing cooperative and competitive interactions can help drive complexity growth and avoid simple equilibria; however, solution quality is computed based on a historical record of actions (referred to as “strategies”) and subsequent results, which limits unbounded growth in much the same way as the computational expense imposed by comparing new discoveries to a behavioral archive. De Jong [30] also proposed a method that

maintains an archive of non-dominated (i.e. pareto-optimal) solutions, but the scalability limitations imposed by comparison against an ever-expanding archive still hold. Moreover, both of the aforementioned methods explicitly measure and reward solution quality, incentivizing both populations to converge on narrow, high-performing regions of the space, thereby pruning diversity rather than encouraging it. Additionally, recent work in applying novelty to search to coevolution has suggested that replacing fitness with a novelty metric may help minimize coevolutionary stagnation in both a competitive and cooperative context by maintaining healthier levels of diversity [53, 54]; however, these methods still require that behavioral novelty is explicitly characterized and compared to promote coevolutionary divergence.

Coevolutionary interactions play a fundamental role in the present work; however, such interactions are neither competitive nor cooperative. Instead, individuals are permitted to interact freely within the confines of their MC, thus promoting a divergent search dynamic that is intrinsically immune to evolutionary stasis.

2.5 Neuroevolution of Augmenting Topologies

Complexity and the perpetual increase thereof is a hallmark feature of open-ended systems [4, 144, 146]. Algorithms that are developed with the intent of producing an open-ended dynamic must therefore possess an extensible representational capacity that scales as evolution progresses and discovers increasingly complex artifacts. Experiments in this dissertation involve agents that are controlled by evolved neural networks; specifically, using the Neuroevolution of Augmenting Topologies (NEAT) algorithm [152].

Neuroevolution is a subfield of artificial intelligence that uses evolutionary algorithms to evolve the weights, and often the topology, of a population of artificial neural networks [39, 149, 173]. NEAT

is a neuroevolution method that incrementally adds structure (connections and nodes) to a NN – a process known as *complexification*. Evolution starts with a population of single layer NNs, each with an input layer that is fully connected to the output layer (i.e. no hidden layer). Each structural component is encoded as a gene in the NEAT genome and assigned a unique historical marking. This marking simplifies comparison of genetic similarity between networks, thereby mitigating the destructive potential of crossover (a pervasive problem in neuroevolution prior to NEAT) and facilitating the subdivision of networks into groups or *species*. Speciation is a method of protecting networks that have undergone structural modification so that they have adequate time to become well-adapted, thereby reducing initial competition and promoting increased population diversity.

The complexification operations in NEAT are especially relevant to its use in open-ended search because adding network structure increases the network’s free parameters (i.e. connection weights), which is akin to increasing its representational capacity. In a robot control domain, the level of representational capacity is proportional to the complexity of behaviors and control strategies that the network is capable of learning. Given that an open-ended search method should be capable of discovering artifacts of unbounded complexity, the ability to dynamically and principally add structure to meet representational demands is required to avoid evolutionary stagnation.

2.6 Compositional Pattern Producing Networks

Most neuroevolution methods, including NEAT, employ a *direct encoding* [152], meaning that each gene in the genotype maps to a single structural component in the phenotype. Despite their ubiquity in the neuroevolution literature, it is well-accepted within the field of developmental biology that the human brain and other natural structures are the byproduct of a highly-compressed indirect mapping that exploits gene reuse, producing geometric regularities, such as symmetry and repetition, to enable encoding-efficiency [33]. For example, the human brain contains 100 trillion

connections, but is fully-described by a genome with only 30,000 genes [34, 177].

Artificial developmental encodings are a type of indirect encoding rooted in developmental biology, where the phenotype is gradually assembled from a lower-dimensional genotype through an embryonic development process called embryogenesis [3, 171]. During embryogenesis, local chemical interactions produce morphogen concentration gradients that differentiate tissue cells, describing general body plans and situating high-level organismal components. For example, in many animals, the dorsal-ventral and anterior-posterior axes are established by initial morphogen gradients [166, 171], and further elaborations operate within this frame-of-reference, thereby generating arbitrarily complex spatial patterns.

Various artificial abstractions of development have been proposed [26, 147, 153], many of which attempt to directly simulate low-level chemical dynamics that occur during growth [25, 174, 175]. Stanley [148] offered a unique perspective on development that preserves the essential characteristics of developmental encodings while skipping the costly and time-consuming development process. This approach, called compositional pattern producing networks (CPPNs), can encode regular patterns in substrates of arbitrary dimensions. In chapter 7, a CPPN encodes material properties at each location in a three-dimensional voxel substrate, while also a separate CPPN specifies the weights of a distinct NN controller that dictates actuation characteristics of each voxel.

The key insight is that development can be described by composing functions that generate geometric patterns that are similar to gradient patterns observed in developing embryos. For example, a Gaussian function is symmetric about its center, thereby describing the concept of bilateral symmetry when positioned along an anterior-posterior axis. Similarly, periodic functions, such as sine and cosine, generate repeating patterns which can encode concepts like body segmentation.

A CPPN, which is a directed graph, composes a set of canonical functions through weighted connections that control the degree of influence exerted by each transformation. Conveniently, CPPNs

are structurally similar to NNs, meaning that neuroevolution algorithms intended to evolve NNs can be used to evolve CPPNs with few modifications. CPPN-NEAT is one such method that uses the NEAT algorithm to evolve CPPNs by extending it to encode hidden-layer nodes with varying activation functions (conventionally, only a single activation function is used). When adding a new node, an activation function is randomly selected from a predefined set of canonical functions (which typically includes Gaussian, sigmoid, sine, cosine and other symmetric, periodic or linear functions).

Conventionally, CPPNs are applied over a spatial substrate, taking as input the coordinates for each location along with a radial distance from the center and a bias, and producing a phenotypic representation of that location. Given their propensity for creating regular spatial patterns, CPPNs are often demonstrated in creative visual domains, such as genetic art [141, 176], where CPPNs can effectively encode images. Pixels coordinates on an artificial canvas are queried and the output is interpreted as a pixel shade or color value.

2.7 Non-objective Search

The field of EC has traditionally focused on fitness-driven optimization, where the quality of candidate solutions are assessed based on their distance to an objective. An iterative process of reproduction, evaluation and selection is intended to drive the population to convergence on the optimal solution as defined by the objective function. This approach is inspired by the process of natural selection, where individuals survive and propagate their lineage based on their degree of environmental adaptation. However, while framing biological evolution as an optimization process is a convenient way to reason about local adaptations, the whole of evolution has significantly more profound implications. Instead of converging on a single, globally optimal solution, natural evolution has produced a multitude of diverse species, each well-adapted to their respective niche. It

was this insight that gave rise to a new class of *non-objective* algorithms that explore the search space without the guidance of fitness.

Non-objective search methods promote divergence, discovering diverse solutions to a given problem rather than converging on the globally-optimal according to a narrow definition of fitness. One of the first methods that proposed such an alternative approach to search was an algorithm called viability evolution (ViE) [92, 95]. ViE imposes a set of thresholds, or *viability boundaries*, that individuals have to meet to survive and reproduce. At the beginning of evolution, viability boundaries encompass the entire population, but are incrementally tightened as evolution progresses, spurring the removal of individuals who no longer satisfy the more stringent constraints. This process repeats until the algorithm has converged on a set of individuals who satisfy all of the viability constraints. A convenient property of ViE is that it requires no explicit objective function, which is particularly advantageous for multiobjective problems where objective weighting has a significant impact on solution characteristics. Moreover, the loose criterion of viability allows ViE to maintain more diversity and sample a broader range of the search space. While MCC draws original inspiration from ViE, non-objective algorithms are fundamentally different in that they have no drive toward convergence.

Novelty search [76, 78, 80] was the first of such non-objective algorithms and represented a radical departure from prior methods. Rather than navigating a search space based on the path of increasing fitness, novelty search rewards solutions that are behaviorally different than what it has seen in the past (as compared to an archive of novel behaviors), without regard to objective performance. While such an approach may initially seem inefficient, it turns out to be effective for ambitious objectives in which the search space exhibits *deceptive* characteristics. In these domains, following the gradient of fitness often leads search into inescapable traps that fall short of the overall objective. Fitness-based search fails because the stepping stones that lead to the global optimum may initially pass through low fitness regions of the search space, and therefore be unreachable by

a fitness-driven trajectory. The key insight is that searching for novelty without regard to fitness induces a *divergent* search process that “pushes outward,” allowing more stepping stones to be sampled and illuminating trajectories that were previously shrouded in deception.

A common criticism of non-objective search methods is their susceptibility to expending resources in uninteresting areas of potentially vast behavior spaces [22, 72]. To address this concern and to better align novelty search with selection processes observed in natural evolution, an extension called minimal criteria novelty search (MCNS) was introduced [77]. MCNS imposes boundaries on the behavior space which define the minimal criteria for reproduction. Individuals who traverse beyond those boundaries fail to meet the MC and are therefore ineligible to reproduce. This avoids propagating a lineage of individuals who would explore areas of the search space orthogonal to those of interest. While the present work preserves the notion of an MC, it demonstrates how the MC alone is capable of producing open-ended evolutionary artifacts. Moreover, in contrast to prior non-objective algorithms, MCC functions without the added overhead of viability boundaries and without the need for a novelty archive, or even a behavior characterization.

2.8 Quality Diversity

While non-objective search methods, like novelty search, are effective at avoiding deception by ignoring uninformative fitness gradients, the outcome of search is still interpreted with regard to a global performance criterion, and the stepping stones discovered along the way are discarded. A new class of algorithms have emerged that build on the novelty paradigm, but recognize the intrinsic value of intermediate discoveries, optimizing within each diverse niche in much the same way as nature discovered a diversity of locally-adapted organisms. The goal of these algorithms is to collect *quality diversity* (QD): a set of diverse solutions that are each as high-performing as possible with respect to a measure of performance [123].

Lehman and Stanley [79] introduced the first QD algorithm, called novelty search with local competition (NSLC), which combines a novelty objective with a local competition of objective that rewards phenotypically-similar individuals based on their comparative performance. NSLC was originally demonstrated in a virtual creature evolution domain where it evolved a diverse array of morphologies along with effective walking gaits for each distinct body plan. It has also excelled in other robot locomotion domains, discovering diverse and performant walking gaits for hexapod robots [23] as well as bipedal gaits for humanoid locomotion [19]. NSLC has also shown compelling application outside of robot locomotion, finding novel methods of resource sharing in swarm coordination tasks [52] and discovering diverse but informative features for image classification [159].

Mouret and Clune [106] later demonstrated an alternative method of QD, dubbed the Multi-dimensional Archive of Phenotypic Elites (MAP-Elites), where the behavior space is discretized along each specified dimension of variation, creating a map in which every cell contains the highest-performing individuals for that behavioral intersection. Unlike NSLC, which uncovers behavioral niches during search, MAP-Elites imposes an explicit structure over the behavior space, which may be beneficial in scenarios where an experimenter has a preconceived idea of the relevant aspects of behavior. Like NSLC, MAP-Elites has shown promising application to robot locomotion [37, 106, 111], including the ability to effect resilient control in a novel damage-recovery task [24] where the behavior map stores a diversity of pre-evolved walking gaits which are used to guide a trial-and-error search to rapidly evolve compensatory strategies that function in spite of the damage. MAP-Elites has also been applied to generative design tasks, discovering diverse and efficient aerodynamic designs [45, 46], generating novel and aesthetically-pleasing images [108] and producing textured, three-dimensional printable structures [84].

While QD algorithms represent the state-of-the-art in divergent search, any analogy to nature’s propensity for uncovering functional novelty remains limited. Like novelty search, all QD al-

gorithms introduced to date require a behavior characterization (BC), which is used to quantify behavioral attributes of the phenotype so that divergence can be promoted by rewarding behavioral novelty. Moreover, defining a BC requires an algorithm designer to specify the relevant dimensions of behavioral novelty a priori, which may fundamentally limit evolutionary discovery. For example, MAP-Elites is restricted to behaviors that can be encoded in the behavior map; once all cells in the map are full, MAP-Elites can only improve upon existing solutions, but can't discover novel individuals along undefined behavioral axes. MCC does not attempt to compare or structure behaviors, and is therefore immune to such artificial scalability constraints.

CHAPTER 3: APPROACH: MINIMAL CRITERION COEVOLUTION

While the artifacts of evolution on earth are clearly not the result of a convergent process, there is also no evidence that natural systems exhibit an explicit preference for behavioral novelty. Instead, biological evolution enforces a criterion of reproductive viability, but organisms or species may discover a multitude of ways in which to meet that constraint and have the potential to evolve their own unique route to the MC. For example, the developmental path toward reproductive viability for bacteria tends to be much shorter and less complex than for mammals. Furthermore, if populations were evolving in isolation, their MC may remain static; however, in nature, organisms are influenced by their environment and other populations with which they are coevolving. These coevolutionary interactions result in an on-going flux of species' path toward the MC.

Inspired by this perspective of evolution, Minimal Criterion Coevolution (MCC) [7] is introduced as a dual-population coevolutionary algorithm, wherein each population is evaluated in accordance with a domain-specific MC. Importantly, MCC does not align with either of the traditional categories of competitive [131] or cooperative [169] coevolution. It does not evaluate and rank individuals based on one's ability to outcompete the other, nor does it grade according to the extent individuals cooperate to achieve an objective. Instead, individuals are permitted to reproduce based solely on their ability to satisfy the MC. A notable characteristic of this binary evaluation criteria is that it imposes no ranking among individuals from either population (they either satisfy the MC or not), thereby necessitating a method of selection that is free of bias. One such approach that has proven effective in alife domains is storing each population in a fixed-size queue, similar to the "parent queue" in the alife world of Chromaria [144]. The parent queue is a circular queue in which operations are performed in a "first in, first out" (FIFO) manner. It stores individuals who satisfy the MC in the order of insertion, and maintains a queue pointer that points to the location of the next individual in line for reproduction. If the pointer reaches the end of the queue, it will

simply loop back around to the beginning. Additionally, if insertion of a new individual were to exceed the queue's capacity, the oldest individual is removed from the queue to make room. This process ensures that every individual who satisfies the MC gets *at least one* chance to reproduce. In MCC, each population has its own queue, thus the parent queue is rebranded as the *population queue*. A key feature of the queue data structure is that it maintains a flat, unranked representation of the population, and therefore introduces no selection bias (apart from MC satisfaction), thereby allowing evolution to maintain many divergent paths within the confines of the MC.

It is common practice for evolutionary algorithms to begin evolution with a randomly initialized population; however, the requirement that each population queue contains only individuals who have satisfied the MC necessitates special consideration for the MCC initialization process. In particular, it is unlikely that randomly generated individuals will be capable of satisfying a non-trivial MC, meaning they would be denied admittance to any of the population queues and the evolution process would be a non-starter (i.e. the population queues would be empty). To avoid this causality dilemma, MCC must undergo a bootstrap process wherein the requisite number of individuals are pre-evolved and used to seed the respective population queues. In principle, any evolutionary algorithm (including fitness-based algorithms) could be used to pre-evolve the requisite seed genomes; however, the present implementation of MCC uses novelty search due to its propensity for discovering diverse solutions and therefore producing a more diverse seed population with which to begin evolution.

The selection process employed by MCC closely resembles that of a steady-state evolutionary algorithm; however, while steady state algorithms typically evolve the population in a serial manner (i.e. only one offspring is produced and evaluated on each iteration), MCC is easily parallelizable because there is no fitness score to evaluate and compare. More concretely, MCC's absolute measure of performance (i.e. the MC) allows large chunks of the population to be batched and evaluated in parallel. This design feature facilitates a distributed execution paradigm wherein eval-

uations are spread across multiple nodes, allowing MCC to scale when simulating evolutionary processes. Algorithm 1 formalizes the MCC selection, evaluation and removal process in detail.

Algorithm 1 MCC Evaluation Process

Require:

batchSize - # of individuals to evaluate simultaneously
numSeeds - # of seed genomes to evolve that satisfy the MC

▷ Evolve seed genomes that satisfy MC

randPop \leftarrow *GenerateRandomPopulation*()

viablePop \leftarrow *EvolveSeedGenomes*(*randPop*, *numSeeds*)

loop

▷ Reproduce children and add parents back into queue

parents \leftarrow *viablePop.Dequeue*(*batchSize*)

children \leftarrow *Reproduce*(*parents*)

viablePop.Enqueue(*parents*)

for all *child* \in *children* **do**

▷ MC involves interaction with the other coevolving pop.

mcSatisfied \leftarrow *EvaluateMC*(*child*)

if *mcSatisfied* **then**

viablePop.Enqueue(*child*)

end if

end for

▷ Remove oldest if queue capacity exceeded

if *viablePop.Size* > *viablePop.Capacity* **then**

numRemovals \leftarrow *viablePop.Size* - *viablePop.Capacity*

RemoveOldest(*viablePop*, *numRemovals*)

end if

end loop

While MCC has no explicit objective or mechanism that biases selection toward behavioral novelty, it is still likely, only if by random chance, that some individuals will be better at producing competent offspring than others. Over time, this reproductive disparity could implicitly bias selection toward lineages that demonstrate a higher level of competency, resulting in a coevolutionary convergence. In nature, organisms are segregated into species, each of which occupy a particular

ecological niche [134, 167]. Every niche has a finite carrying capacity, which imposes a form of local regulation on the population size of the occupying species, thereby encouraging species to diverge and found a wide array of diverse niches. Genetics-based speciation is a long-standing, popular method for diversity preservation in EAs [32, 94].

In addition to the base MCC algorithm, speciation can be optionally implemented as a method of diversity preservation within the population queues. While individuals within a given population remain physically stored together in their respective queue, they are clustered into separate logical groups, or *species*, based on a measure of their genetic similarity. At the beginning of evolution, the seed genomes of each population constitute the initial centroids of their respective clusters, and new additions to the queues are speciated based on genetic distance to the seed genomes. Additionally, the queue capacity is evenly distributed among species such that each species has a maximum size given by:

$$\text{capacity}(i) = \frac{n}{s}, \quad (3.1)$$

where i is the species under consideration, n is the number of individuals in the population and s is the number of species.

The speciated variant also effects a slight modification to the selection process. Rather than selecting the next batch of individuals according to their queue order, an equivalent number of individuals are selected from each species for reproduction (though queue insertion order still dictates order of selection within the species). Following evaluation, the offspring who satisfy the MC are added to their respective population queue and assigned to the species with whom they share the greatest genetic similarity (i.e. they are respeciated). If any of the species exceed their capacity as a result of speciating the offspring, the oldest individuals *assigned to those species* (rather than the oldest

in the queue) are removed, thus ensuring that the queue remains at or below capacity. Algorithm 2 formalizes the speciated MCC selection, evaluation and removal process. This method of speciation is conveniently lightweight because distances are computed only at the genotype level, which avoids the more expensive process of decoding to the phenotype or characterizing and comparing behaviors.

While the next few chapters will investigate the variants of MCC introduced in this chapter, later in chapter 6 an additional variant based on resource limitation will be introduced as an alternative to the speciated variant.

The intent is for MCC to induce divergence in both populations through a process of *genetic drift*, but the MC is utilized to guide drift in directions of desirable or non-trivial interactions, yielding a continuous, virtually open-ended process.

Algorithm 2 MCC Evaluation Process with Speciation

Require:

batchSize - # of individuals to evaluate simultaneously

numSpecies - # of species

▷ Evolve seed genomes that satisfy MC

randPop \leftarrow *GenerateRandomPopulation*()

viablePop \leftarrow *EvolveSeedGenomes*(*randPop*, *numSpecies*)

▷ Seed species with each viable individual as centroid

speciesPop \leftarrow *SeedSpecies*(*viablePop*)

loop

▷ Reproduce children from each species and add parents back into queue

for all *species* \in *Species* **do**

▷ Produce offspring from selected parents in current species

parents \leftarrow *speciesPop.Dequeue*(*batchSize*)

children \leftarrow *Reproduce*(*parents*)

▷ Reinsert parents into queue

speciesPop.Enqueue(*parents*)

for all *child* \in *children* **do**

▷ MC involves interaction with the other coevolving pop.

mcSatisfied \leftarrow *EvaluateMC*(*child*)

if *mcSatisfied* **then**

viablePop.Enqueue(*child*)

end if

end for

end for

▷ Respeciate based on addition of viable children

viablePop.Respeciate()

▷ Remove oldest if species capacity exceeded

for all *species* \in *Species* **do**

▷ Remove oldest from species if species capacity exceeded

if *speciesPop.Size* > *speciesPop.Capacity* **then**

numRemovals \leftarrow *speciesPop.Size* - *speciesPop.Capacity*

RemoveOldest(*speciesPop*, *numRemovals*)

end if

end for

end loop

CHAPTER 4: INITIAL MAZE DOMAIN

Mazes are a common and easily-understood domain where deception and overall task difficulty are easily visualized. This chapter presents an initial maze navigation task (which will be enhanced in future chapters) where MCC evolves increasingly difficult mazes along with maze-navigating agents capable of solving those mazes. The aim of this is to show that MCC is capable of discovering incrementally complex artifacts in both populations, limited only by the evaluation domain, thereby offering hints of a truly open-ended process.

4.1 Experiment

An ideal domain for initially investigating the capability of MCC is one in which non-trivial adaptations and complexity level are explicitly visualized. One such domain that has been used extensively to evaluate non-objective search algorithms are mazes [77, 78, 99, 105, 107, 123]. In particular, maze domains yield an interpretable sense of complexity through the number and configuration of interior walls. They also make search space deception and non-optimal trajectories through the solution space visually explicit in the form of dead-ends and wandering or looping paths, thereby facilitating the identification of principled, non-trivial evolved strategies.

Mazes are also well-suited to depicting the diversity and complexity of navigational strategies evolved by MCC; however, they take on a unique significance in the following experiments. In particular, rather than demonstrating a diversity of solutions to a single maze, MCC produces several different solutions to *many different mazes* of varying complexity. That is, MCC evolves both maze navigating agents along with maze environments. Maze generation is itself an instance of procedural content generation, which is reviewed in [165]. It is important to note, however, that

MCC is able to produce this level of diversity in both populations within the scope of a single run and in the absence of any kind of behavior characterization (BC) or method to compare that BC.

Maze navigating agents are simulated, wheeled robots controlled by evolved artificial neural networks (NNs). The intent is for these agents to learn how to efficiently navigate from the start location of a maze to its ending location. The NeuroEvolution of Augmenting Topologies (NEAT) method (introduced in section 2.5) has a proven track-record in complex control tasks, and robot control in particular. Importantly, NEAT's method of incremental complexification is ideal for demonstrating an open-ended process because it can lead to increasingly complex behaviors as more free parameters are added to the evolved NNs. The architecture of the agent's NN controller is identical to [76], with six rangefinder sensors (positioned at heading offsets of -90° , -45° , 0° , 45° , 90° and 180°) that measure distance to line-of-sight obstructions, and four pie-slice radar sensors that cover the full circumference of the agent and activate when the target falls within the sensors' arc. Two effectors apply forces that turn and propel the agent. Figure 4.1 depicts the agent sensor array and NN topology.

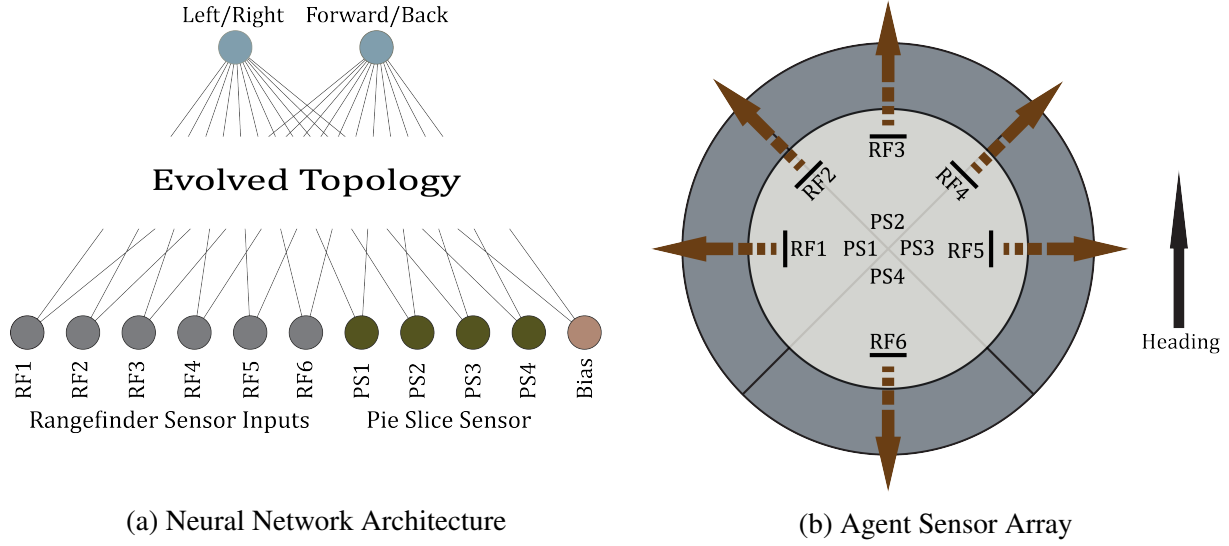


Figure 4.1: **Neural network architecture and sensor array of the maze navigating agent.** The NN that controls the maze navigating agent is shown in (a). The first six inputs receive signal from the agent’s laser range finding sensors, which measure distance to an obstruction, while the next four are fed from the agent’s pie slice sensors, which fire when the goal is within the sensor’s arc. Two output nodes turn and propel the agent respectively. The agent’s sensor configuration is shown in (b). The first five laser range finder sensors are placed at 45° offsets between -90° and 90° , while the sixth is looking behind the agent at 180° . Additionally, four equally-sized pie slice sensors span the circumference of the agent. The arrow to the right of the sensor array indicates the agent’s heading.

The mazes used in this experiment are square grids with a start location in the upper-left corner and target location in the lower-right corner; both points remained fixed throughout evolution. An agent, controlled by a NEAT-evolved NN, begins a trial at the start location and is evaluated based on its ability to reach the target location within a fixed simulation time. The maze consists of multiple internal obstructions (i.e. walls) that impede a direct path from start to finish, thereby necessitating the development of complex, nontrivial navigation strategies.

4.1.1 Maze Evolution

MCC requires two coevolving populations: one consists of the maze-navigating agents while the other is the mazes. Evolving mazes requires that the maze structure be reduced to a genetic encoding on which standard evolutionary operators can be applied. In this experiment, mazes are a collection of walls that are arranged in a manner that permits one possible trajectory from start to finish. The exterior walls bound the maze space while interior walls introduce obstructions, which prohibit a straight-line trajectory and increase the difficulty of solving the maze. Given that the difficulty of navigating the maze is generally proportional to the number of interior walls that it contains, interior wall count is used as an estimate of maze complexity. Additionally, mazes maintain a fixed size throughout evolution, thereby imposing an upper bound on maze complexity that ensures tractability of this initial investigation (though the maze domain enhancements discussed in chapter 5 removes this constraint).

Each wall is encoded as a gene in the maze genome, and is described by its relative position in the maze and the relative position of an opening, or “passage”, within that wall (these are two separate values that combine to form a single gene). A randomly generated, non-evolved orientation seed is also tagged on to each gene to facilitate exact reproduction of each interior wall. Like the NEAT connection gene weights, maze genomes adopt a real-valued encoding in the interval $[0, 1]$ to dictate the placement of both the wall and the passage. Maze genome reproduction is asexual (i.e. crossover is not implemented) and is controlled by three separate mutation parameters: wall mutation probability, passage mutation probability and add wall probability. The *wall mutation* probability adjusts the position of the wall within the maze, while the *passage mutation* probability shifts the position of the passage within a wall. The *add wall* probability is a complexification operation that lengthens the affected maze genome by adding a new wall gene. When the genome reaches maximum complexity, the add wall mutation is disabled. Just as with NEAT, mazes start

with minimal structure (i.e. interior walls) and gradually complexify as evolution progresses.

Maze genotypes are decoded into their equivalent phenotypic representation by iteratively bisecting the maze and scaling the relative wall and passage position to the dimensions of the affected sub-space. The first gene in the genome bisects the entire maze space, creating two sub-spaces with a passage through which the agent can move from one to the other. If the bisection is vertical, the second gene will bisect the leftmost sub-space, creating two additional sub-spaces. Similarly, if the initial bisection is horizontal, the second gene will bisect the uppermost sub-space. The orientation of each wall is dictated by the dimensions of the sub-space that it bisects such that the wall is placed perpendicular to the dimension of greatest magnitude. For example, if the sub-space is taller than it is wide, the wall that bisects that space will be assigned a horizontal orientation. Conversely, if the sub-space is wider than it is tall, the wall will be assigned a vertical orientation. In the event that the sub-space is square, a randomly assigned orientation attached to the applicable gene will dictate wall orientation.

This process of iterative bisection continues until all of the genes have been decoded. The maze generation implementation is inspired by the recursive division algorithm [42, 73], but it uses a breadth-first variant to avoid unbalanced wall placement within mazes that have not yet reached maximum complexity. Figure 4.2 depicts the maze complexification process.

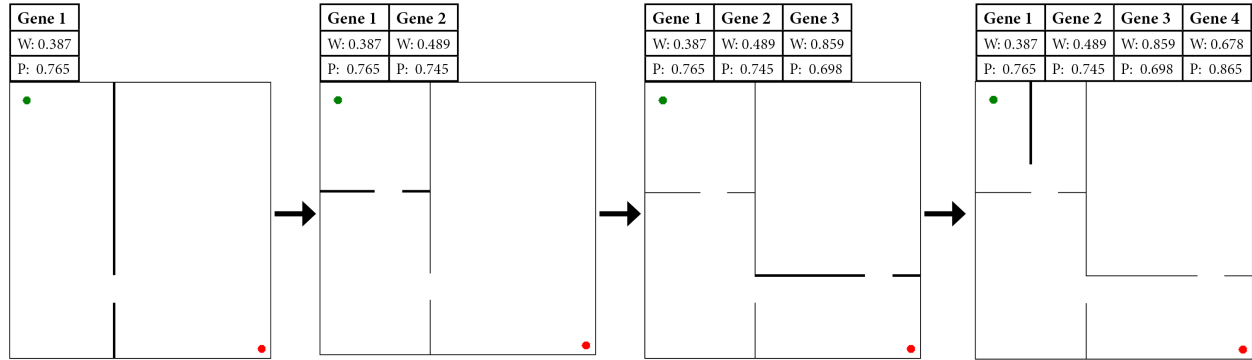


Figure 4.2: **Maze genotype (top) and phenotype (bottom).** An example of the maze complexification process is shown. The table above each maze enumerates the genes within the genome that decode to the phenotype immediately below them. Each gene contains a wall position and passage position, both real numbers in the interval $[0, 1]$. The wall position describes the relative position of a wall within its subspace while the passage position describes the relative position of a passage within that wall. Each gene maps to one wall in the phenotype. The wall that corresponds to the rightmost gene in its genotype is emboldened in the phenotype depiction. The starting position and target position are fixed and are shown in the phenotype at the upper left and lower right respectively.

4.1.2 Maze Domain Minimal Criterion

Soros and Stanley recently proposed that a nontrivial MC is a necessary condition for open-ended evolution because it prevents the population from falling into degeneracy [144, 145]. Successfully navigating a maze requires automatically learning an association between sensory inputs and appropriate course-of-action given those inputs. Given the indirect and often sparse error signal, it is reasonable to consider maze navigation a nontrivial capability. Each agent in the population is evaluated on potentially many different mazes from the population queue, and the agent must solve at least one of those mazes to meet the MC. Conversely, the maze population MC is to be successfully navigated by at least one agent, thereby preventing the persistence of mazes that are unnavigable by the extant population.

Recall that for evolution to begin, the individuals in the two population queues must themselves meet the MC. To ensure that this condition is met, the maze queue is seeded with mazes that have eight interior walls and the requisite number of agents are evolved, as part of a “bootstrap” phase, to solve *at least one* of these mazes. This ensures that mazes remain nontrivial, in that there are some obstructions that prevent a straight-line trajectory, while also not requiring significant agent complexification during the bootstrap stage. Agents are pre-evolved using novelty search due to its ability to quickly discover highly diverse solutions (specific novelty search parameters are detailed in section 4.1.4).

4.1.3 *Experimental Configurations*

The evolutionary process of MCC and the implications of its discoveries represent a significant departure from both objective-driven and, to a lesser extent, non-objective search algorithms published to date. While an objective-driven search algorithm may be evaluated based on its resulting fitness score or some other well-defined performance metric, casting an objective interpretation on a divergent or open-ended search method is fundamentally flawed because it bounds what is, in principle, an unbounded process of discovery. Non-objective and QD algorithms are similar to MCC in their rejection of an overarching objective function as a guiding force, but differ in terms of the artifacts that they produce. In particular, while QD algorithms may evolve multiple agent controllers that demonstrate a diverse array of navigation strategies for a *single* maze, MCC would coevolve similarly diverse controllers in tandem with *multiple* mazes, yielding a diverse array of nontrivial solutions in both populations within the scope of a single run. Such a disparity in both method and outcome render a direct quantitative comparison between MCC and other existing algorithms unfeasible and potentially misleading. Instead, the purpose of this initial evaluation is to survey the results of a search process that is almost entirely undirected and potentially open-ended, so as to better understand its evolutionary dynamics and identify opportunities for enhancement.

In the first experiment, agents and mazes are admitted to their respective queues based on satisfaction of their respective MC (described in section 4.1.2). On each iteration, a batch of mazes and agents are selected and evaluated asynchronously. Specifically, the mazes are evaluated against the existing contents of the agent queue and the agents are evaluated on the contents of the maze queue. When either queue reaches its capacity, excess individuals are removed based on their comparative age (i.e. the oldest are removed first) to make room for new offspring who have satisfied the MC. This configuration provides a simple baseline of MCC that acts as a control for the speciated variant and other possible extensions.

The second experiment imposes an aspect of local regulation by speciating both the agent queue and the maze queue. Given that agents are evolved using the NEAT algorithm, the agent queue is speciated based on NEAT’s built-in speciation mechanism [152]. Recall that mazes are represented as a collection of internal walls, with each wall having an assigned position and passage location; therefore, maze similarity should be computed as a function of wall and passage proximity. Accordingly, the maze queue is speciated based on a cantor pairing of each gene’s relative wall position and relative passage position. More concretely, both gene components (wall position and passage position) are combined to yield a scalar value for each gene m_i in maze genome m :

$$m_i = \frac{1}{2}(w_i + p_i)(w_i + p_i + 1) + p_i, \quad (4.1)$$

where w_i and p_i are the relative wall and passage positions encoded by gene i , respectively. Genetic similarity between two mazes is then determined based on computing the Euclidean distance between two maze position vectors. Maze vectors may vary in magnitude based on the number of interior walls they contain. If two maze vectors of differing length are being compared, the missing genes in the shorter vector are assigned position zero. Individuals in both the agent queue and the maze queue are logically grouped with genetically similar agents or mazes respectively (this is a

logical grouping because they still occupy their single, respective queue). Selection and removal are subsequently carried out on a per-species basis (detailed in section 3). Importantly, speciation in both populations is performed at the genotype level, thus avoiding the additional overhead of characterizing behavior or maintaining an archive [76, 78, 130] or map [46, 106], a property of MCC that sets it apart from all other divergent search algorithms proposed to date.

4.1.4 Experimental Parameters

The novelty search bootstrap process begins with 250 randomly generated agents and uses NEAT parameters identical to those given in [78]. Agents are evaluated on 10 mazes, each with 8 walls having randomly generated maze and passage positions. Bootstrap execution halts when 20 distinct agents are evolved that can solve one or mazes, and each of the 10 mazes is solved by at least one agent. The 20 agents and 10 mazes are used as the seed population of their respective queues. In the speciated variant, the seed genomes form the initial centroids of their respective species cluster.

Both the baseline and the speciated variant are executed for 20 runs of 2,000 batches each. The agent queue begins evolution with 20 seed genomes and has a maximum capacity of 250 genomes. NEAT is parameterized with a 0.7 probability of mutating a connection weight, 0.1 probability of adding a connection, 0.01 probability of adding a node and a 0.0001 probability of deleting a connection. The maze queue begins evolution with 10 seed genomes and has a maximum capacity of 50 genomes. The maze-specific evolutionary algorithm is parameterized with a 0.05 probability of mutating a wall location, a 0.05 probability of mutating a passage location and a 0.7 probability of adding a new wall gene. An initial parameter sweep revealed that the aforementioned values yielded reasonably diverse and complex solutions.

Agents have a terminal velocity of 3 units per time step and are allotted a maximum of 600 time steps to successfully navigate (i.e. from start to finish) a 320x320 unit maze. SharpNEAT 3.0 [62] is

used as the neuroevolution framework, and was extended to implement MCC and novelty search, and to support encoding, decoding and evolving maze genomes.

4.2 Results

One of the primary hypothesized features of MCC is its ability to evolve a broad array of diverse and increasingly complex artifacts in both populations within the same run. To validate that assertion, the results here focus on a qualitative assessment of population diversity as well as a quantitative analysis of complexification trends. Control (i.e. non-speciated) and speciated experimental variants are also compared to gauge the extent to which speciation aids in maintaining more diversity in both the agent and the maze population.

Figure 4.3 depicts a sample of mazes and agent trajectories evolved by a typical *single* run of MCC. Several of the mazes shown have reached maximum complexity and agents have developed similarly complex navigation strategies to solve them (as indicated by the solid line trajectory). This indicates a healthy process of coevolutionary complexification; however, many of the solution trajectories permitted by the mazes are similar. Separate runs of the control experiment also result in a high degree of structural similarity between mazes, but they typically converge to different such structures. This suggests that certain structural themes and navigation patterns may be becoming canalized. Figure 4.4 depicts a sample of mazes and agent trajectories evolved by a typical single run of the speciated variant. Maze structures and the resulting agent trajectories are visibly more diverse, suggesting that enforcing genetic diversity through maze and agent queue speciation may be a viable approach for maintaining adequate phenotypic diversity in MCC.

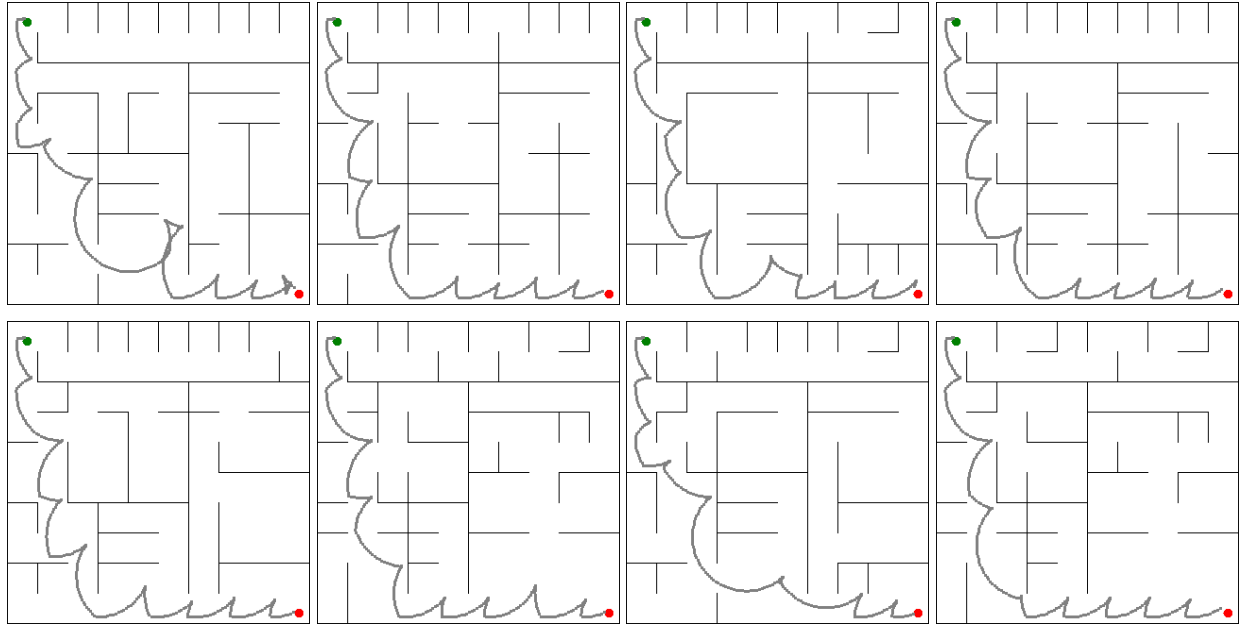


Figure 4.3: **Agent Trajectories from the MCC Control Experiment.** A sample of eight agent trajectories through different mazes evolved by the MCC control experiments during the same run are shown. These exemplify how solved mazes from a run of the control tend to look similar. While mazes are still getting more complex, their topological similarity suggests that diversity preservation may be a missing ingredient.

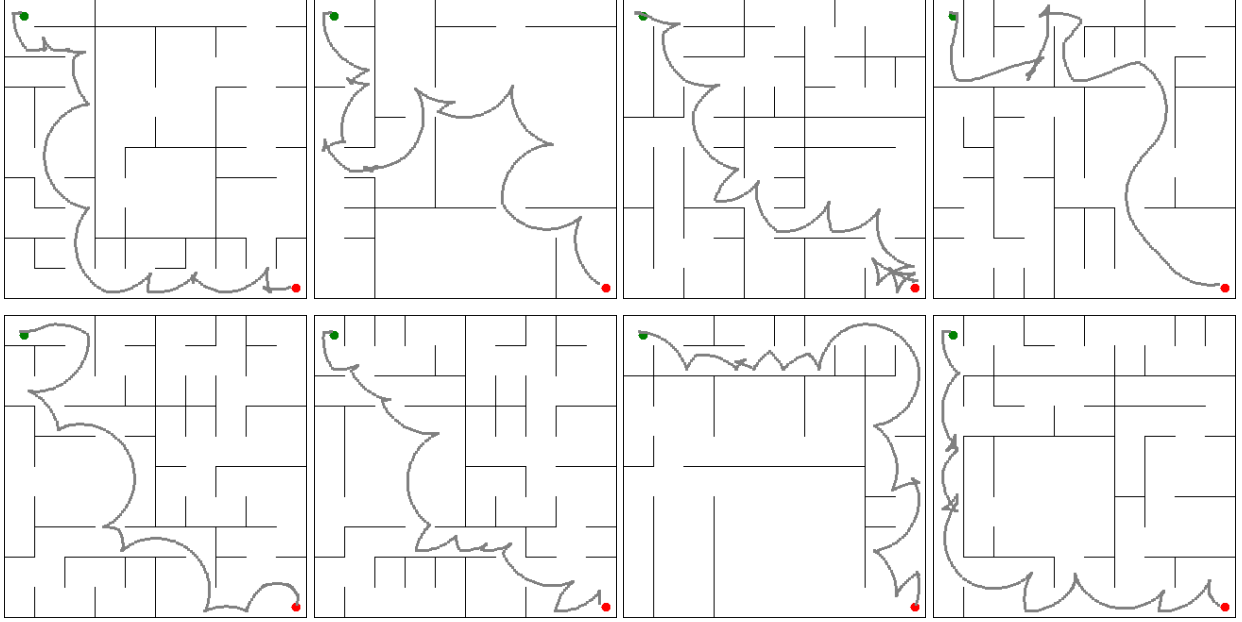


Figure 4.4: **Agent Trajectories from the MCC Speciated Variant.** A sample of eight agent trajectories through different solved mazes evolved by the MCC speciated variant during the same run are shown. Mazes maintain a healthy diversity, both in terms of complexity and arrangement of walls, while remaining nontrivial. This result is perhaps one of the first signs of open-ended divergence.

In addition to a sample-based assessment of the phenotypic and behavioral artifacts of MCC, a quantitative study is performed to provide a more holistic view of global diversity and complexity trends over the entirety of a run. Population diversity is computed by measuring the Euclidean distance between each point on a given agent’s trajectory and the corresponding point at the same time step on the trajectory of all other viable agents. Formally, trajectory diversity is given by:

$$\text{div}(a) = \frac{1}{n} \sum_{i=1}^n \left(\frac{1}{m} \sum_{t=1}^m \text{dist}(a_t, \mu_{i,t}) \right), \quad (4.2)$$

where a is the agent whose trajectory is being measured, a_t is the position of the agent at time

step t , $\mu_{i,t}$ corresponds to the position of the agent i against which a is compared at time step t , m is the number of time steps in the simulation, n corresponds to the number of agents who also successfully navigated a maze in the current maze population, and $dist$ is the function that computes the distance between any two agent’s trajectories (any distance metric could be plugged in, but in this case, the Euclidean distance is used). The global diversity score is computed by simply averaging the agent trajectory diversity scores. This approach has been used in prior work [75, 107, 135] and shown to be effective for measuring phenotypic and behavioral diversity.

A key feature of open-ended systems is the continuous accumulation of diverse and high-functioning artifacts throughout a single run. Figure 4.5 compares the diversity of agent trajectories (produced by maze-navigating agents throughout entire runs) from both the control and the speciated experiments. The results demonstrate that speciating the agent and maze queues lead to a significantly more diverse pool of maze solutions ($p < 0.001$; Welch’s t-test), thus affirming the merit of a diversity preservation mechanism. This dynamic is also evidenced in figure 4.6, which compares the proportion of mazes navigable by agents at various points during evolution. In particular, mazes in the control experiment are navigable by a significantly higher proportion of agents throughout evolution than those evolved by the speciated variant ($p < 0.001$; Welch’s t-test); a side-effect of speciation maintaining more diverse maze structures and navigation strategies. In addition to being more diverse, the mazes are also nontrivial to solve because there is no single navigation strategy that can navigate more than a small fraction of them.

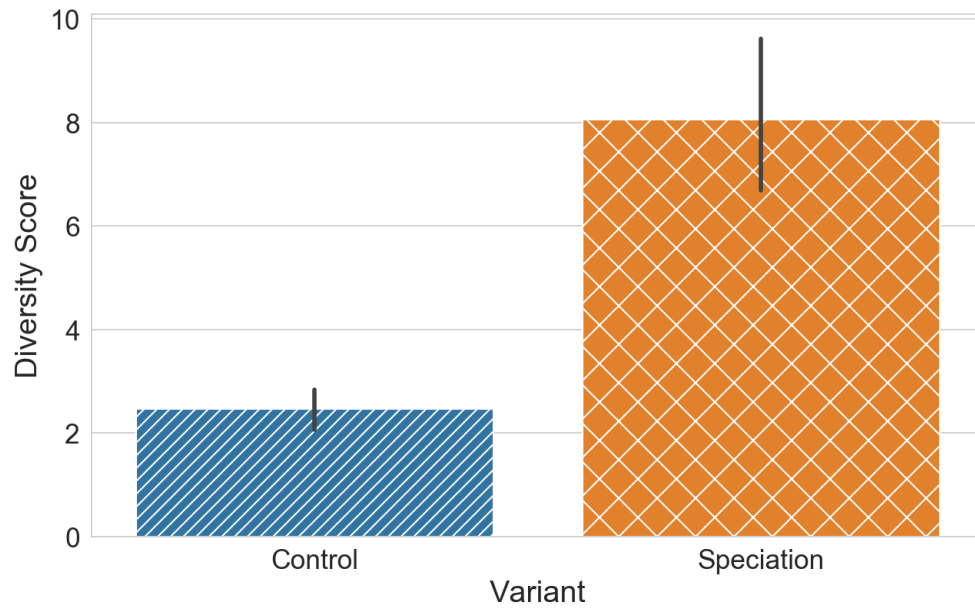


Figure 4.5: **Diversity of Agent Trajectories collected over Full Run (averaged over 20 runs of both variants).** The speciated variant configurations produce significantly more diverse solutions over the entirety of a given run, an indication that speciation is aiding in the preservation of diverse and nontrivial mazes, which in turn require specialized control strategies in the agent population.

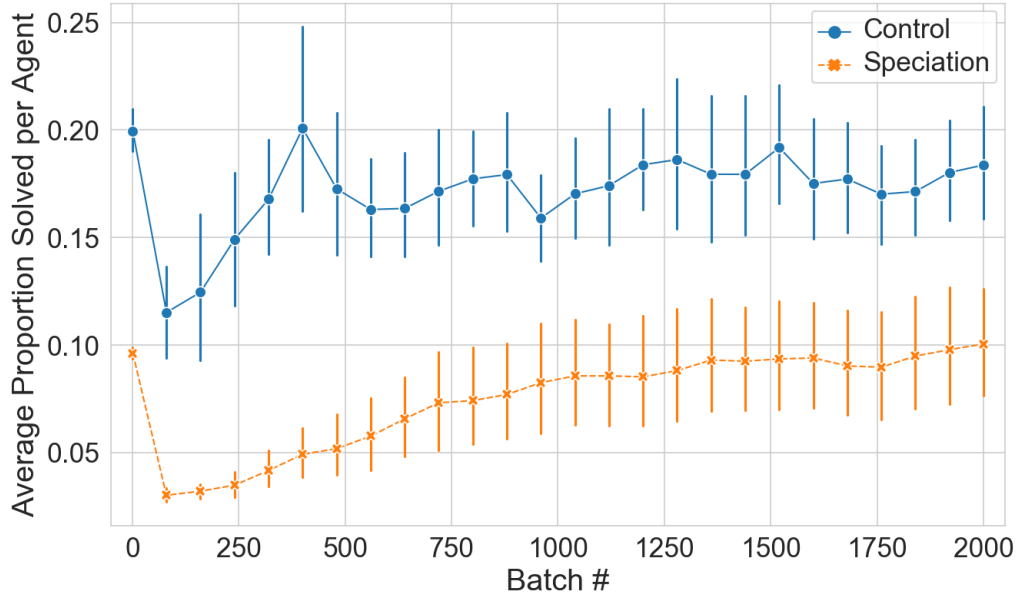


Figure 4.6: **Average Proportion of Mazes Solved per Agent over Evolution (over 20 runs of both variants).** The agents in the control population demonstrate the ability to solve a higher proportion of mazes as a result of convergence in the maze population to similar structures, as well as over-training of agents. Conversely, speciated agents solve a lower proportion because their population maintains greater heterogeneity.

While MCC demonstrated the ability to evolve artifacts that are both functional and diverse, perhaps its most notable capability is the evolution of increasingly complex artifacts. Figure 4.7 depicts rapid complexification in the maze population, leveling off only when the maze complexity ceiling is reached, and implying a similarly brisk developmental process in the agent population, which must be evolving progressively more advanced control policies to solve those mazes. This result hints at a genuinely open-ended process, and motivates the maze domain enhancements that follow.

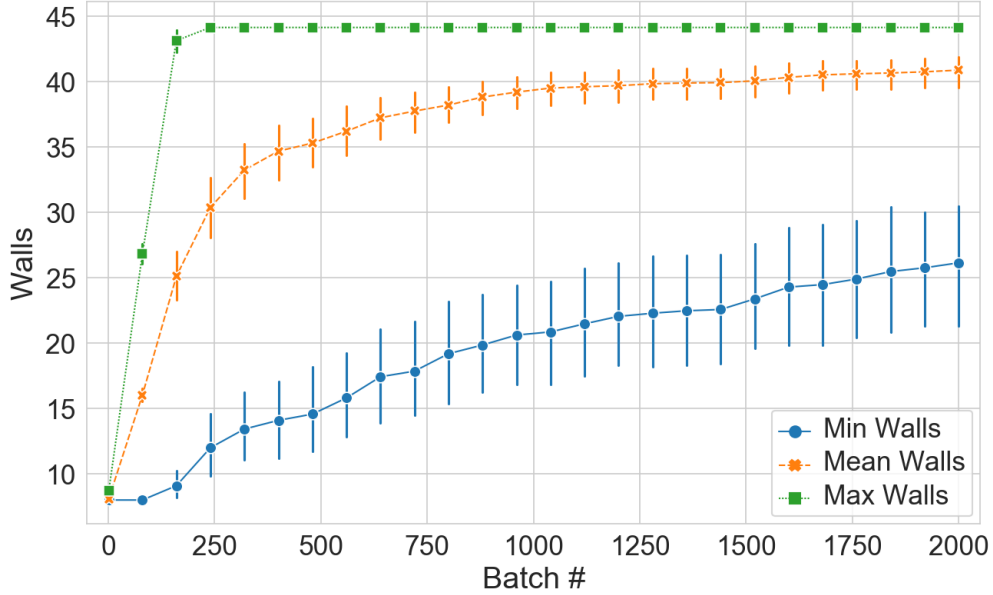


Figure 4.7: **Speciated Maze Complexity Trend.** The average minimum, mean, and maximum maze complexity over evolution is shown for the speciated variant. Note the rapid increase in complexity throughout the entire population while also maintaining a healthy *range* of complexities throughout the run. The leveling off of the maximum and mean maze complexities is a side effect of the fixed-sized maze domain, an artificial constraint that was imposed only for the initial investigation (and is removed in chapter 5).

4.3 Implications

The initial experiments demonstrate the ability of MCC to induce a reciprocating increase in complexity between both populations, ultimately reaching the maze domain complexity ceiling. This phenomenon persisted across all runs, even without explicitly promoting divergence through characterizing and archiving behavioral novelty. While both experimental variants produced a co-evolutionary divergence, speciation enabled MCC to explore divergent lineages simultaneously, discovering and elaborating on multiple unique pairings in a single run.

MCC demonstrated surprising efficiency at evolving agents capable of solving mazes at their max-

imum complexity (i.e. with the maximum number of walls supported by mazes of the given dimensions) well before the end of each run. The complexity ceiling was imposed to ensure computational tractability during initial experiments, wherein the dynamics of MCC were unknown. However, as an algorithm designed to facilitate open-ended discovery, even beyond the reach of conventional QD methods, MCC should be evaluated in a domain that is *unbounded* to the extent that complexity can continually be added in both populations. The next chapter demonstrates such a domain, where mazes can expand, add structure and diversify without limit, producing an unending series of navigation challenges.

CHAPTER 5: ENHANCED MAZE DOMAIN

The initial maze domain demonstrated MCC’s ability to coevolve principled navigation strategies in maze navigating agents along with increasingly difficult maze environments; however, a limitation in these experiments (which were bounded to maintain tractability for initial investigation) was that mazes were static in size and had a complexity ceiling based on the number of walls that fit within the maze boundaries. A notable result of the initial experiments was that MCC rapidly exhausted maze domain complexity, raising the compelling question of whether MCC would be able to continue to discover novel and more complex pairings if this artificial constraint were removed. The aim of the enhanced maze domain [8] is to address that question by allowing mazes to expand, creating more room for complicated solution paths and deceptive offshoots, and producing an unending curriculum of challenging tasks for agents. Additionally, the maze domain itself is offered as a lightweight benchmark for testing and comparing open-ended coevolutionary methods in the future.

5.1 Maze Encoding

As in the initial maze experiments, evolved mazes are square grids with a start location in the upper-left corner and target location in the lower-right, both of which remain fixed throughout evolution. Maze-navigating agents are simulated, wheeled robots that are controlled by evolved NNs; their architecture is identical to the initial experiments described in chapter 4, with six rangefinder sensors (positioned at heading offsets of -90° , -45° , 0° , 45° , 90° and 180°) that measure distance to line-of-sight obstructions, and four pie-slice radar sensors that are positioned along the agent’s circumference and activate when the target is within the sensors’ arc.

Agents begin a trial at the start location and are evaluated on their ability to reach the target location within a given simulation time; however, because mazes will be allowed to expand, the simulation time is scaled linearly to the length and complexity of the solution path. The mazes consist of internal walls that impede trivial, straight-line trajectories, well as winding and deceptive traps that are introduced by explicitly evolving and convoluting the solution path (detailed in 5.1.2), presenting opportunities for an agent to wander into a dead-end corridor and expend the allotted time. Both of these features are intended to encourage the development of complex and principled behaviors.

5.1.1 *Maze Expansion*

The MCC bootstrap process remains unchanged from the initial experiments: seed agent genomes are evolved to solve a set of randomly-generated mazes with fixed boundaries and uniform size. Following completion of the bootstrap, the MCC maze reproduction process includes an *expand maze* mutation probability controls the proportion of mutations that result in a maze expansion. When an expand maze mutation is applied, the outer boundaries of the selected maze are each lengthened by one unit, maintaining a square geometry. Maze boundaries expand down and to the right, thereby moving the target location further away from the starting location. Interior walls are each lengthened proportionally to fill the additional space.

Once MCC has run for several iterations, the maze population includes mazes that not only vary in structure, but also in size, with larger mazes accommodating the addition of more interior walls and allowing the formation of more complex and convoluted solution paths. Figure 5.1 (right column) shows a larger maze generated by the new maze encoding. The hope is that the continual process of maze expansion and elaboration will elicit incremental development of increasingly advanced navigation strategies in the agent population, producing a reciprocating complexity explosion in

both populations and promoting continual discovery of unimagined problems, each paired with novel and effective solutions.

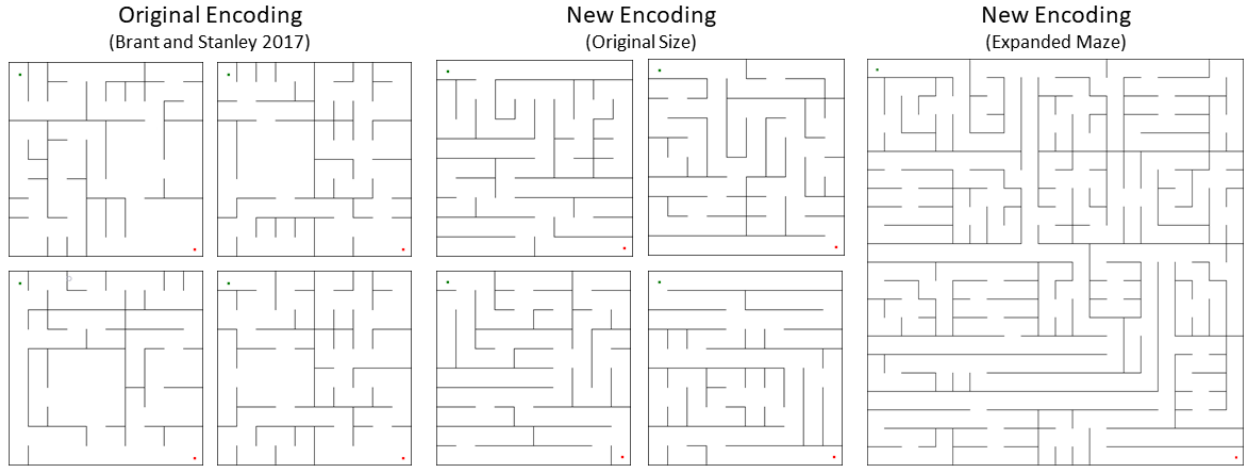


Figure 5.1: **Comparing the original (published in [7] and described in chapter 4) and new maze encodings.** Mazes generated in the initial MCC experiments (left) are shown next to mazes of a similar size generated by the new encoding (middle), and a larger (expanded) maze, also produced by the enhanced encoding (right). While the initial maze encoding prevents trivial strategies by obstructing straight line navigation, the solution path lacks a distinct shape, and wall placement is often unbalanced and overly segmented, filling up some corridors with short stubs while others are left largely empty. In contrast, the new maze encoding produces winding and non-trivial solution paths with balanced wall placement, and maintains these properties as mazes expand.

5.1.2 Path Evolution

In the initial MCC experiments, walls were placed so as to permit only one route from start to finish. Effectively, the solution path was an artifact of wall placement, which exhibited two distinct shortcomings: (1) small changes in wall placement (resulting from a mutation operation that shifted or added a new wall) could substantially modify the path, potentially inducing a highly rugged search landscape, and (2) as mazes expand and more walls are added, the path is further subdivided into short directional segments with similarly shallow corridors, yielding a degenerate

and unconvincing aesthetic to the mazes (figure 5.1, left column). While adding walls may make the maze as a whole may appear imposing, more detailed investigation occasionally reveals that a trivial navigation strategy may be able to succeed through repeated application of a simple policy (e.g. move up, move down, repeat).

In the new encoding is, rather than relying upon wall placement to mold the path, the *path itself is evolved*, specified by a series of “waypoints,” that are joined by a simple set of rules that derive a single valid solution trajectory. The solution path carves out sub-sections of the maze that open onto the path and are complexified by distributing and arranging wall genes, creating multiple opportunities for an agent to get stuck by veering into long and winding corridors, producing an aesthetic that is similar in appearance to conventional mazes (figure 5.1, middle and right columns).

Waypoint coordinates, called *path genes*, are directly encoded in the maze genome and augmented with an “intersection orientation,” which specifies the direction of the path segment (i.e. vertical or horizontal) as it intersects the waypoint. A *mutate waypoint* probability shifts an existing waypoint by one unit in one of four directions (up, down, left or right) while an *add waypoint* probability controls the frequency at which new waypoints are added to the maze. The intersection orientation informs how the solution path is structured when connecting two waypoints; it is randomly-assigned and fixed (i.e. not mutated). Waypoints are connected in the order that they are added to the maze genome, and the direction of the solution path respects waypoint intersection orientations.

Figure 5.2 visually depicts the effect of the path, wall and maze expansion mutation operators. The leftmost maze consists of one waypoint and with a horizontal intersection orientation at coordinate (6, 4). The solution path is laid down vertically from the start location, then turns at (1, 4) to horizontally intersect the waypoint, after which it descends to the bottom of the maze and reorients again to form a horizontal intersection with the target. Note that the only waypoint in the first

column is the one at coordinate $(6, 4)$; the other path segments, including locations where the path changes direction (known as “junctions”), are generated by the path generation procedure. The bottom row depicts the final maze product, with walls positioned around the path. In the first column, there is only one wall gene with a vertical orientation, and it is repeated so that all sub-spaces are filled to capacity.

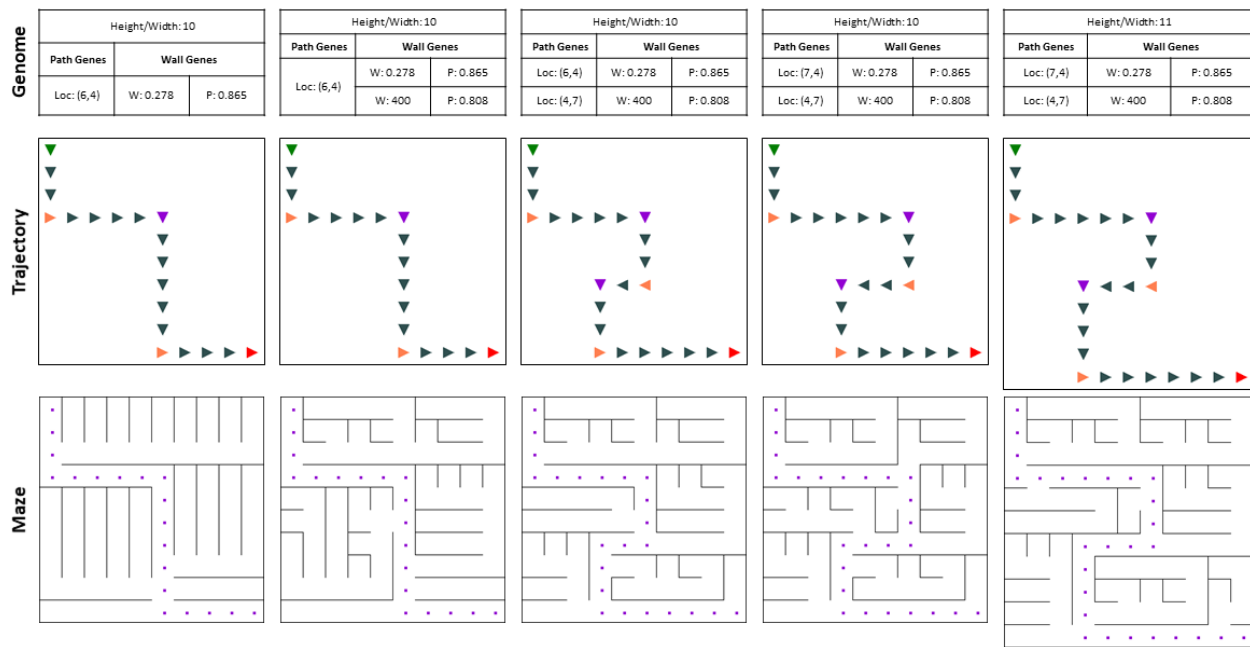


Figure 5.2: **The maze evolution process.** Maze genotypes (top), respective trajectories (middle), and respective phenotypes (bottom) are shown. The table above each maze enumerates the path (waypoint) and wall genes, while the visualization directly below depicts the solution path that is generated by connecting each waypoint. In the maze phenotypes at bottom, dots are shown to depict the solution path. Each path gene specifies the coordinates of a waypoint, and each wall gene contains a wall position and passage position, both real numbers in the interval $[0, 1]$. The wall position (W) describes the relative position of a wall within a sub-space induced by the solution path, while the passage position (P) describes the relative position of a passage (opening) within that wall. The initial genome (first column) consists of one path gene and one wall gene; the single wall gene is repeated to fill out all sub-spaces surrounding the solution path. The second column depicts an *add wall* mutation, where the two walls repeat to fill out each subspace, adding variation to the resulting corridors. In the third column, an *add waypoint* mutation and the resulting solution path modification is depicted. A *mutate waypoint* operation is carried out in the fourth column, and the solution path is modified to intersect the new waypoint location without substantially changing the shape of the solution path. Finally, the fifth column depicts an *expand maze* mutation, extending the outer boundaries down and to the right, and resulting in a corresponding extension of the solution path that modifies only the last vertical segment while leaving the rest intact.

The second column demonstrates an *add wall* mutation. The second wall has a horizontal orientation and it is alternated with the first to fill up all sub-spaces, leading to mazes that exhibit walls

with varying lengths and orientation. In the third column, a second waypoint (path gene) is added at $(4, 7)$ with a horizontal intersection orientation. The path generation procedure accommodates the new waypoint and its horizontal intersection orientation by inserting a juncture at $(6, 7)$ such that the path orientation is changed from vertical to horizontal. There are many different ways in which the two waypoints could have been connected, but a governing principle of the path generation procedure is to apply the modification that results in the smallest possible path modification. Additionally, waypoint additions are constrained such that they are added either below (as in the present example) *or* to the right of all existing waypoints. Similarly, mutation cannot shift a waypoint both above and to the left of prior waypoints. These constraints prevent trajectory overlap in mazes that have dense concentrations of waypoints, while still permitting solution paths that force movement in all cardinal directions.

A *mutate waypoint* operation is shown in the fourth column, shifting the original waypoint from $(6, 4)$ to $(7, 4)$ (recall that the intersection orientation remains fixed). Once again, the path generation procedure applies the smallest change possible, simply extending the first horizontal segment of the solution path to intersect the waypoint at its new location. Finally, the fifth column demonstrates an *expand maze* mutation, where each of the outer boundaries are extended by one unit down and to the right. To intersect the target (which is now at $(11, 11)$ instead of $(10, 10)$), the last vertical path segment is extended by one unit while the rest of the path is unmodified. The maze generation algorithms are described with detailed pseudocode in appendix A.

5.2 Experiment

The bootstrap process is executed in a manner identical to the initial maze domain experiments, using the novelty search algorithm with parameters identical to those in Lehman and Stanley [78]. Agents are evaluated on ten simple, randomly-generated mazes, each of size 10×10 and consisting

of two waypoints and two wall genes. Execution halts when 20 distinct agents are evolved that can solve one or more mazes. Those 20 agents and 10 mazes seed their respective population queues and constitute the initial centroids of each species cluster. During MCC execution, each agent must solve at least one maze to satisfy their MC while each maze must be solved by at least one agent. This relaxed constraint permits rapid complexity growth while ensuring that mazes remain solvable by at least a portion of the extant population. Agents are evaluated on random mazes from the current population and vice versa, and evaluation halts when an individual satisfies their MC or when they have been evaluated on all members of the opposite population.

Each experiment is executed for 20 runs of 2,000 batches, each batch evaluating 40 agent genomes and 10 maze genomes. The agent queue has a maximum capacity of 250 with a 0.6 probability of mutating connection weights, 0.1 probability of adding a connection, 0.01 probability of adding a node and a 0.005 probability of deleting a connection. The maze queue has a maximum capacity of 50 with a 0.05 probability of mutating a wall, passage or waypoint location, 0.1 probability of adding a wall, 0.005 probability of deleting a wall, 0.1 probability of adding a waypoint and a 0.1 probability of maze expansion. These values produced complex mazes of variable size and complexity during a parameter sweep.

Agents are restricted to a maximum velocity of 3 units per second and are allotted a simulation time equivalent to two times the length of the solution path (where length is the sum of single-unit path segments, as shown in figure 5.2). This limit grants a simulation time proportional to the initial MCC experiments with fixed-size mazes, while scaling with the size of the maze and the complexity of the solution path. That is, if two mazes are of the same size but one has a longer, more winding solution path than the other, then it will be granted a longer simulation time.

SharpNEAT version 3.0 [62] is the neuroevolution platform, and was extended to implement the MCC and novelty search algorithms, as well as to support encoding and evolving maze genomes.

Note that a direct quantitative comparison of the present results with those presented in chapter 4 would not be feasible because these experiments enable unbounded domain expansion – a characteristic that the original experiments lacked.

5.3 Results

Recall that a primary goal of MCC is to induce an open-ended search process through coevolutionary interactions that result in increasingly complex and diverse discoveries. The results shown below demonstrate MCC’s ability to exploit the enhanced maze domain, evolving agents that are capable of solving larger, more complex mazes. The qualitative results showcase a diverse array of mazes that were evolved in a single run, each with varying size and complexity, and solved by an agent controller. Additionally, a quantitative analysis demonstrates a systematic increase in both maze and navigator complexity.

Maze expansion creates space for additional waypoints, which can often further complicate the solution path by morphing it in non-trivial ways and introducing additional opportunities for deception through adjoining cul-de-sacs. Figure 5.3 depicts a sample of agent trajectories (from evolved NN controllers) through mazes evolved by a *single* run of MCC, each varying in both size and structure. These results show visually how mazes are expanding in size and complexity while continuing to be solved.

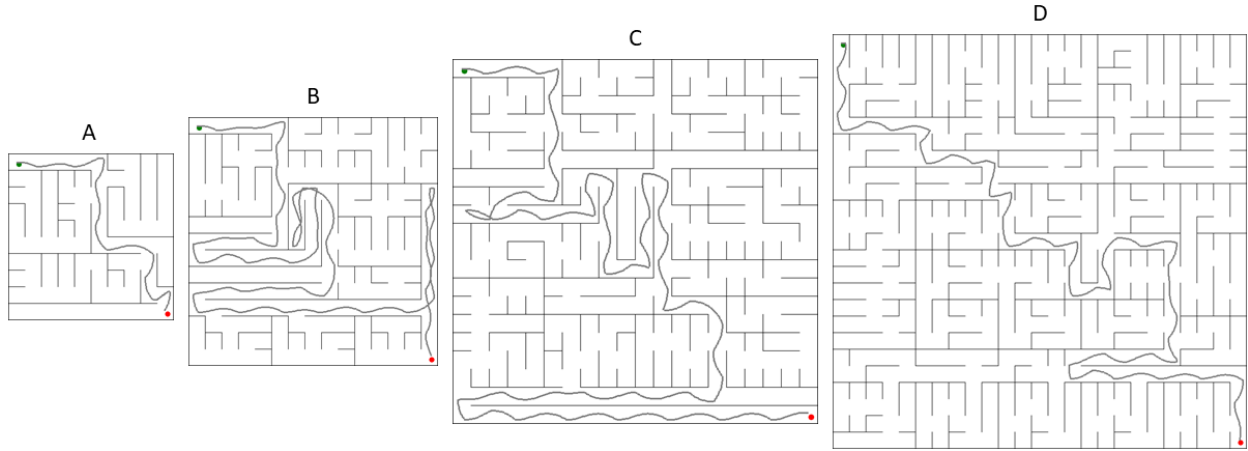


Figure 5.3: **Agent trajectories discovered within a single run of MCC.** A sample of four solution trajectories through different mazes evolved within a single run of MCC are shown. Maze A is similar to the bootstrap mazes in size and complexity, with outer walls that are 10 units in length and containing 2 waypoints and 2 interior walls. Maze B is 15x15 with 4 waypoints and 3 walls, while maze C is 22x22 with 7 waypoints and 12 walls. The largest maze (D) is 25x25 with 8 waypoints and 6 walls.

A key feature of the reconceived maze domain is the ability to dynamically expand, allowing additional space for longer, more convoluted trajectories, which can force agents to evolve increasingly complex navigation strategies to remain viable (i.e. to satisfy their MC). Figure 5.4 demonstrates that mazes indeed increase in size, linearly on average, without leveling off, while the number of junctures (where turns are necessary in the main solution path) continues to increase at a similar rate (figure 5.5). As both results also show, MCC, on average, maintains a healthy diversity of maze sizes and configurations throughout each run. Regarding the agent NNs, figure 5.6 depicts a linear increase in maximum and mean number of connections throughout evolution, demonstrating, in part, the agents' evolutionary response to increasingly difficult mazes. However, the minimum complexity remains flat as less complex agents maintain viability by exploiting smaller, less complex mazes. Moreover, the widening gap between minimum and maximum sizes suggests a continued search space divergence that shows no sign of leveling off.

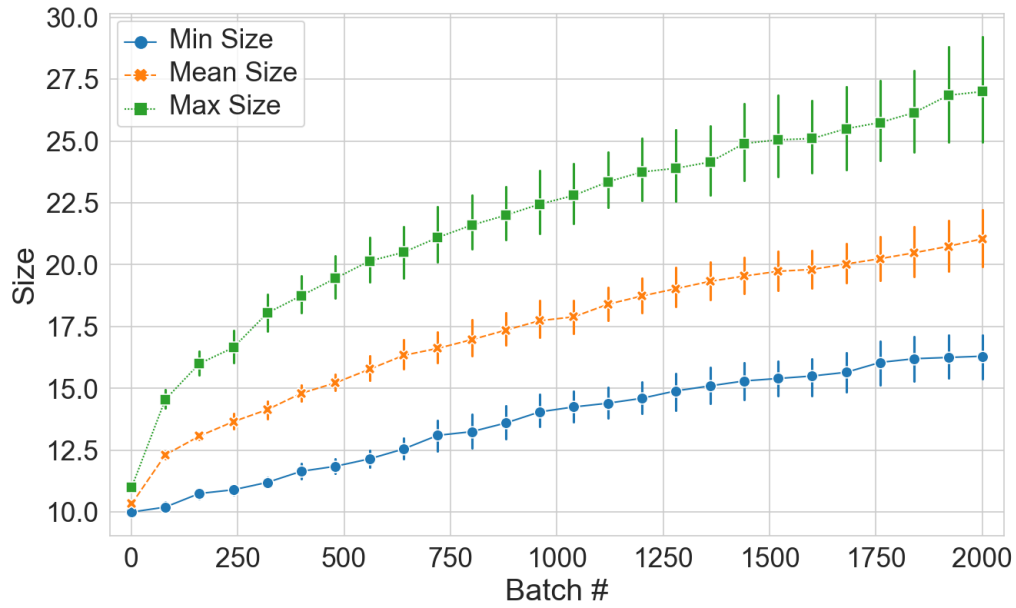


Figure 5.4: **Maze Expansion Trend.** The average minimum, mean and maximum maze dimensions over evolution are depicted. Mazes expand linearly throughout the course of a run with no evidence of tapering off, suggesting MCC’s ability to induce a process of unbounded maze elaboration.

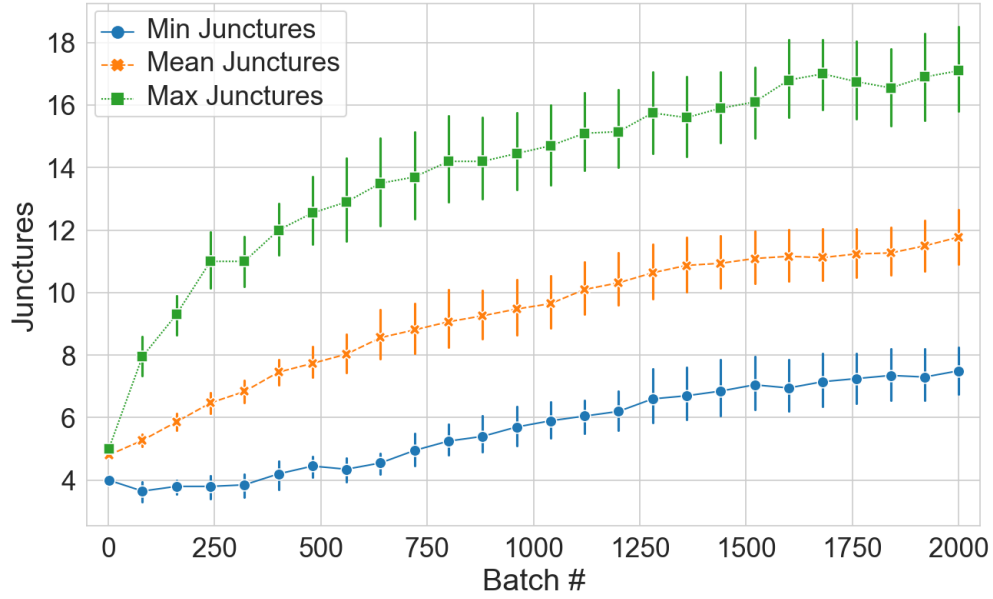


Figure 5.5: **Solution Path Complexification Trend.** The average minimum, mean and maximum junctions over evolution are shown. As waypoints are added, the solution path is convoluted by junctions—perpendicular intersections from waypoint connections which force agents to develop a policy capable of principled-reorientation along the solution path.

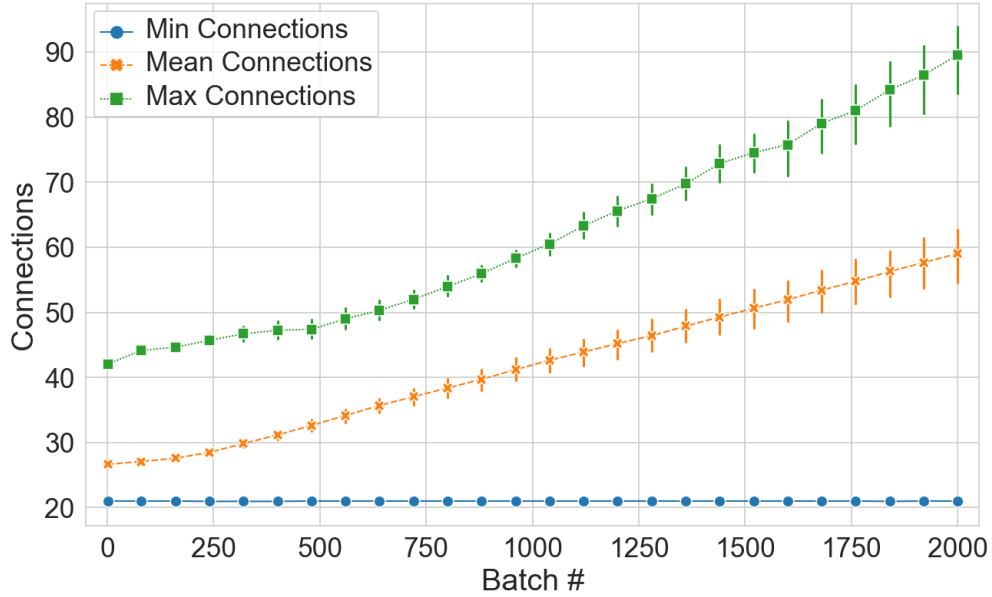


Figure 5.6: **Agent NN Size Trend.** The average minimum, mean and maximum agent NN connections over evolution are shown. Maximum and mean NN size increases as agents evolve to solve more difficult mazes, while some agents remain small (indicated by the minimum connections trend), but refine their existing weights and satisfy their MC using smaller, less complex mazes. As with the diversity of maze sizes, this result demonstrates MCC’s ability to maintain diversity in both populations, keeping many distinct paths to reproductive viability open simultaneously.

Maze expansion and path complexification slice the maze space into quadrants that open on to the solution path. Some of these openings may initially appear to provide a more direct route to the target, introducing a deceptive characteristic that can lead agents astray and prevent timely solution path traversal. The addition of these “deceptive entrances” is proportional to the rate of maze growth and path complexification (figure 5.7), adding further evidence of increasingly robust agent adaptation.

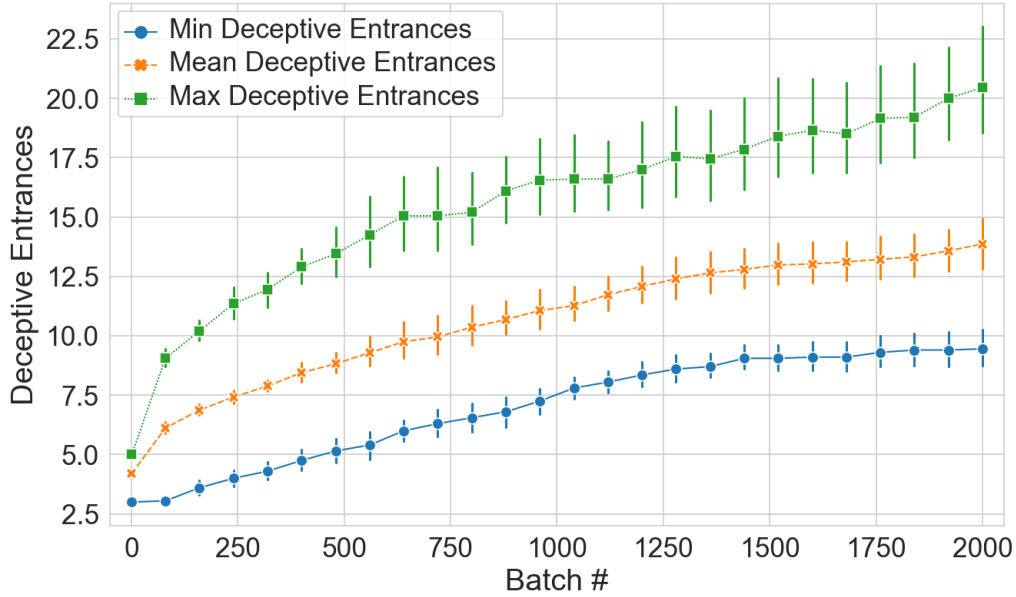


Figure 5.7: **Solution Path Deception Trend.** The figure depicts the average minimum, mean and maximum quadrant entrances throughout evolution. Entrances that face the solution path introduce opportunities for deception by allowing an agent to wander into regions that may initially seem promising, but that are ultimately dead-ends. The continued addition of path openings indicate MCC’s ability to evolve navigation strategies that are robust to variable levels of domain deception.

A notable, yet intuitive distinction between the initial maze domain used in the original MCC experiments and the enhanced, expanding maze domain is the trend in population viability over the course of evolution. In particular, after batch 511, the percentage of maze offspring that are viable in a run of fixed mazes (averaged over 20 runs) is significantly higher than for expanding mazes ($p < 0.05$; Welch’s t-test) because the fixed mazes have reached their complexity limit and therefore stop consistently posing new challenges to the agent population (figure 5.8). On average, only between 10% and 15% of the agent offspring satisfy their MC on expanding mazes at any point in evolution. This observation is further reinforced by examining the proportion of mazes that a given agent is able to solve at any point during evolution, shown in figure 5.9. In the initial maze domain, agents are able to apply similar strategies to solve a significantly higher proportion

of fixed size mazes than expanding mazes ($p < 0.05$ after batch 650; Welch’s t-test) because while maze walls are rearranged, the solution path remains comparatively trivial. Additionally, because mazes are of a fixed size, there exists a finite number of ways in which the solution path can be perturbed. In the enhanced maze domain, however, evolution extends and warps the solution path of each maze in different and complex ways to which a single navigation strategy would unlikely be able to generalize. By allowing mazes to expand unboundedly and modify their solution path, MCC is able to consistently generate new challenges for the agent population.

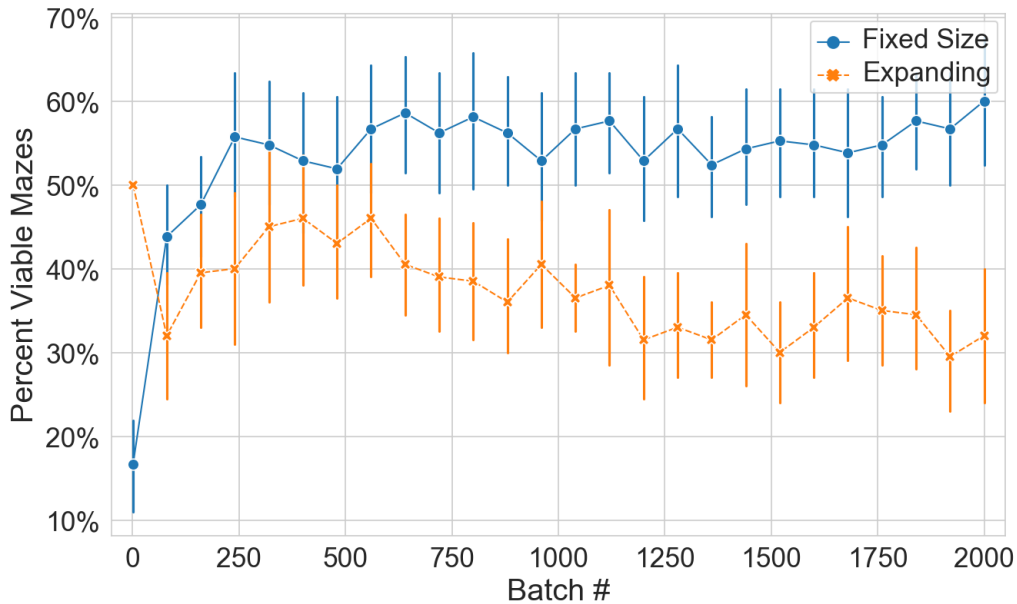


Figure 5.8: Maze Population Viability Trend. The figure depicts the comparative percentage of viable mazes over evolution for the fixed and expanding maze domains. In the fixed-size maze experiments, mazes quickly reach their complexity ceiling, and further mutations yield minor wall rearrangements that prove trivial for navigation, as an increasing proportion of the agent population drifts toward a small set of strategies. When this complexity constraint is removed, domain difficulty varies organically, leading to more difficult mazes that prove challenging (though not impossible) to navigate.

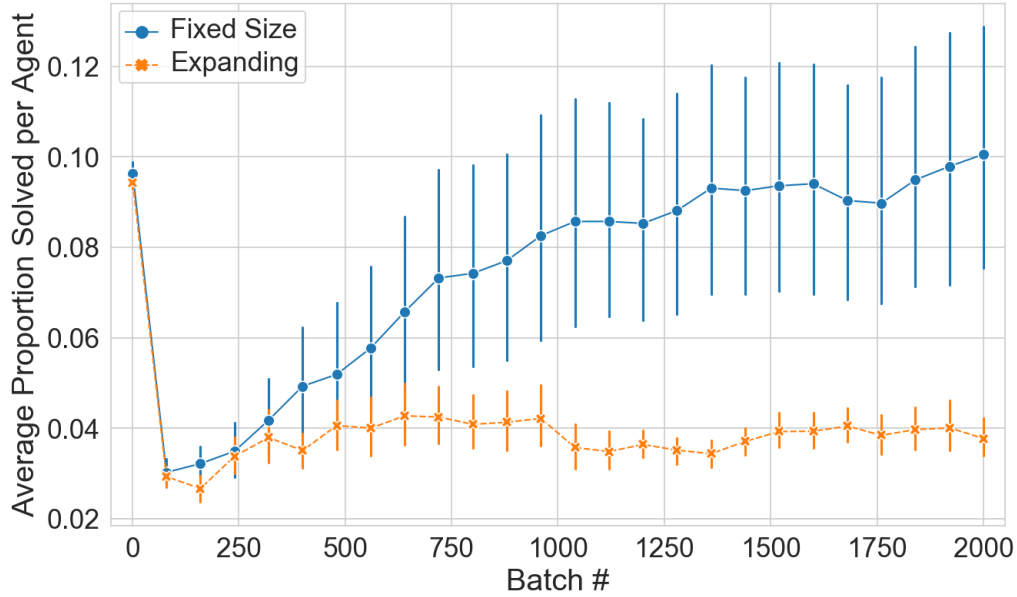


Figure 5.9: **Average Proportion of Mazes Solved per Agent over Evolution (over 20 runs of both variants).** The agents in the initial maze domain experiments demonstrate the ability to solve a higher proportion of mazes because the fixed maze canvas size imposes a limit on the number of solution path perturbations that can emerge from wall additions and rearrangements, which allows agents to eventually hone in on a finite subset of successful navigation strategies. The enhanced maze domain imposes no such limit; because mazes can expand, an effectively infinite number solution path shapes can be discovered, thereby preventing agents from relying on a small set of well-adapted policies.

5.4 Implications

The expandable maze encoding makes it possible to observe the MCC algorithm’s ability to exploit a novel maze domain, relieved of complexity constraints and designed to grow unboundedly subject to the MC. Reciprocating increases in size are evident in both the maze and agent populations (as measured by the number of NN connections), with no apparent stagnation. In all runs, agents evolved solutions to a breadth of non-trivial mazes, inducing a population-wide drift toward increasingly large and intricate mazes, each with complicated solution paths that exhibited multiple

opportunities for deception. Both populations achieved an MC-constrained equilibrium with each other, maintaining a sustainable rate of complexification and producing a diverse array of mazes and associated solutions, even without relying on the guidance of fitness or the divergent pressure of novelty.

Although these results are among the first to explore open-ended coevolutionary growth and complexification, there remain several opportunities for further investigation. A close examination of learned navigation strategies hints that, in some cases, agents may be relying on simpler heuristics than their trajectories would suggest. For example, in figure 5.3 maze A, the agent is forced to make five decisions regarding the direction to turn. However, for all but one of those decisions, the agent only has one option that does not lead into a wall or backtrack on the solution path. This fact is a potential hint of some degree of collusion in the maze and agent populations: mazes discover the “path of least resistance” for adding complexity that is easily exploitable by agents. However, the extent to which these system dynamics can be observed and addressed is vastly improved in the enhanced maze encoding.

The next chapter presents a method that reduces such collusion, exploiting the enhanced maze encoding to evaluate a new method of diversity preservation that uses resource limitation (instead of speciation) to induce divergence by limiting the number of agents that can use the same maze to satisfy their MC.

CHAPTER 6: RESOURCE LIMITATION

The expanding maze domain produced an array of novel results that demonstrate MCC's capacity for open-ended innovation and incremental complexification in multiple simultaneously evolving lineages. However, while MCC's evaluation process (i.e. evolution subject to an MC) is rooted in nature's loose reproductive viability criterion, its method of diversity preservation remains ad-hoc and lacks natural precedent. Like many other EAs (including QD algorithms) MCC employs speciation to segregate members of both populations into one of a predetermined number of species based on an explicit comparison of genetic similarity (chapter 2 section 2.2 reviews speciation in EC). Though speciation is a well-established, ecological phenomena [114, 134], its implementation is contrived and determining how to meaningfully compare similarity imposes a priori structure on a system that is intended to produce novel and perhaps unanticipated artifacts. This chapter introduces an alternative method of diversity preservation through resource limitation, and demonstrates its application to the maze domain where a far greater diversity of maze sizes and solution path motifs are discovered without the need to explicitly measure and compare genetic similarity.

6.1 Diversity Preservation through Limited Resources

In nature, species occupy niches whose carrying capacity is limited by environmental factors, including the availability of natural resources (i.e. consumable material that is required for the persistence of an organism, such as habitat and nutrients) and the frequency predation [134, 167], both of which impose a form of local regulation on niche membership. A long-standing hypothesis in population ecology is that competition for limited resources has played a critical role in adaptive radiation and the formation of ecological communities [137, 163, 164]. In particular, as resources become increasingly scarce, organisms are forced to diverge and found new niches in which to

continue their lineage. Those organisms capable of exploiting underutilized resources will be favored over those who cling to resource-depleted niches [138].

Using the Avida digital evolution platform [113], Cooper and Ofria [20] demonstrated that competition for limited resources can have a stabilizing affect on simulated community structures. Resource limitation was also shown to be an effective form of diversity preservation in EC [35, 48, 49], but it has primarily been leveraged in the context of objective-driven, convergent search processes. Recent work in EC, however, has shown that genetic drift (a process on which MCC heavily relies) can lead to the founding of new niches [29, 82], thereby creating novel and varied opportunities for individuals to persist. This effect is especially true for organisms who exhibit a high degree of *evolvability* because they are innately more adaptable to new niches, thereby promulgating new lineages of similarly evolvable offspring.

Viewing speciation as a process that emerges from limited environmental resources and random drift across the genotype space is well-aligned with MCC’s open-ended approach to search. Rather than computing a genetic distance between individuals and grouping the most similar into one of a predefined number of species, in this new version of MCC a limitation is placed on the number of times an individual in one population can “use” an individual from the other population to satisfy their MC. This approach is particularly intuitive when individuals in one population are framed as evaluation environments and those in the other population as organisms whose MC is to interact with an environment in some desirable manner.

This way, each member of the population on which a resource limit is imposed maintains a tally of the number of times it has been utilized by a member of the opposite population for satisfying their MC. When the usage tally reaches the predefined resource limit, it can no longer be used for MC satisfaction. Algorithm 3 formalizes the MCC selection, evaluation and removal process with the resource-limited population. Maintaining diversity through resource limitation negates

the requirement for computing genetic distance, creating a dynamic, self-regulating system that exploits MCC’s propensity for open-ended divergence to discover new and varied adaptations.

6.2 Experiment

The enhanced maze domain (which is described in chapter 5) is used to evaluate resource limitation as a potential method of diversity preservation. Mazes consist of a solution path that is specified by a series of evolved waypoints (path genes), along with walls that are placed in spaces carved out by the path into which an agent can traverse and become lost in winding corridors. An agent begins a trial at the top left of the maze and must navigate the solution path to reach the target (placed at the bottom right) within the allotted simulation time.

While a differing arrangement of walls may increase or decrease the level of domain deception, it may or may not necessitate modification of an agent policy for successful navigation. Conversely, perturbing a path gene directly modifies the solution path, which is more likely to require explicit adaptation. Path genes are the fundamental units of maze genomes, and were therefore used to assess maze similarity for speciation in the original MCC experiments. Specifically, maze similarity was computed as the manhattan distance between every pair of path waypoints. If mazes had an unequal number of path genes, the last waypoint in the maze with fewer path genes was used for remaining distance calculations. In the agent population, the standard NEAT method of speciation was used [152] wherein similarity is determined by the number of connections with matching historical markings.

While deriving an intuitive distance metric is fairly straightforward for simple structures such as mazes, there may exist other domains for which quantifying distance is more difficult or less informative. For example, in a robot locomotion domain where MCC is used to coevolve morphology

Algorithm 3 MCC Evaluation Process with Resource Limitation

Require:

batchSize - # of individuals to evaluate simultaneously
numSeeds - # of seed genomes to evolve that satisfy the MC
resourceLimit - max # of evaluations that count toward MC

▷ Evolve seed genomes that satisfy MC

randPop \leftarrow *GenerateRandomPopulation*()

viablePop \leftarrow *EvolveSeedGenomes*(*randPop*, *numSeeds*)

▷ This loop runs for both populations

loop

▷ Reproduce children and add parents back into queue

parents \leftarrow *viablePop.Dequeue*(*batchSize*)

children \leftarrow *Reproduce*(*parents*)

viablePop.Enqueue(*parents*)

for all *child* in *children* **do**

if *popType*(*child*) = *agent* **then**

for all *env* \in *environments* **do**

 ▷ Only select environments below resource limit

if *env.Usage* < *resourceLimit* **then**

evalIndiv \leftarrow *env*

break

end if

end for

else

 ▷ Agent population does not have resource limit

evalIndiv \leftarrow *next*(*agents*)

end if

mcSatisfied \leftarrow *EvaluateMC*(*child*, *evalIndiv*)

if *mcSatisfied* **then**

viablePop.Enqueue(*child*)

end if

end for

▷ Removed oldest if queue capacity exceeded

if *viablePop.Size* > *viablePop.Capacity* **then**

numRemovals \leftarrow *viablePop.Size* - *viablePop.Capacity*

RemoveOldest(*viablePop*, *numRemovals*)

end if

end loop

and control, the dimensions by which to characterize morphology distance are perhaps less intuitive, and could also vary substantially based on the physical substrate (e.g. rigid joints and materials vs. soft body structures with varying degrees of material compliance). More importantly, nature requires no such ad-hoc means of explicitly measuring organism distance to encourage diversity. Instead, evolutionary divergence is in part a byproduct of niche carrying capacities, or *resource limits*.

To emulate such limitation, each maze is assigned a fixed resource limit, and only successful navigations within that limit are counted toward satisfying an agent MC. For example, if a maze has a resource limit of five, and five agents have already solved the maze, then it has reached its resource limit and is no longer available for agent evaluation (because successful navigation would not count toward satisfying an agent's MC, and would thus be an inefficient use of computation). Note also that resource limitation is applied *asymmetrically* in that it is only imposed on the maze population; agents do not have an associated resource limit, so there is no restriction on the number of times they can be used to satisfy a maze MC. In practice, implementing resource limitation in only one direction is sufficient to promote diversity in both populations, because as individuals in one population maintain viability by using alternate, under-utilized individuals in the other, they naturally evolve a wider range of adaptations that enable continued divergence in the opposite direction as both populations interlock to satisfy a mutual MC. For example, in the maze domain, agents must evolve diverse navigation strategies to solve mazes that have not reached their resource limit, which, conversely, enables them to successfully navigate similarly diverse mazes, thus satisfying the MC of those mazes.

To facilitate direct comparison and isolate the effects of the diversity preservation method, the bootstrap parameters and MCC parameters are setup exactly as in chapter 5. The bootstrap process is executed using the novelty search algorithm wherein agents are evaluated on ten simple, randomly-generated mazes of identical size, each with two path genes and two wall genes. Twenty

agents are evolved such that each agent is able to solve at least one maze, while respecting the maze resource constraints (i.e. no more than five agents can solve a single maze). Mazes retain the resource usage incurred during initialization, and seed the maze population queue while the evolved agents seed the agent queue.

MCC experiments are executed in 20 runs of 2,000 batches, where 40 agents and 10 mazes are evaluated within each batch. Agents have a maximum velocity of 3 units per second and are allotted a simulation time of two times the maze solution path length. SharpNEAT version 3.0 [62] is the neuroevolution platform, which was extended to implement novelty search and MCC with speciation and resource limitation, and to support encoding and evolving maze genomes. Configuration parameters (e.g. NEAT and maze genome mutation rates) for both the resource limited version of MCC and the original speciated version (the control in this paper) are identical to those used in chapter 5, thereby isolating effects to the selected diversity preservation method.

6.3 Results

Recall that the aim of these experiments is to investigate whether a simple resource limitation can maintain similar levels of population diversity as speciation, and thus serve as a much simpler and more natural method of diversity preservation. The results of both methods (i.e. resource limitation and speciation) are evaluated by comparing the diversity of mazes both discovered and solved during evolution. Diversity is calculated by measuring the Manhattan distance between the solution paths of every pair of mazes in the population, and dividing by the number of path segments to normalize by path length, thereby accounting for larger mazes that may have longer, but not necessarily more diverse, solutions paths. Formally, the diversity score of a maze is given

by

$$\text{div}(m) = \frac{1}{n} \sum_{i=1}^n \left(\frac{1}{l} \sum_{j=1}^l \text{dist}(m_j, \mu_{i,j}) \right) \quad (6.1)$$

where m represents the maze whose diversity is being measured, m_j is the j -th location in the maze solution path, $\mu_{i,j}$ corresponds to the same solution path index in the maze against which m is compared, l is the length of the solution path, n is the number of mazes in the population, and dist represents the function that measures distance between two solution paths (in this case, the Manhattan distance). The figures in this section compare speciation and resource limitation across a range of metrics, depicting the significance of each measurement with a 95% confidence bound (shown as an error bar) on the means.

Figure 6.1 depicts the average distance between solution paths in mazes discovered by the speciation and resource limitation variants. Imposing a resource limit on mazes leads to significantly more diverse solution paths ($p < 0.001$; Welch's t-test) over the course of a run. Resource limitation also accelerates the rate at which larger mazes are discovered and solved. Figure 6.2 depicts a significantly more rapid maze expansion rate ($p < 0.001$ after batch 100; Welch's t-test) that culminates in mazes that are, on average, twice the size of mazes evolved by the speciated variant. Resource limitation also evolves a much broader range of maze sizes (figure 6.3). Note that the sharp decline in the number of mazes at the end of both experimental variants is caused by underrepresentation of mazes in that size range; runs were terminated before evolution fully explored the space of similarly-sized mazes.

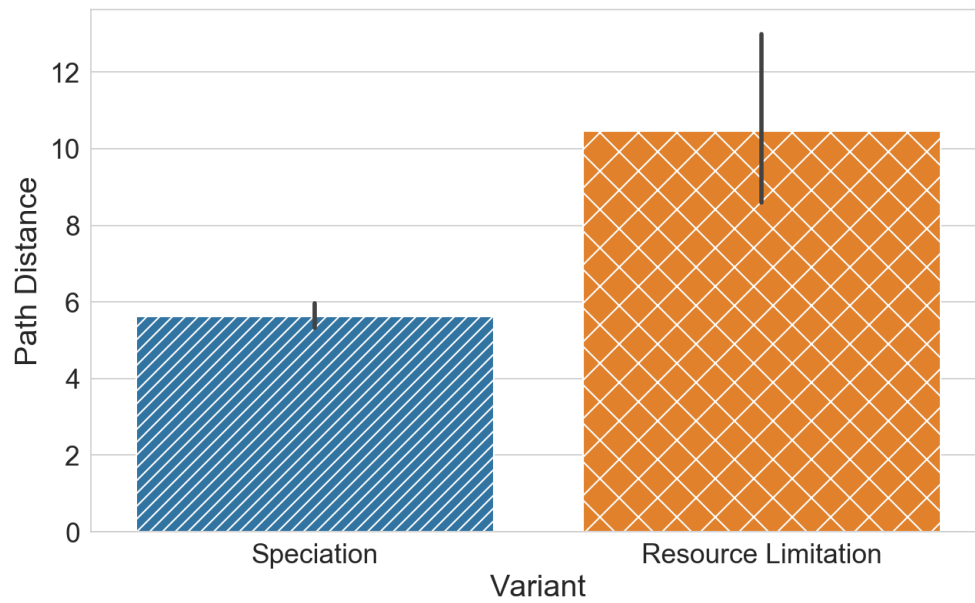


Figure 6.1: **Solution Path Diversity by Experimental Variant.** The average solution path distance between mazes for the speciation and resource limitation variants are compared. The resource limitation variant exhibits significantly higher solution path diversity.

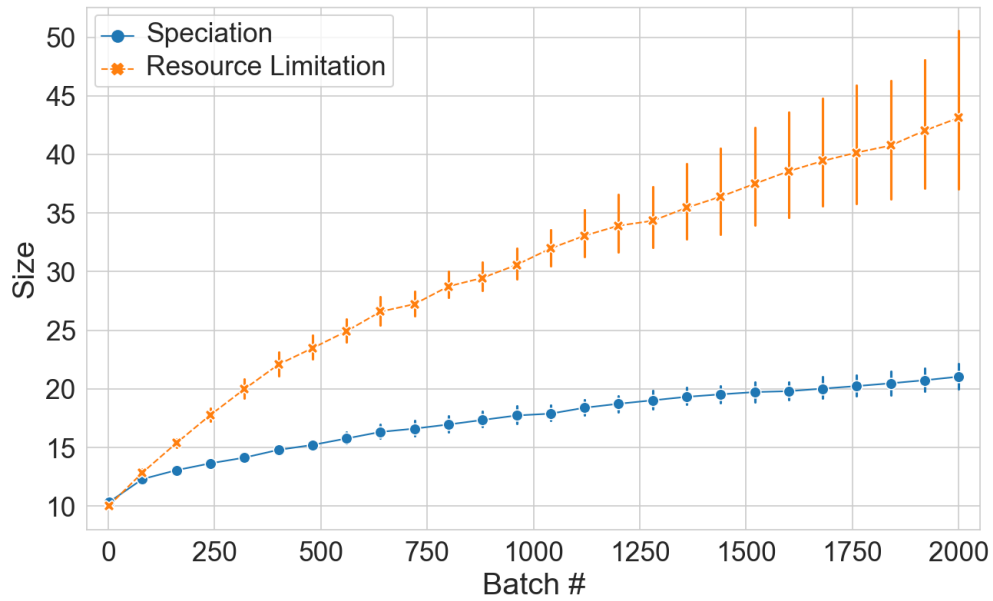


Figure 6.2: **Maze Size Trend.** The average maze size over evolution for the speciation and resource limitation variants are compared. The resource limitation variant discovers larger mazes, with an approximately linear rate of maze expansion throughout evolution.

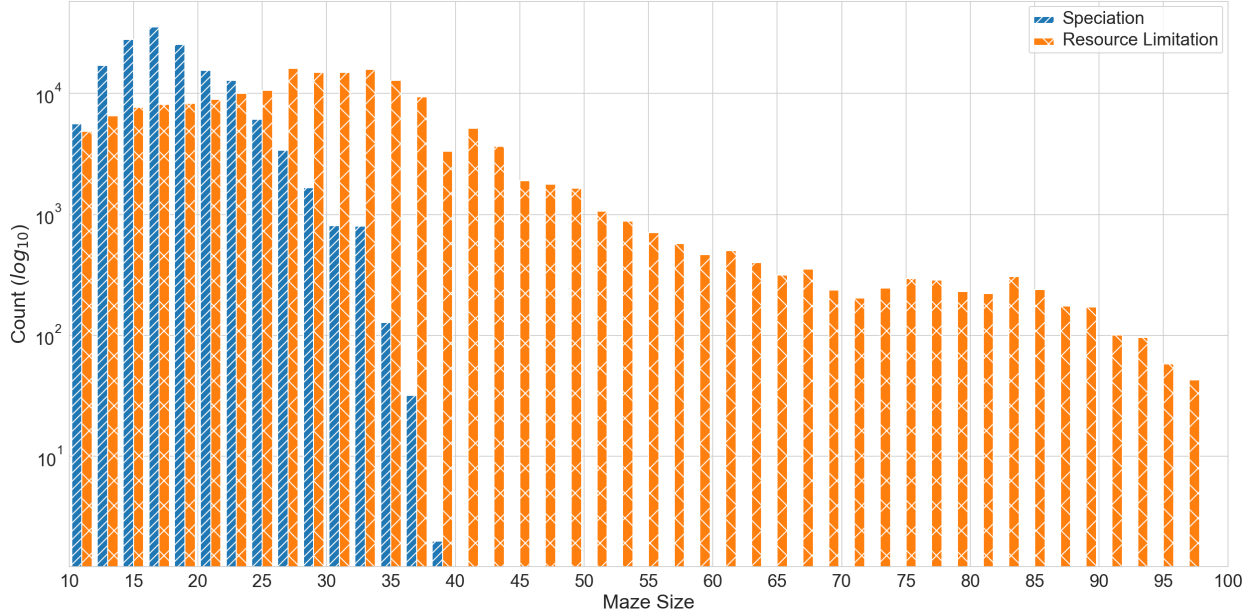


Figure 6.3: Maze Size Distribution. The count of mazes discovered over all runs for the speciation and resource limitation variants are compared. Each column pair depicts a bin of 5 consecutive maze sizes (e.g. the first contains sizes 10–14). Also, the y-axis is log-scaled so that the maze counts at the tail-end of both variants are visible. The resource limitation variant evolved a much wider range of maze sizes, some more than twice the size of the largest mazes discovered using speciation.

Figure 6.4 depicts the rate at which new agent NN connections are incorporated successfully. While agent controllers in the resource limitation variant consistently add structure throughout evolution, they remain significantly more compact than those produced by the speciation variant ($p < 0.001$; Welch’s t-test), despite learning advanced navigation policies for solving larger mazes. This trend indicates that resource limitation may be reducing bloat by finding effective pairings between NNs and maze niches whereas speciation produces arbitrary boundaries for both populations in isolation, and without regard to their interaction dynamics. Evidence for more advanced navigation policies also includes a significant increase ($p < 0.01$ between batches 250 and 600 and $p < 0.001$ thereafter; Welch’s t-test) in the average number of deceptive junctures (figure 6.5) in

the resource limitation variant, where agents at a juncture (i.e. a turn in the maze solution path) are forced to choose one of multiple possible routes with incomplete information of where the chosen route may lead.

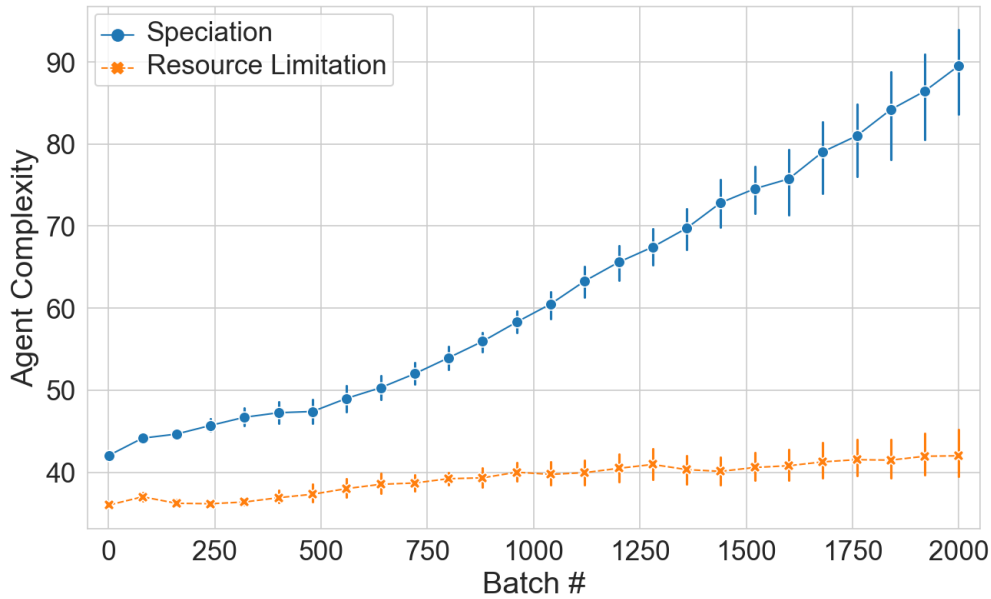


Figure 6.4: **Agent NN Size Trend.** The average agent NN connections over evolution for the speciation and resource limitation variants are compared. The resource limitation variant continues to add structure, but maintains significantly more compact NNs throughout evolution while solving more complex mazes than the speciation variant. This result suggests that resource limitation may reduce bloat by finding effective pairings between maze and agent.

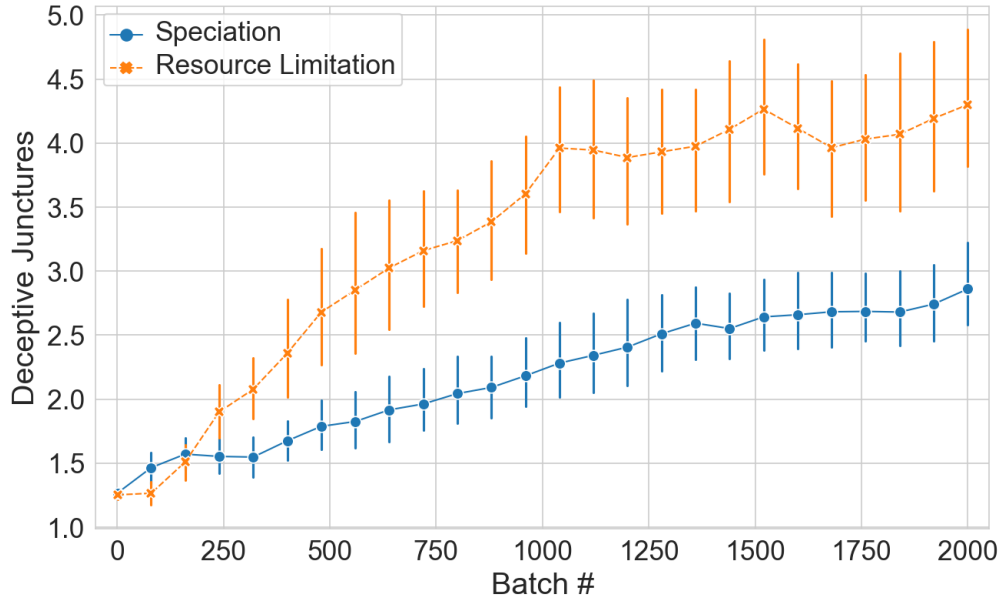


Figure 6.5: **Maze Deceptive Junctionures Trend.** The average number of junctionures (turns) in maze solution paths with multiple possible routes for the speciation and resource limitation variants are compared. A higher number of such junctionures requires a more advanced navigation policy because an agent must decide along which route to continue its trajectory with incomplete information as to where that route may lead. The resource limitation variant discovers solution paths that are more deceptive, along with agent NN controllers that overcome such deception.

Figure 6.6 showcases a sample of agent trajectories (from evolved NN controllers) through mazes evolved by a *single* run of MCC, each varying in size and solution path complexity. All runs of MCC with resource limitation discovered mazes larger than those produced with speciation, with the largest mazes being more than three times the size and with effective agent solutions.



Figure 6.6: Mazes and agent trajectories discovered within a single run of MCC with resource limitation. A sample of eight solution trajectories through different mazes evolved within a single run of MCC are shown. Maze (a) is 11×11 (slightly larger than the 10×10 bootstrap mazes) with 2 waypoints and 2 walls (i.e. wall genes), maze (b) is 15×15 with 4 waypoints and 6 walls and maze (c) is 22×22 with 6 waypoints and 8 walls. Maze (d) is the same size as the largest maze shown in the chapter 5 enhanced maze domain experiments, at 25×25 with 7 waypoints and 8 walls. Mazers (e) through (h) are larger and have more complex solution paths than any previously-demonstrated results in the MCC maze domain. Maze (e) is 28×28 with 7 waypoints and 10 walls, maze (f) is 33×33 with 10 waypoints and 13 walls, and maze (g) is 36×36 with 11 waypoints and 13 walls. The largest maze (h) is 40×40 with 10 waypoints and 22 walls.

6.4 Implications

By restricting the number of times a single maze can be used for satisfying an agent MC, agents are forced to sample a wider variety of mazes that each vary in size and complexity. Those who

are more evolvable are able to exploit underutilized resources, founding new niches and producing offspring who exhibit a similar degree of adaptability. This exploration across niche space creates a diverse curriculum of challenges that facilitate novel and complex agent adaptations, even without directly enforcing diversity through conventional methods such as speciation.

Resource limitation discovers complex mazes with more diverse solution paths than speciation, while also evolving agents capable of rapidly adapting their policies to new challenges. A notable side-effect of resource limitation is an acceleration of the evolutionary process: larger mazes are discovered while NN controller size remains modest, yielding compact encodings that reduce NN bloat while encoding more advanced control policies. The results suggest that resource limitation is a suitable, even superior, replacement for speciation. In addition to offering a more natural form of diversity preservation, resource limitation also alleviates the need for ad-hoc decisions about how to meaningfully measure distance between encodings, and the computational overhead of exhaustively comparing similarity, facilitating the application of MCC to domains for which such a measurement would be uninformative, non-trivial or computationally expensive to execute.

It is also notable that the resultant algorithm is among the simplest in EC. There is no measure of fitness, no ranking, no proportional selection, and no measure of diversity or explicit speciation. And yet it is still producing an open-ended phenomenon of increasing complexity and diversity rarely observed in any other algorithm, hinting that the fundamental ingredients necessary for open-ended evolution to flourish may be surprisingly minimal.

The experiments in the next chapter explore the application of MCC to a robot locomotion domain where body and brain are coevolved. Robot morphologies (bodies) and neurocontrollers (brains) are extremely complex structures, and a meaningful similarity metric for speciation would be both difficult to craft and highly-expensive to compute. As MCC is applied to complex tasks, including robotic control and other practical, “real-world” domains, it stands to benefit from the simplicity

and efficiency afforded by resource limitation as the primary method of diversity preservation.

CHAPTER 7: BODY-BRAIN COEVOLUTION

Maze navigation is a common domain for benchmarking QD algorithms because it is lightweight and easily-interpretable, making it ideal for rapid experimentation and comparison of algorithm variants. The maze domain presented in this dissertation is of particular value for open-endedness research because of its ability to encode navigation tasks of varying difficulty at limitless scale. Despite its utility as a research tool, however, the maze domain has limited real-world significance. To better understand the performance of MCC on a difficult problem of practical interest, this chapter details experiments in a robot locomotion task where MCC coevolves morphology and control – two components that must work in synchrony to ambulate the robot a minimum distance. These experiments are the capstone of this dissertation, and demonstrate how MCC can be harnessed as a powerful tool for open-ended coevolutionary design and optimization.

7.1 Evolutionary Robotics Domain

Evolutionary robotics (ER) is the application of selection and variation processes to the design of robot controllers (brain), morphologies (body) or both [64, 110]. A key focus of the field of robotics in general is designing controllers and body plans that exhibit fluid, robust and principled patterns of movement like those seen in nature, yet this goal remains elusive [17, 70, 89, 90, 116]. ER has the potential to address this shortcoming because it employs computational abstractions of the process that produced robotics benchmarks: natural evolution. However, ER has not enjoyed widespread adoption in the broader robotics community, largely due to its failure to deliver designs with the same robustness and adaptation of its biological equivalent. Solutions discovered by simulated evolutionary processes are often just as brittle and uninspired as those created by mainstream methods. This limitation is especially apparent when evolving both controller and morphology

because a change in one will often deleteriously affect the other [121].

In natural evolution, body and brain are tightly coupled; control strategies persist as a function of their ability to advantageously utilize a morphological trait (e.g. the ability to grasp in primates [5, 120]), while morphology is rewarded to the extent that it permits formulation of novel control mechanisms (e.g. dexterous hands) and yields selective viability. While explicit separation between body and brain is a false dichotomy when considered from a natural perspective, the complex interplay between morphology and control nonetheless provide an opportunity for MCC to discover novel pairings of both as each are evolved to satisfy a mutual MC. Just as agents evolve increasingly complex NNs to discover novel trajectories through progressively more intricate maze structures, controllers can be evolved to exploit ongoing morphological elaborations, and evaluated to the extent that the combination of both meet the MC of continuing functionality in their environment. The intent is for this dynamic to lead to an open-ended, divergent exploration through the space of controller and morphological innovations, where MCC evolves many different strategies for a particular controller and corresponding morphology within a single run.

7.2 Experiment

Soft robotics is a biologically-inspired sub-field of robotics research that has received widespread attention in recent years [71]. While “hard” robots contain rigid links and fixed joints, each with their own actuator that produces a limited range-of-motion, soft robots are composed of compliant materials that offer a theoretically infinite range-of-motion and a high degree of elasticity. This compliance allows soft robots to perform deformation tasks that are impossible for rigid bodies, such as conforming to uneven or rugged terrain [21, 74].

Voxel-based soft robots (VSR) are a particular category of soft robots that have garnered significant

interest in the ER community [160]. They exhibit a modular body plan composed of cubic *voxels*, which provides a flexible sandbox for morphological evolution. Voxels are positioned within a three-dimensional Cartesian grid and each voxel exhibits a variable degree of elasticity, allowing it to expand or contract as a result of forces applied from six neighboring voxels, environmental stimulus or direct control. The synchronized actuation of voxels and the force that they exert on neighboring voxels can produce complex behavioral patterns, including principled locomotion behavior. The versatility of a voxel-based representation is especially advantageous for MCC because voxels can be easily rearranged and the grid can be expanded, permitting a theoretically-infinite range of voxel-based bodies that could be discovered.

The experiments in this chapter use a simulator called *Voxelyze* [65], which is a well-tested and verified VSR simulation platform, and also includes a package for designing and visualizing voxel body structures (called *VoxCad*). *Voxelyze* also has a substantiated history as a simulation engine for ER research. Cheney et al. [15, 16] introduced a method for evolving voxel robot morphologies using CPPN-NEAT [148] (discussed in section 2.6), and demonstrated that CPPNs are able to produce more natural-looking robots with distinct regularities that facilitate principled behavior, attaining higher locomotion performance with regard to speed and distance traveled. While Cheney’s work used a fitness-based EA, it was later shown that non-objective algorithms, including novelty search [98] and surprise search [61], can discover even more diverse morphologies while still maintaining similar functional characteristics.

The aforementioned work focused only on evolving morphology around a fixed control policy (based on voxel expansion and contraction induced by cyclic environmental temperature fluctuations); however, Cheney et al. [17] went a step further, demonstrating how morphology and control can be co-optimized (albeit through a fitness-based process) to produce multiple morphologies, each with a well-adapted controller capable of ambulating its respective body. While Cheney’s work is not the first to simultaneously evolve morphology and control [55, 96, 112, 128], the ap-

plication of co-optimization in a VSR domain is novel and makes it an ideal point-of-comparison for MCC. Voxel bodies are particularly well-suited to demonstrating open-endedness because the voxel grid can be continuously expanded, requiring the evolution of new control policies to exploit the additional material. This process of morphological growth and elaboration is reminiscent of the unbounded complexity increase characteristic of open-ended systems. The MCC experiments in this chapter thus derive inspiration from Cheney’s co-optimization work and use a sample of his results as a baseline for comparison; however, MCC’s method and implications are far different. Instead of rewarding body and brain to the extent that they produce incrementally longer trajectories, MCC only requires that a brain-body pair satisfy a modest MC of ambulation distance, within which there is no preference for a particular design or control strategy. Moreover, all prior research in this domain has restricted body size, thus limiting the number of morphological variations that evolution can discover. These experiments remove that constraint, opting instead for a voxel grid that expands unboundedly throughout evolution, thereby creating a design space in which endless body plans and configurations can be discovered and evaluated.

To facilitate comparison to co-optimization, bodies and brains in the body-brain coevolution domain adopt the structure and encoding introduced in Cheney et al. [17]. Bodies are represented within a three-dimensional grid where a grid cell can be either empty or occupied by a voxel of a given material type. Material is categorized as either *active* or *passive*, where active voxels expand and contract per an input stimulus (e.g. a NN that dictates a contraction value) and passive voxels deform to comply with intrusion from neighboring voxels, but do not initiate structural deformation. The body population consists of CPPNs evolved by CPPN-NEAT, each of which generate a corresponding morphology by querying each cell in the voxel grid, accepting as input the three-dimensional Cartesian coordinates of that cell and the Euclidean distance from the center of the grid, and returning a binary value indicating the presence of a voxel (i.e. whether the cell is empty or not) and, if present, the voxel’s material properties (i.e. active or passive). Figure 7.1 depicts

how the body CPPN produces the resulting morphology. CPPNs are directed graphs that output a geometric pattern built up from a composition of activation functions (which, in these experiments, includes linear, bipolar sigmoid, Gaussian and Sine). Functions that are positioned early in the graph establish a spatial motif on which later functions elaborate to create increasingly complex and regular patterns. This property of CPPNs leads to regular voxel placement with principled groupings of active or passive material. Moreover, CPPNs are *scale-invariant*; consistent patterns of arbitrary dimensionality can be generated simply by querying the spatial substrate at a higher resolution. In this way, material for each cell within a 3-D voxel grid is generated by querying the 3-D substrate at the resolution appropriate for the desired body dimensions. For example, a CPPN that originally generated a $5 \times 5 \times 5$ body (125 voxels) could also produce a $6 \times 6 \times 6$ body (216 voxels) simply by querying each of the 216 voxel coordinates, with no change to the CPPN topology, effectively creating a dynamic and unbounded voxel substrate [127] while maintaining thematically-consistent voxel configurations.

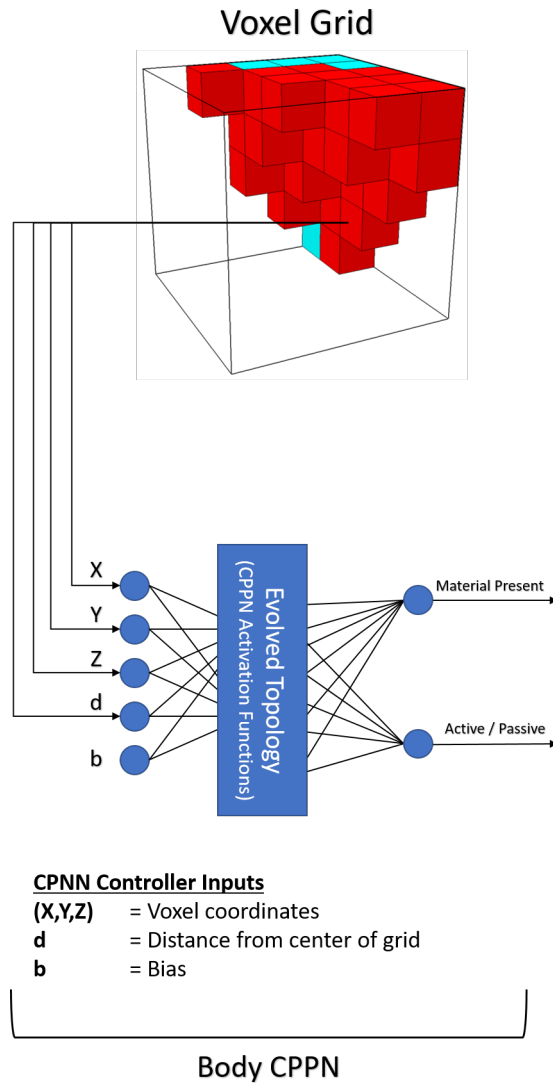


Figure 7.1: **Body CPPN Architecture.** The architecture of a CPPN in the body population is shown. The CPPN queries each location in a three-dimensional Cartesian grid of variable size, taking as input the Cartesian coordinates of each cell along with the Euclidean distance from that cell to the center of the grid, and returning a binary indicator of voxel material presence at the given location and the material type if present (i.e. active or passive). If there is no material present at the location, the material type output is ignored, resulting in an empty grid cell.

Brains, which are also based on Cheney et al. [17], are fixed-topology NNs with four inputs, two

hidden-layer nodes, each with two recurrent connections (one to themselves the other to the opposite hidden-layer node), and one output node. A separate population of CPPNs are evolved to produce NN controllers. A *single* CPPN generates a separate NN controller for every voxel in the body on which it is being evaluated, encoding the presence and binary weight (-1 or 1) of every possible connection between the input and hidden layer (including recurrent connections within the hidden layer) and the weights of the two connections between the hidden and output layer (the connections between hidden nodes and the output node are always present in the NN topology). Brain CPPNs query each location in the voxel grid that contains active material, accepting the three-dimensional Cartesian coordinates of the applicable grid cells and the Euclidean distance from those cells and the grid center as input, and outputting the resulting NN controller connections and weights for each voxel. While evolving NNs for each voxel using a direct-encoding [152] would impose a substantial computational expense (because every voxel in each body would require a unique NN controller in the brain population), querying every location in the body with a single evolved CPPN to *generate* an NN for each voxel is comparatively cheap, allows a single CPPN to be evaluated on bodies of varying sizes (i.e. by simply querying each voxel to generate a voxel-specific NN) and also exploits the CPPN’s intrinsic ability to capture regularity. Additionally, prior methods in generative multi-agent reinforcement learning [27] have demonstrated how CPPNs learn a *policy geometry* that dictates how smaller components should behave based on their position within a larger structure. Just as CPPNs generate spatial regularities through function composition and learn policy geometries from complex, interconnected formations, so too can they generate NNs that effect regular and coordinated control policies for each voxel based on its location within the morphology. Figure 7.2 depicts how the brain CPPN generates NN connection weights.

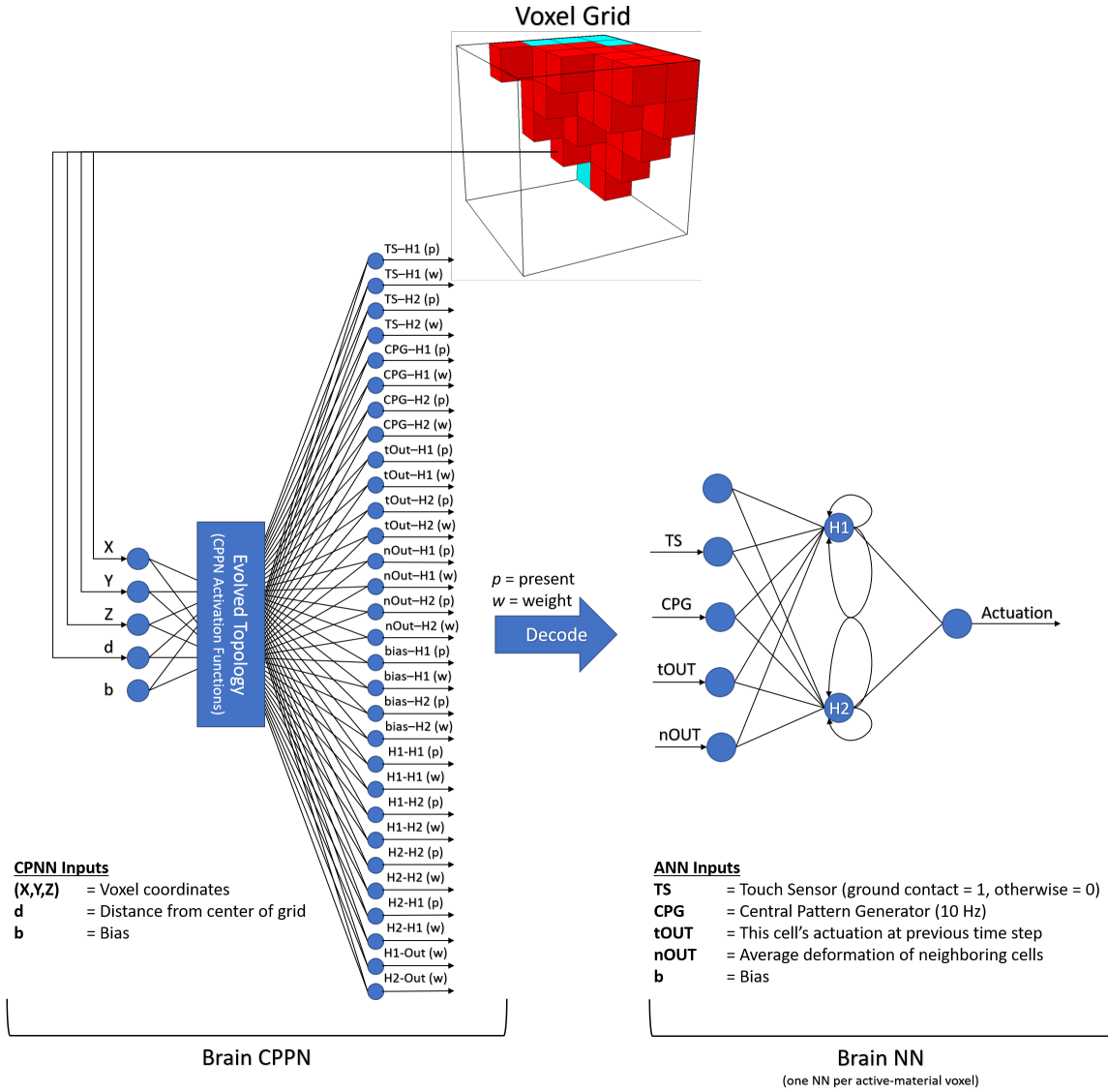


Figure 7.2: Brain CPPN and NN Architecture. The architecture of a CPPN in the brain population is shown. The CPPN queries each location of a three-dimensional Cartesian grid that contains active material, taking as input the Cartesian coordinates of the applicable cells along with the Euclidean distance from those cells to the center of the grid, and returning a binary presence indicator (0,1) and binary weight (-1,1) for each input-hidden and hidden layer connection, and weights for hidden-output connections (which are always present). A fixed-topology NN is generated for each active-material voxel. The NN is supplied with a binary value indicating ground contact, the output of a 10 Hz CPG based on the simulator clock and the actuation of the current cell in the last time step as well as an average deformation of the neighboring cells. The NN outputs the volumetric actuation of the given cell at the current time step.

The first NN input is driven from a touch sensor that registers 1 if the voxel is in contact with the ground and 0 otherwise. A central pattern generator (CPG) supplies the second input, deriving its initial value from the global simulator clock, and scaling it to the range $[-1, 1]$ within a fixed 10 Hz cycle. CPGs have frequently been utilized for producing walking gaits [69, 103, 104, 126, 142] because they generate regular oscillatory patterns that can be harnessed for creating effective locomotion strategies. The final two NN inputs accept the deformation value of the given voxel in the previous time step and the average deformation of the six neighboring voxels in the current time step. The output returns the volumetric actuation of the voxel at the current time step. All actuation values are in the range $[-1, 1]$, where negative values induce contraction and positive values result in expansion.

In the body-brain experiments, novelty search evolves CPPNs that generate NN controllers capable of ambulating compact bodies a minimum distance of 15 units (which is the MC for reproduction). The initial voxel grid is of length 5 along all three dimensions, creating 125 possible locations for voxels. Ten randomly-generated CPPNs generate morphologies with varying voxel placements and material properties. Novelty search executes with a population of 100 CPPNs (a full list of bootstrap parameters can be found in appendix B) and characterizes behavioral novelty based on the endpoint of a trajectory. The bootstrap halts when 20 distinct CPPNs are evolved to generate brains capable of ambulating one or more bodies the minimum distance. The 20 brain CPPNs and 10 body CPPNs are used to seed their respective MCC queue.

Experiments consist of 20 runs of 2,000 batches each. The brain queue begins evolution with 20 CPPN genomes and has a maximum capacity of 150, while the body queue starts with 10 CPPN genomes and can grow to 30. These population size constraints were imposed to ensure experiment tractability given the computational expense imposed by high-fidelity Voxelyze simulator. Both populations employed identical parameters, with a 0.6 probability of mutating a CPPN connection weight, 0.1 probability of adding a connection, 0.01 chance of adding a node and a 0.005

probability of deleting a connection. Additionally, the body population has a 0.01 probability of resizing the dimensions of the voxel grid. If a resize mutation is applied, it has a 0.8 probability of increasing the grid size by one unit along all three dimensions, and a 0.2 probability of decreasing the grid size by an equivalent amount.

Brains are allotted 2 simulated seconds to meet the locomotion MC of 15 units as measured from the body's center-of-mass. Similarly, the body MC is to be ambulated the same distance by at least one brain in the population. Resource limitation (introduced in chapter 6) is used to preserve diversity by allowing a single body to be used by no more than 5 brains for satisfying their MC.

7.3 Results

In the maze navigation domain, MCC evolved a broad diversity of large and complex maze environments and effective solutions, all within the scope of a single run and without any indication of convergence. While maze navigation is a simple benchmark, the results in this section demonstrate the application of MCC to the much more difficult domain of robot locomotion. The results demonstrate that MCC is able to discover a range of diverse and increasingly complex morphologies along with principled control policies, thereby suggesting the utility of MCC for problems of practical interest. Error bars on all figures depict the significance of each measurement with a 95% confidence bound on the means.

Figure 7.3 depicts the average minimum, mean and maximum body size over evolution. MCC discovers increasingly complex morphologies while also maintaining a breadth of voxel body sizes (the distribution of which is shown in figure 7.4) – a trend that is reminiscent of the open-ended divergence observed in the maze domain. Moreover, expanding the voxel grid creates additional cells in which material can be placed, thereby changing body dynamics through altering mass dis-

tribution and potentially adding active (i.e. self-actuating) material that must work in tandem with existing structure. The monotonically-increasing body size trend indicates that MCC is simultaneously evolving CPPNs that are resilient to large-scale disruptions by generating brains that exhibit flexible control policies.

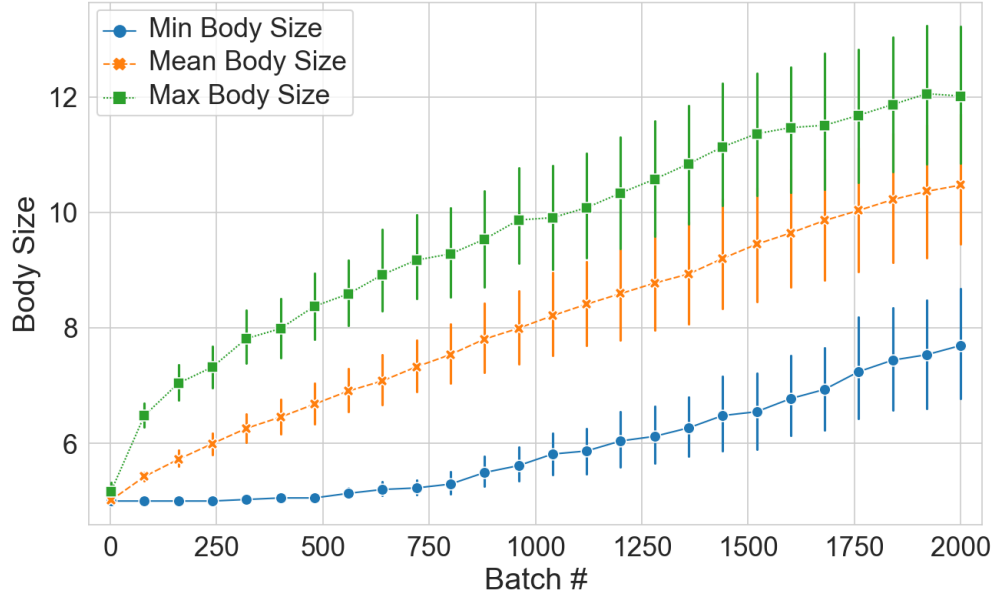


Figure 7.3: **Body Size Trend (averaged over 20 runs).** The average minimum, mean and maximum body size over evolution is shown. In all cases, MCC continually discovers more larger and invariably more complex bodies, along with brains that effect successful locomotion.

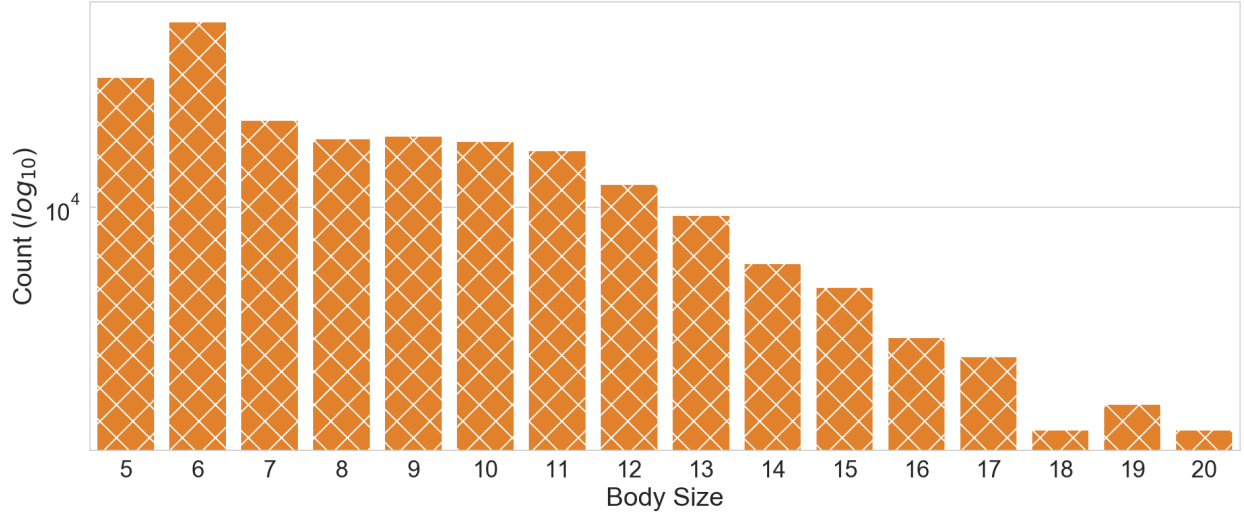


Figure 7.4: **Distribution of Body Sizes.** The count of bodies collected over all runs is shown. Note that the y-axis is log-scaled so that body counts at the tail end of the range are visible. Expanding the voxel grid not only adds additional material cells, but it can also alter mass distribution and body dynamics depending on material type and placement. MCC discovers a range of body sizes, indicating that it is evolving brains that adopt flexible control policies, enabling them to adapt to new morphological configurations.

Measuring the distance that a body traverses within a trial is an intuitive method of evaluating alignment between control policy and morphology, and one that is well-represented in QD and ER literature [17, 79, 158]. To gauge the longer-term ability of controllers to effect principled locomotion, those who ambulated bodies the minimum-required distance during evolution (where trials were constrained to 2 seconds with an MC of 15 voxels traversed) are evaluated on the same body for an extended duration of 15 seconds (23,416 time steps) and the total distance traveled during each simulated trial is recorded.

Figure 7.5 compares total ambulation distance for MCC-evolved bodies to the distance ambulated by the highest performing bodies from each of 30 evolutionary runs of co-optimization [17], which explicitly optimizes for distance traversed. The body and brain genomes evolved by co-

optimization are evaluated under identical conditions as the original experiments in [17], but with a trial duration of 15 seconds to match the extended duration MCC evaluations. While MCC evolved a range of body sizes, only those of size $10 \times 10 \times 10$ (which is the body dimensions used in [17]) are used for comparison. Recall that beyond the locomotion MC of 15 voxels, MCC imposes no additional requirements on performance, instead relying on resource-constrained drift to uncover new elaborations on form and function. Yet even in the absence of a guided search process, MCC evolves controllers and morphologies that significantly exceed those evolved by co-optimization in their ability to effect sustained locomotion ($p < 0.001$ for all variants; Welch's t-test). Moreover, because body dimensions are variable, MCC is able to search multiple divergent morphological lineages simultaneously. Figure 7.6 demonstrates that MCC evolves effective control for bodies across of range of differing sizes, in some cases exceeding more than 140 voxel lengths traversed. This behavior persists even in larger bodies, which have substantially more material that must be controlled in such a way as to produce coordinated action.

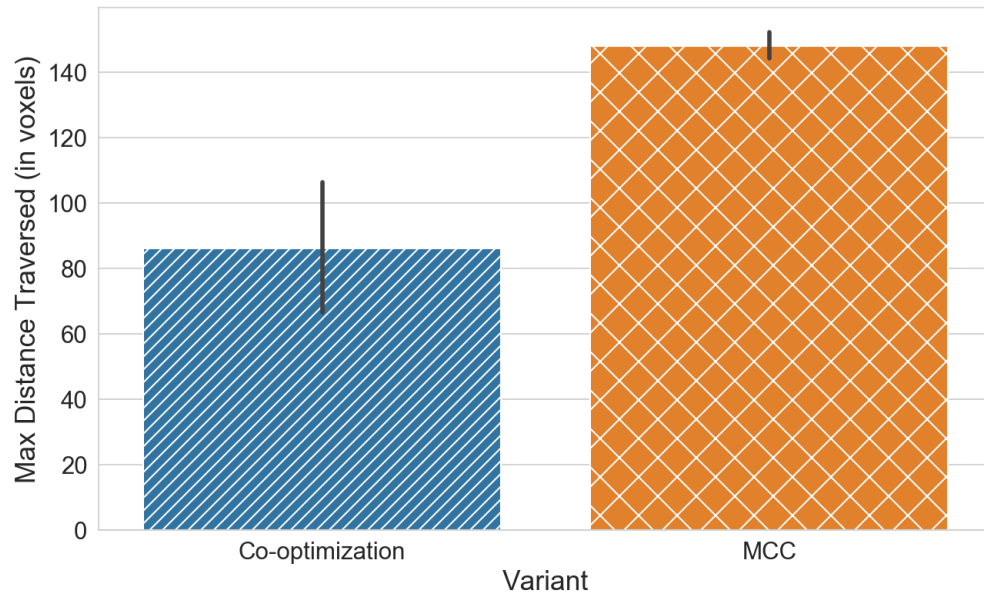


Figure 7.5: **Ambulation Distance Comparison with Co-optimization [17]** The average distance traversed by bodies evolved by MCC is shown in comparison to those evolved by co-optimization. To compare equivalent morphologies, MCC-evolved bodies, which covered a range of varying sizes, from which only size $10 \times 10 \times 10$ individuals were selected for this comparison. The primary result is that MCC produces body-brain pairs that cover a greater total distance than those evolved by co-optimization, which explicitly optimizes for distance traveled from the origin.

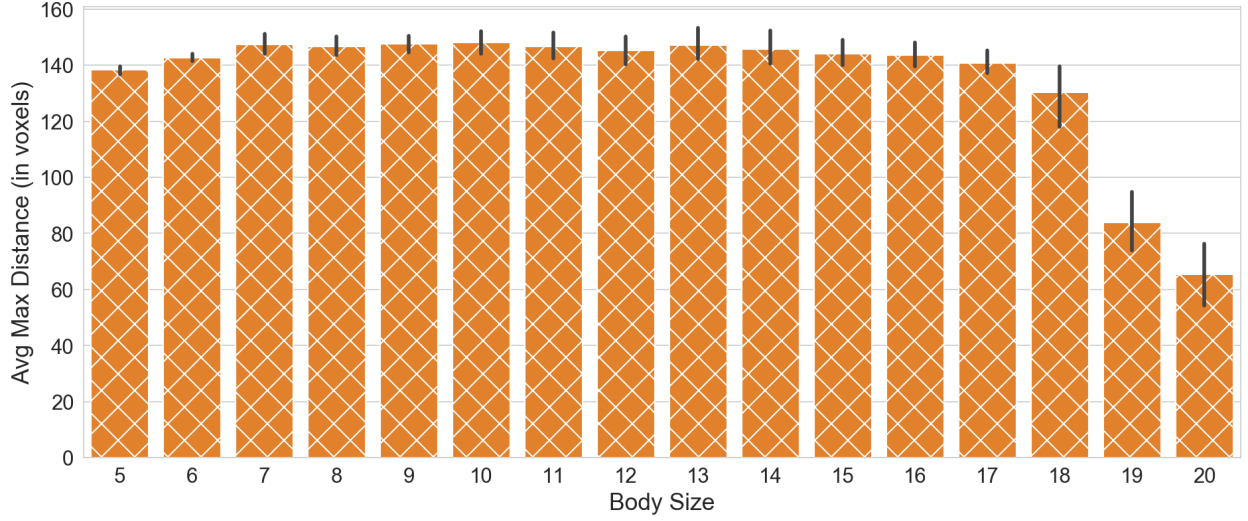


Figure 7.6: **Ambulation Distance by Body Size.** The total distance traversed by MCC-evolved bodies across a range of sizes is shown. A unique aspect of MCC is its ability to search multiple divergent paths simultaneously. In this domain, that characteristic is demonstrated by MCC’s ability to collect bodies of varying size and configuration, each with effective control that matches a much narrower range of solutions produced by co-optimization.

MCC also discovers a diversity of viable control strategies that ambulate bodies along novel trajectories. The following trajectory analysis computes the Euclidean distance between corresponding pairs of points in two trajectories at equal time steps, and averages over the trajectory length and also over the number of distinct trajectories (which are produced by a paired body and brain). Formally, trajectory diversity is given by:

$$\text{div}(t) = \frac{1}{n} \sum_{i=1}^n \left(\frac{1}{p} \sum_{j=1}^p \text{dist}(t_j - \mu_{i,j}) \right), \quad (7.1)$$

where t is the trajectory whose diversity is being measured, t_j is the position of the body’s center-of-mass at time step j , $\mu_{i,t}$ corresponds to the position along a comparison trajectory i at time step j , p is the number of time steps in the trial, n corresponds to the number of trajectories, which was

produced by a unique body-brain pair that was also successful in meeting its ambulation MC, and *dist* represents the Euclidean distance function applied to each point along the trajectories. This approach has been used in the QD literature to characterize and compare behavior [129], but here it is only used to analyze the results of a method that has no BC.

Figure 7.7 depicts the trajectory diversity for MCC compared to that of co-optimization. As in the distance analysis, body sizes are limited to $10 \times 10 \times 10$ to align with the dimensions used by co-optimization experiments. MCC produces more diverse control policies than co-optimization ($p < 0.001$; Welch's t-test), effectively discovering multiple ways to satisfy the MC. MCC's propensity for discovering novel locomotion is further reinforced in figure 7.8, which depicts the distribution of trajectory end-points for MCC and co-optimization. While control policies evolved by co-optimization produce trajectories that remain primarily in the vicinity of the starting location (which is the origin, $(0, 0)$), MCC evolves policies that more thoroughly explore the environment.

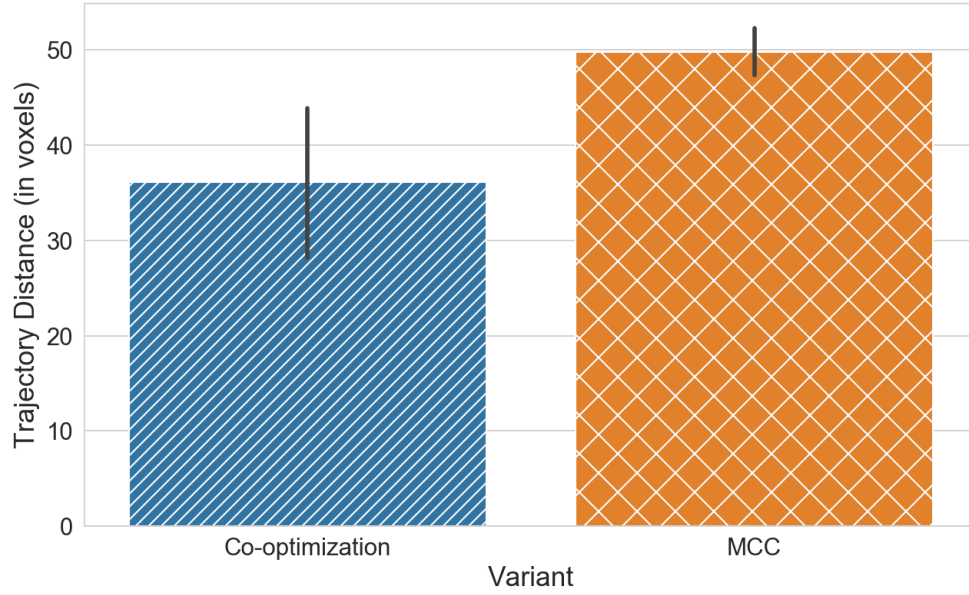


Figure 7.7: **Trajectory Diversity Comparison with Co-optimization.** The average trajectory diversity for MCC with body size $10 \times 10 \times 10$ is shown in comparison to co-optimization. Diversity is measured by computing the average Euclidean distance between all pairs of trajectories. MCC produces significantly more diverse trajectories than co-optimization ($p < 0.001$; Welch's t-test), demonstrating its ability to evolve artifacts that are functionally-equivalent while also more diverse.

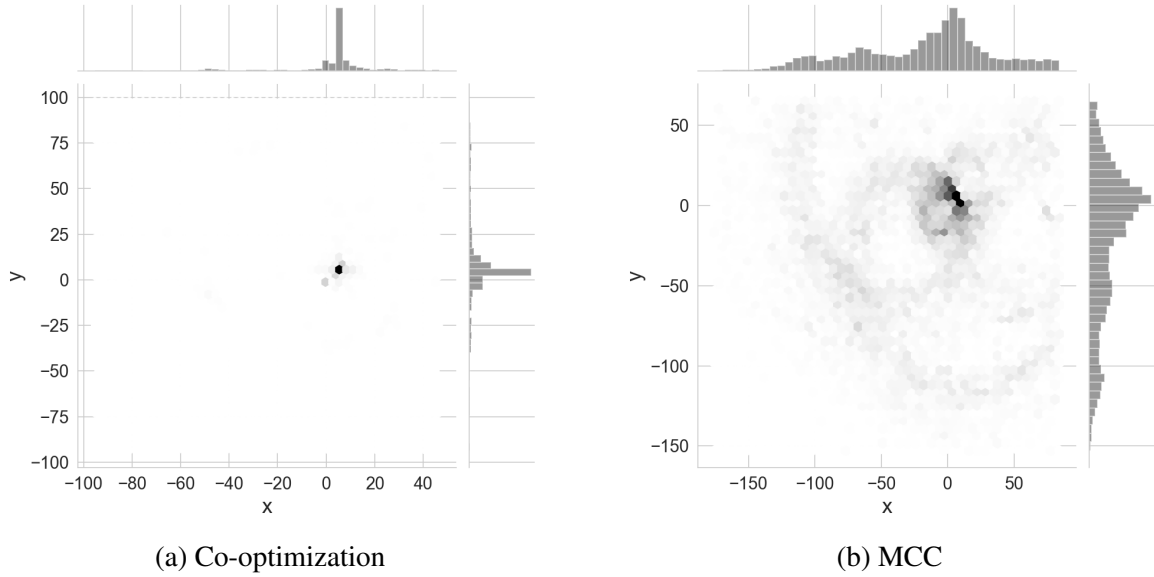


Figure 7.8: **End-Point Distribution Comparison.** The distribution of trajectory end-points for MCC and co-optimization are shown, along with histograms that depict the number of end-points in various regions along each axis of the 2-D environment. While the end-points of trajectories produced by co-optimization tend to remain near the origin (which is at $(0, 0)$), MCC-evolved control policies produce trajectories that culminate in a noticeably more diverse set of end-points.

These results exhibit a particular strength of QD algorithms in general, that is, collecting functional diversity in a *single* evolving population. However, MCC takes this paradigm a step further by collecting diversity along two separate coevolving lineages within the same run. Figure 7.9 demonstrates that such novel trajectories were not limited to a particular body size, but persisted across morphological dimensions.

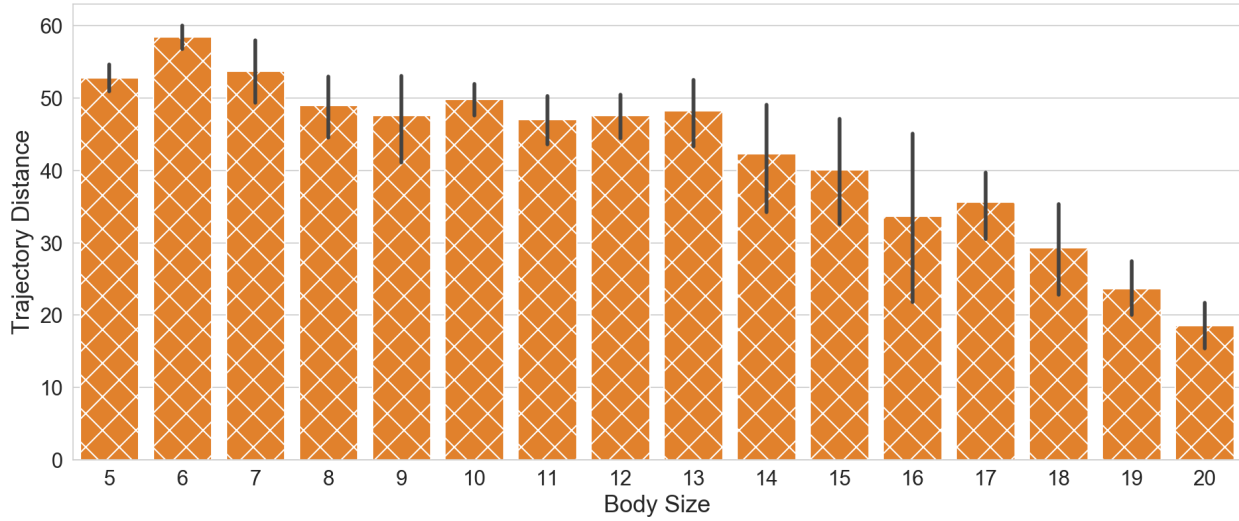


Figure 7.9: **Trajectory Diversity by Body Size.** The trajectory diversity for MCC across a range of body sizes is shown. This result showcases MCC’s ability to collect a diversity of control policies that successfully ambulate a similarly-diverse array of morphologies.

In the body population, MCC produced novel morphological variations within each body size, thereby presenting a unique set of ambulation challenges to brains. Diversity is measured by comparing the material at each location within the three-dimensional voxel grid to equivalent locations of all other bodies in the population. Each mismatch counts as a unit difference. For example, if a body contains active material at coordinate $(1, 3, 2)$, and the body to which it is compared contains passive material or no material (i.e. an empty grid cell) at the same location, the mismatch count is incremented by one. Each body is compared cell-by-cell to every other body in the population, averaging over the number of bodies to derive a material difference score. Formally, body diversity is given by:

$$\text{div}(b) = \frac{1}{n} \sum_{i=1}^n \sum_{x,y,z=1}^{l \times w \times h} V(b_{x,y,z}, \mu_{x,y,z}^i),$$

$$V(p, q) = \begin{cases} 1, & \text{if } p = q \\ 0, & \text{otherwise} \end{cases}, \quad (7.2)$$

where b is the voxel body whose material diversity is being measured, n is the number of bodies in the population, l , w and h are the length, width and height of the three-dimensional voxel grid, $b_{x,y,z}$ specifies the coordinate of material in body b , $\mu_{x,y,z}^i$ corresponds to the same coordinate in comparison body μ^i and V is a conditional that yields 1 if the material at the specified location is the same in both bodies and 0 if not.

Figure 7.10 depicts the average minimum, mean and maximum material difference over evolution (the y-axis is log-scaled so that the minimum material difference trend is visible). In all cases, there is a noticeable upward trend in morphological diversity – an indication of sustained divergence in the body population. A similar dynamic is observed in figure 7.11 where diversity increases monotonically with body size. Figure 7.12 depicts the average material difference trend for a selected body size (size 11), demonstrating how MCC pushes the search toward higher complexity artifacts, and subsequently elaborates within the newfound design space.

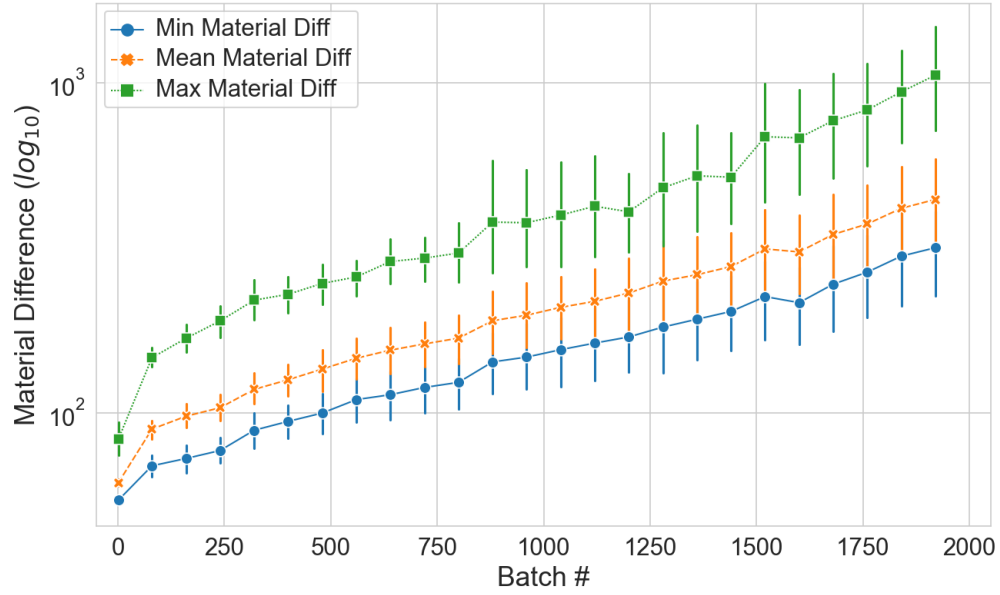


Figure 7.10: **Morphological Diversity Trend.** The average minimum, mean and maximum material difference over evolution is shown. Morphological diversity is characterized by average, per-voxel material difference. Enlarging the voxel grid leads to more positions in which to place material. MCC exploits these opportunities for morphological variation, evidenced by a gradual upward trend in diversity – a demonstration of continued divergence in the body population.

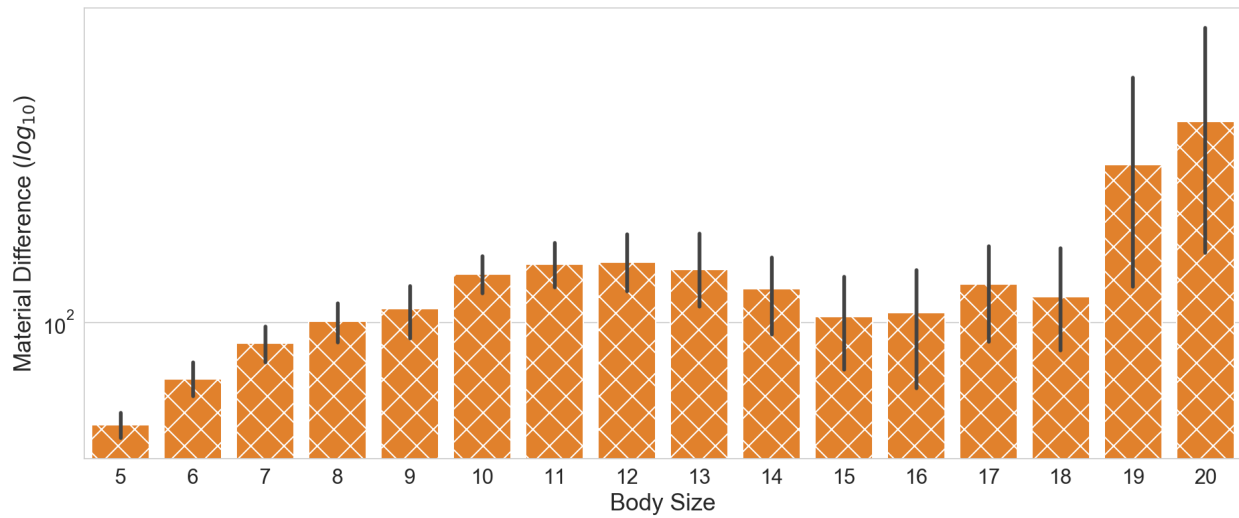


Figure 7.11: **Morphological Diversity by Body Size.** The morphological diversity over a range of body sizes is shown. Larger bodies tend to have more diverse structure because there are more options for material placement.

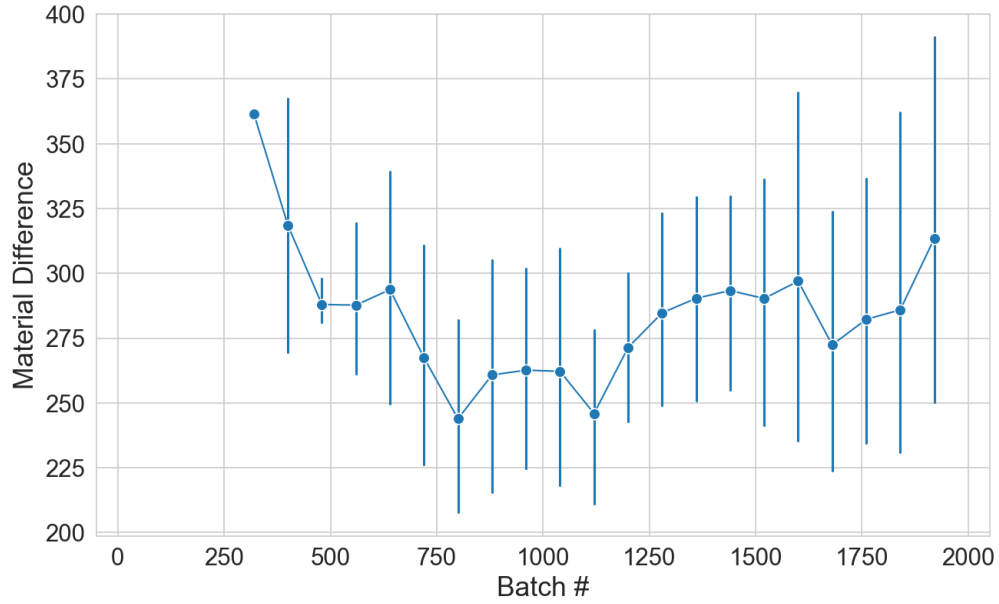


Figure 7.12: **Morphological Diversity Trend for Body Size $11 \times 11 \times 11$.** The average morphological diversity over evolution for body size $11 \times 11 \times 11$ is shown. Bodies of this size were first discovered in batch 320 (recall evolution began with bodies of size $5 \times 5 \times 5$), initially undergo a sharp dip in diversity, then experience a gradual diversity increase as MCC discovers functional elaborations within the space of similarly-sized body plans. A similar trend persists across most body sizes discovered by MCC.

A sample of bodies that were evolved by MCC in a *single* run are shown in figure 7.13, demonstrating MCC’s ability to produce a wide variety of body sizes and material configurations, each with well-adapted controller that produce synchronized material deformations in such a way as to locomote bodies a non-trivial distance.

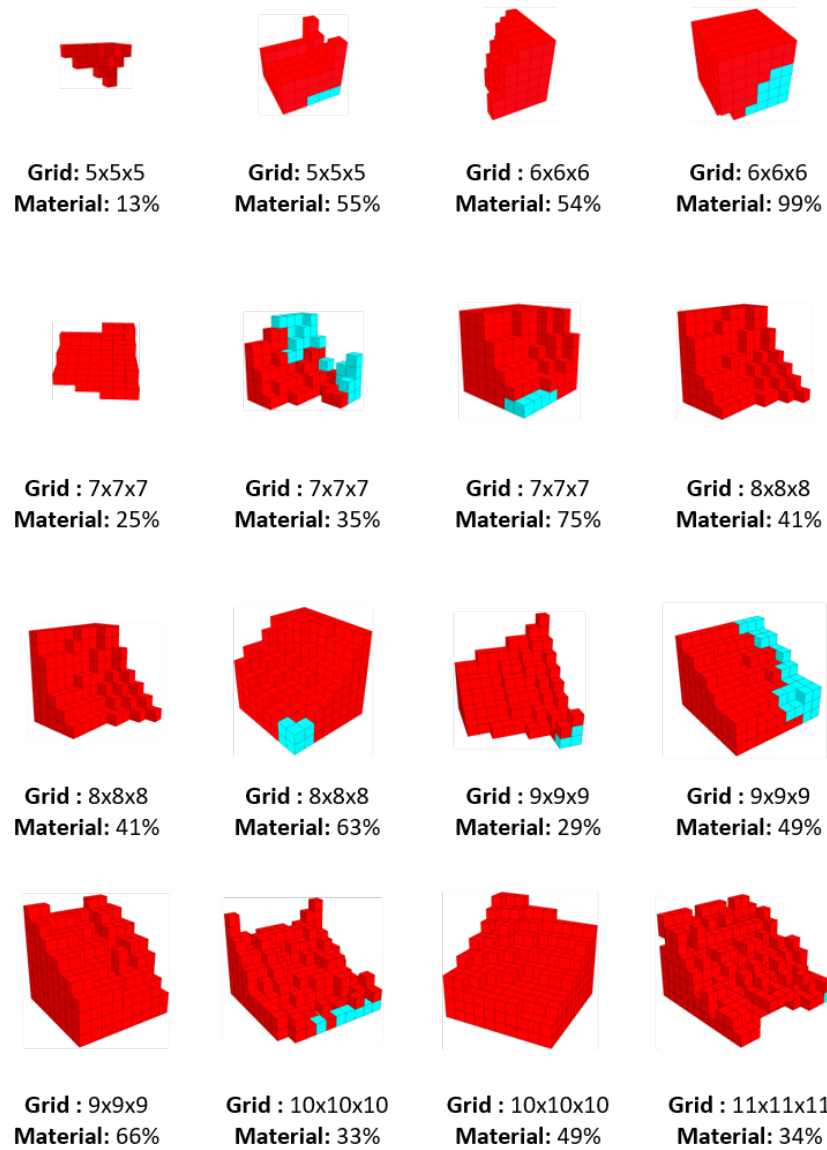


Figure 7.13: **Sample of Evolved Bodies.** A sample of 16 voxel bodies evolved by MCC in a single run is shown. Bodies vary widely in size and configuration, yet each is paired with a set of voxel-specific controllers that produces principled locomotion.

7.4 Implications

The diversity of morphologies and high-performing control policies collected by MCC demonstrate its effectiveness beyond simpler benchmark problems like maze navigation, thus reinforcing the potential of MCC as a general-purpose open-ended search algorithm. MCC evolved NNs to exploit ongoing morphological elaborations, producing novel morphologies with effective control, even without the aid of a locomotion objective to directly reward distance traversed, or a behavior characterization from which to encourage novel end-points and indirectly promote exploration of a more distant expanse.

A particularly interesting result is MCC’s ability to match or outperform fitness-based search on tasks that are conventionally thought to be the domain of objective-oriented optimization. An intuitive measure of fitness for evolving locomotion ability might be to reward distance traveled, and the same metric could also serve as a reasonable interpretation of morphological quality given a fixed control policy (i.e. improvements in locomotion are attributable to a refined body design). MCC incorporates neither construct, but is nonetheless unencumbered by its unilateral rejection of conventional optimization practice, producing *quality diversity* that is reminiscent of the algorithmic insights from which it is inspired. Yet the implications of MCC as a general purpose algorithm are greater than those of its QD counterparts. In particular, while QD algorithms might discover multiple diverse locomotion strategies for a fixed body or a collection of novel morphologies for a static control policy, MCC discovers both simultaneously and with unbounded size.

Additionally, the reliance of QD algorithms on an explicit description of behavior is fundamentally limiting in a domain where a complete list of the dimensions of phenotypic and behavioral variability either cannot be fully captured a priori or are intractably expensive to compute. For example, if morphological novelty is characterized by two material types (i.e. active and passive),

then once the full space of materials in a fixed-size grid is sampled, novelty pressure is reduced. If the grid is expanded and larger body plans are evolved, behavior comparison can quickly become a computational bottleneck. Conversely, if other properties of body dynamics, such as material deformation characteristics, change during evolution but are not incorporated into the behavior descriptor, then QD methods may overlook an important trait that would otherwise lay down a path toward unanticipated elaboration. This behavioral agnosticism is one of MCC's most distinguishing features, which when combined with an MC-constrained coevolutionary drift facilitates unbounded discovery in a general context.

CHAPTER 8: DISCUSSION

Formalizing and reproducing the dynamics that lead to continual, open-ended innovation has long been an ambitious and elusive pursuit that, until recently, has been constrained to theoretical and abstract inquiry, typically within the domain of alive worlds. The main contribution of this research is a general algorithmic framework capable of facilitating the production of open-ended innovation in a domain-independent manner and without artificial constructs, such as quantification of fitness and behavior. A key insight of this framework is the extent to which coevolutionary interactions can lead not only to a ratcheting effect in complexity, but also to novel elaborations that occur as a result of genetic drift within both populations, interlocked by the satisfaction of the MC with respect to one another.

Throughout this work a significant focus is placed on architecting novel and flexible domains that are suitable for demonstrating open-ended search. A hallmark trait of open-endedness in nature is the continual production of complex organisms that exploit an increasingly diverse collection of environmental niches. Just as natural evolution grew a vast diversity of life from humble origins and over a single evolutionary timeline, so should MCC generate abundantly diverse and non-trivial artifacts from minimal structure and within the scope of a single run. The maze domain provides a simple and intuitive method of visualizing this evolutionary dynamic. In the initial maze experiments in chapter 4, coevolutionary progress was demonstrated by adding walls that complicate navigation and encourage the emergence of complex navigation policies among agents. While mazes in this initial experiment are constrained to a predetermined size, the extent to which they are able to increase in complexity at all, in the absence of fitness and behavioral guidance, is an initial hint of MCC's ability to coevolve principled and non-trivial artifacts.

The initial maze domain experiments demonstrate MCC rapidly reaching the domain complexity

ceiling (i.e. the maximum number walls that can be placed within a fixed-size enclosure), and while compelling, these results leave a pressing question unanswered: *what sort of maze and navigator pairings would MCC discover if mazes could expand throughout evolution?* The enhanced maze domain experiments in chapter 5 address this question by allowing mazes to expand without limit. Additionally, instead of merely adding walls to indirectly complicate navigation, the solution path itself is evolved as a series of waypoints that create winding paths and deceptive corridors. Within a single run, MCC discovers mazes of varying size and structure, each with effective solutions resulting from increasingly advanced navigation policies evolved in the agent population. Importantly, this trend continues for the entirety of each evolutionary run, with no indication of leveling off. While existing QD algorithms are capable of evolving functional diversity, they do so within a single population (i.e. with no coevolution) and by relying on an explicit characterization of behavior to promote divergence [123]. These experiments are the first to illustrate a characteristically open-ended process in a practical domain (i.e. outside of an alive world) where both populations continually produce complex and novel forms, relying only on coevolution drift subject to an MC.

A foundational tenet of MCC is that open-ended divergence can be induced by coevolving two populations subject to a mutual MC of baseline functionality. The maze experiments demonstrate how this simple framework can discover surprising results that are both principled and diverse. However, while MCC's primary selection apparatus is organic, its initial method of diversity preservation, which explicitly groups individuals into a predetermined number of species based on a measure of genetic distance, is ad hoc. In nature, diversity is encouraged through a competition for limited resources where each niche has a finite capacity and organisms must find new niches to persist. In MCC, this concept is realized through imposing a resource constraint on the maze population (chapter 6) such that only a finite number of agents can use a distinct maze to satisfy their MC. The results demonstrate that resource limitation is an effective, even superior, replacement for speciation, producing larger mazes with more diverse solution paths with viable

navigation policies within the same evolutionary time frame as prior experiments. A convenient property of this more organic approach to diversity preservation is that it avoids the computational overhead of an exhaustive comparison of genetic similarity, while also mitigating the need for a method by which to meaningfully assess similarity – a potentially difficult task for more complex domains that exhibit multiple dimensions of variation. Resource limitation is thus recommended as the primary method of diversity preservation in MCC and its future variants.

As a general framework for open-ended search, MCC should be expected to produce divergent artifacts in any domain wherein two populations can be set up to coevolve in accordance with a mutual MC. The maze domain provides an intuitive and light-weight benchmark, but still leaves a question about its application in more complicated domains. This work culminates by applying MCC to a difficult task in soft evolutionary robotics (chapter 7), where voxel-based bodies and brains are coevolved to effect locomotion and meet an MC of distance traveled. Bodies are rewarded to the extent that their morphology is amenable to ambulation, and brains are rewarded in accordance with whether they are able to effect principled coordination in body components. Moreover, bodies are permitted to expand along all three of their morphological axes, creating new challenges for brains by adding material that requires principled control. As in the enhanced maze domain, MCC demonstrated the ability to collect a range of diverse and increasingly complex morphological structures, each of which met the MC of ambulation. Moreover, MCC discovered control policies for all body sizes that were superior, in terms of resulting locomotion distance, to those evolved by fitness, which explicitly rewarded for distance traveled on but one body size. The evolutionary robotics locomotion task provides a relevant exemplar for applying open-endedness to real-world problems, and empirically demonstrates that MCC is capable of coevolving effective solutions for practical domains at the cutting-edge of science and engineering research.

Perhaps the most exciting implication of MCC is that of capturing a small glimmer of the creative spark exhibited by natural evolution. While the creativity of nature remains unmatched by any

algorithm to date, the hope is that this investigation leads to a deeper understanding of open-endedness and the processes that lead to it, through a framework that is able to reproduce at least a portion its dynamics.

CHAPTER 9: CONCLUSION

The primary contribution of this dissertation is a novel algorithm called Minimal Criterion Coevolution (MCC) (1), which demonstrates an approach to open-ended search that relies on a process of coevolutionary drift subject to a minimal criterion. MCC overcomes representational limitations of existing objective and non-objective algorithms by jettisoning any notion of a fitness function or a behavior characterization, thereby scaling to produce an open-ended process of search space exploration and discovery.

A novel maze domain (2) was introduced where mazes added walls to complicate navigation. MCC coevolved a population of mazes with a population of maze-navigating agents, each subject to an MC of successful navigation. The results demonstrated that MCC is capable of evolving complex maze environments (i.e. with many walls) with effective navigation strategies.

To further evaluate the open-ended capacity of MCC, an enhanced maze domain (3) was introduced where mazes are permitted to continually expand. Instead of relying on wall placement to craft a solution path, this new domain directly evolves the path through a series of connected waypoints, allowing evolution to morph the trajectory in non-trivial ways. The results reveal that MCC exploits this unbounded domain to create increasingly large and complex mazes along with effective solutions in the agent population, and with no indication of evolutionary stagnation.

During its initial evaluation, MCC used speciation, a common method of diversity preservation in EC, to preserve genetic diversity in both populations. This mechanism is ad-hoc for an algorithm that is otherwise inspired by the simplicity of reproductive viability in nature, which instead uses the intrinsically-finite carrying capacity of niches to maintain diversity by promoting divergence. Speciation in MCC is thus replaced by an alternate form of diversity preservation through resource limitation (4). In the maze domain, mazes adopted a resource limit by constraining the number

of times that successful navigation on a distinct maze could be used for satisfying an agent MC. Resource limitation not only led to the evolution of more diverse mazes and agents, but also discovered more efficient agent encodings (evidenced by smaller NNs) and evolved more complex mazes during the same evolutionary time frame, effectively speeding up evolution.

Finally, MCC was demonstrated on a difficult, real-world evolutionary robotics task (5) where voxel-based robot bodies were coevolved with brains that were required to ambulate bodies a minimum distance. Bodies could morph in arbitrary ways by rearranging material cells, and could also grow in size by lengthening robot morphology along all three axes. MCC demonstrated the ability to collect increasingly diverse and complex bodies with brains capable of ambulating each the required distance. A surprising result was that MCC produced controllers that were equivalent to those optimized by fitness-based search in terms of distance traveled. As a whole, these results demonstrate the generality of MCC as an algorithmic and architectural framework for the production and investigation of open-ended dynamics.

APPENDIX A: MAZE GENERATION ALGORITHM

This section provides detailed pseudocode for each stage of the enhanced maze generation process (described in chapter 5). Algorithm 4 outlines the top-level maze generation routine, while algorithm 5 describes the maze grid initialization steps. The solution path generation routine is shown in algorithm 6, which calls subroutines to lay down vertical and horizontal path segments and reroute the path in cases where those segments would create an overlap, each of which is shown in algorithms 7, 8, 9 and 10 respectively. Algorithm 11 outlines the top-level wall placement process, which calls algorithm 12 to enclose the solution path with walls around the outer boundaries of the sub-spaces. The process for subdividing the maze into solution path-induced quadrants is outlined in algorithm 13 while algorithm 14 determines the boundaries of such quadrants and algorithm 15 decides the placement of an opening from the solution path into each quadrant. Finally, algorithm 16 dictates the placement of walls within a given quadrant, iteratively bisecting the space until no additional walls can be placed.

Algorithm 4 Maze Generation

```

1: function GENERATEMAZE(mazeGenome)
2:   pathGenes  $\leftarrow$  mazeGenome.PathGenes
3:   wallGenes  $\leftarrow$  mazeGenome.WallGenes
4:   height  $\leftarrow$  mazeGenome.Height
5:   width  $\leftarrow$  mazeGenome.Width
6:   startLocation  $\leftarrow$  (0, 0) ▷ Initialize start location to top left
7:   endLocation  $\leftarrow$  (height - 1, width - 1) ▷ Initialize target location to bottom right
8:   grid  $\leftarrow$  InitializeGrid(height, width) ▷ Creates an empty grid for waypoints and walls
9:   GeneratePath(grid, pathGenes, startLocation, endLocation)
10:  GenerateWalls(grid, wallGenes)
11:  return Maze(grid)
12: end function

```

Algorithm 5 Maze Grid Initialization

```
1: function INITIALIZEGRID(height, width)
2:   grid  $\leftarrow$  Array[width, height]  $\triangleright$  Each cell in grid stores path and wall orientation (if any)
3:   for  $y = 1 : height$  do
4:     for  $x = 1 : width$  do
5:       grid[ $x, y$ ]  $\leftarrow$  None  $\triangleright$  Denotes an empty cell
6:     end for
7:   end for
8:   return grid
9: end function
```

Algorithm 6 Maze Path Generation

```
1: function GENERATEPATH(grid, pathGenes, startPoint, endPoint)
2:   pathIdx  $\leftarrow$  0
3:   while pathIdx  $\leq$  pathGenes.Count do
4:     if pathIdx = 0 then
5:       curPoint  $\leftarrow$  startPoint  $\triangleright$  First waypoint is accessed from maze start point
6:     else
7:       curPoint  $\leftarrow$  pathGenes[pathIdx - 1]  $\triangleright$  Previous waypoint is starting location
8:     end if
9:     if pathIdx = pathGenes.Count then
10:      tgtPoint  $\leftarrow$  endPoint  $\triangleright$  Last waypoint connects to maze end point
11:    else
12:      tgtPoint  $\leftarrow$  pathGenes[pathIdx]  $\triangleright$  Current waypoint is the target destination
13:    end if
14:    orientation  $\leftarrow$  tgtPoint.Orientation  $\triangleright$  Horizontal or vertical orientation
15:    grid[tgtPoint.X, tgtPoint.Y].IsWaypoint  $\leftarrow$  true
16:    if orientation = Horizontal then  $\triangleright$  Vertical followed by horizontal segment
17:      HorizontalPathReroute(grid, curPoint, tgtPoint)
18:      GenerateVerticalPathSegment(grid, curPoint, tgtPoint)
19:      GenerateHorizontalPathSegment(grid, curPoint, tgtPoint)
20:      if curPoint.X  $\neq$  tgtPoint.X & curPoint.Y  $\neq$  tgtPoint.Y then
21:        grid[curPoint.X, tgtPoint.Y].IsJuncture  $\leftarrow$  true
22:      end if
23:    else  $\triangleright$  Horizontal followed by vertical segment
24:      HandleVerticalOverlapCases(grid, curPoint, tgtPoint)
25:      GenerateHorizontalPathSegment(grid, curPoint, tgtPoint)
26:      GenerateVerticalPathSegment(grid, curPoint, tgtPoint)
27:      if curPoint.X  $\neq$  tgtPoint.X & curPoint.Y  $\neq$  tgtPoint.Y then
28:        grid[tgtPoint.X, curPoint.Y].IsJuncture  $\leftarrow$  true
29:      end if
30:    end if
31:  end while
32: end function
```

Algorithm 7 Vertical Path Segment Generation

```
1: function GENERATEVERTICALPATHSEGMENT(grid, curPoint, endPoint)
2:   if curPoint.Y ≤ endPoint.Y then
3:     while curPoint.Y ≤ endPoint.Y do
4:       grid[curPoint.X, curPoint.Y].PathDirection ← South
5:       curPoint.Y ← curPoint.Y + 1
6:     end while
7:   else
8:     while curPoint.Y ≥ endPoint.Y do
9:       grid[curPoint.X, curPoint.Y].PathDirection ← North
10:      curPoint.Y ← curPoint.Y − 1
11:    end while
12:  end if
13: end function
```

Algorithm 8 Horizontal Path Segment Generation

```
1: function GENERATEHORIZONTALPATHSEGMENT(grid, curPoint, endPoint)
2:   if curPoint.X ≤ endPoint.X then
3:     while curPoint.X ≤ endPoint.X do
4:       grid[curPoint.X, curPoint.Y].PathDirection ← East
5:       curPoint.X ← curPoint.X + 1
6:     end while
7:   else
8:     while curPoint.X ≥ endPoint.X do
9:       grid[curPoint.X, curPoint.Y].PathDirection ← West
10:      curPoint.X ← curPoint.X − 1
11:    end while
12:  end if
13: end function
```

Algorithm 9 Horizontal Path Reroute

```
1: function HORIZONTALPATHREROUTE(grid, curPoint, endPoint)
2:   if endPoint.Y < curPoint.Y & endPoint.X > curPoint.X then
3:     if grid[curPoint.X + 1, curPoint.Y].PathDirection ≠ None then
4:       grid[curPoint.X, curPoint.Y].PathDirection ← South
5:       grid[curPoint.X, curPoint.Y + 1].IsJuncture ← true
6:       curPoint.Y ← curPoint.Y + 1
7:     end if
8:     for curPoint.X : rightmostWaypoint.X do                                ▷ Avoids overlap
9:       grid[curPoint.X, curPoint.Y].PathDirection ← East
10:    end for
11:    grid[curPoint.X, curPoint.Y].IsJuncture ← true
12:  end if
13: end function
```

Algorithm 10 Vertical Path Reroute

```
1: function VERTICALPATHREROUTE(grid, curPoint, endPoint)
2:   if endPoint.X < curPoint.X & endPoint.Y > curPoint.Y then
3:     if grid[curPoint.X, curPoint.Y + 1].PathDirection ≠ None then
4:       grid[curPoint.X, curPoint.Y].PathDirection ← East
5:       grid[curPoint.X + 1, curPoint.Y].IsJuncture ← true
6:       curPoint.X ← curPoint.X + 1
7:     end if
8:     for curPoint.Y : lowestWaypoint.X do                                ▷ Avoids overlap
9:       grid[curPoint.X, curPoint.Y].PathDirection ← South
10:    end for
11:    grid[curPoint.X, curPoint.Y].IsJuncture ← true
12:  end if
13: end function
```

Algorithm 11 Maze Wall Generation

```
1: function GENERATEWALLS(grid, wallGenes)
2:   wallGeneIdx  $\leftarrow$  0
3:   loopIteration  $\leftarrow$  0
4:   EncloseAdjacentPathSegments(grid, mazeHeight, mazeWidth)  $\triangleright$  Encapsulate path
5:   partitions  $\leftarrow$  SubdivideMaze(grid, mazeHeight, mazeWidth)
6:   for all partition  $\in$  partitions do  $\triangleright$  Bisect each subdivision with wall genes
7:     partitionQueue  $\leftarrow$  Queue()
8:     if partition.SupportsWalls() = true then  $\triangleright$  Walls supported if > 1 unit
9:       loopIteration  $\leftarrow$  loopIteration + 1
10:      wallGeneIdx  $\leftarrow$  loopIteration % wallGenes.Count
11:      MarkPartitionBoundaries(grid, partition)
12:      InsertPartitionOpening(grid, partition, wallGenes[wallGeneIdx])
13:      partitionQueue.Enqueue(partition)
14:      while partitionQueue.Count > 0 do
15:        curPartition  $\leftarrow$  partitionQueue.Dequeue()
16:        loopIteration  $\leftarrow$  loopIteration + 1
17:        wallGeneIdx  $\leftarrow$  loopIteration % wallGenes.Count
18:        partitions  $\leftarrow$  SubdividePartition(grid, partition, wallGenes[wallGeneIdx])
19:        if partitions[0]  $\neq$  None then  $\triangleright$  Enqueue partition on top/left
20:          partitionQueue.Enqueue(partitions[0])
21:        end if
22:        if partitions[1]  $\neq$  None then  $\triangleright$  Enqueue partition on bottom/right
23:          partitionQueue.Enqueue(partitions[1])
24:        end if
25:      end while
26:    else
27:      MarkBoundaries(grid, partition)  $\triangleright$  Enclose partition and open onto path
28:    end if
29:  end for
30: end function
```

Algorithm 12 Path Encapsulation

```
1: function ENCLOSEADJACENTPATHSEGMENTS(grid, height, width)
2:   curCell  $\leftarrow$  grid[0, 0] ▷ Begin path traversal at start point
3:   while curCell.X < width & curCell.Y < height do
4:     pathOrientation  $\leftarrow$  curCell.PathOrientation
5:     pathDir  $\leftarrow$  curCell.PathDirection
6:     x  $\leftarrow$  curCell.X
7:     y  $\leftarrow$  curCell.Y
8:     if pathDir  $\neq$  North & grid[x, y - 1].PathDirection  $\neq$  None then
9:       grid[x, y - 1].SouthWall  $\leftarrow$  true ▷ Block adjacent path above
10:    end if
11:    if pathDir  $\neq$  South & grid[x, y + 1].PathDirection  $\neq$  None then
12:      grid[x, y].SouthWall  $\leftarrow$  true ▷ Block adjacent path below
13:    end if
14:    if pathDir  $\neq$  West & grid[x - 1, y].PathDirection  $\neq$  None then
15:      grid[x - 1, y].EastWall  $\leftarrow$  true ▷ Block adjacent path to the left
16:    end if
17:    if pathDir  $\neq$  East & grid[x + 1, y].PathDirection  $\neq$  None then
18:      grid[x, y].EastWall  $\leftarrow$  true ▷ Block adjacent path to the right
19:    end if
20:    if pathOrientation = Horizontal then
21:      if pathDir = West then
22:        curCell  $\leftarrow$  grid[x - 1, y] ▷ Next path cell is to the left
23:      else
24:        curCell  $\leftarrow$  grid[x + 1, y] ▷ Next path cell is to the right
25:      end if
26:    end if
27:    if pathOrientation = Vertical then
28:      if pathDir = North then
29:        curCell  $\leftarrow$  grid[x, y - 1] ▷ Next path cell is above
30:      else
31:        curCell  $\leftarrow$  grid[x, y + 1] ▷ Next path cell is below
32:      end if
33:    end if
34:  end while
35: end function
```

Algorithm 13 Maze Subdivision

```
1: function SUBDIVIDEMAZE(grid, height, width)
2:   subdivisions  $\leftarrow$  List() ▷ List of maze subdivisions
3:   for y=0:height do
4:     for x=0:width do
5:       if !grid[x, y].PathDir & IsInSubdiv(grid[x, y]) = false then
6:         startPnt  $\leftarrow$  (x, y)
7:         endPnt  $\leftarrow$  startPnt
8:         wallFound = false
9:         while endPnt.X < width & !grid[endPnt.X, endPnt.Y].PathDir do
10:          endPnt.X  $\leftarrow$  endPnt.X + 1 ▷ Locate right edge of subdivision
11:        end while
12:        while wallFound = false & endPnt.Y < height do ▷ Move to corner
13:          endPnt.Y  $\leftarrow$  endPnt.Y + 1
14:          curX  $\leftarrow$  startPnt.X
15:          while wallFound = false & curX < width do
16:            if grid[curX, endPnt.Y].PathDir  $\neq$  None then
17:              wallFound  $\leftarrow$  true
18:              endPnt.Y  $\leftarrow$  endPnt.Y - 1
19:            end if
20:            curX  $\leftarrow$  curX + 1
21:          end while
22:        end while
23:        subdivisions.Add(startPnt, endPnt)
24:      end if
25:    end for
26:  end for
27:  return subdivisions
28: end function
```

Algorithm 14 Partition Encapsulation

```
1: function MARKPARTITIONBOUNDARIES(grid, partition)
2:   for  $x = \text{partition}.X : \text{partition}.Width$  do
3:     if  $\text{partition}.Y > 0$  then
4:        $\text{grid}[x, \text{partition}.Y - 1].SouthWall \leftarrow true$   $\triangleright$  North boundary
5:     end if
6:      $\text{grid}[x, \text{partition}.Y + \text{partition}.Height - 1].SouthWall \leftarrow true$   $\triangleright$  South boundary
7:   end for
8:   for  $y = \text{partition}.Y : \text{partition}.Height$  do
9:     if  $\text{partition}.X > 0$  then
10:       $\text{grid}[\text{partition}.X - 1, y].EastWall \leftarrow true$   $\triangleright$  West boundary
11:    end if
12:     $\text{grid}[\text{partition}.x + \text{partition}.Width - 1, y].EastWall \leftarrow true$   $\triangleright$  East boundary
13:  end for
14:  if  $\text{partition}.Height = 1 | \text{partition}.Width = 1$  then  $\triangleright$  No room for walls, create opening
15:    if  $\text{partition}.X = 0$  then
16:       $\text{grid}[\text{partition}.X + \text{partition}.Width - 1, \text{partition}.Y].EastWall \leftarrow false$ 
17:    else
18:       $\text{grid}[\text{partition}.X - 1, \text{partition}.Y].EastWall \leftarrow false$ 
19:    end if
20:  end if
21: end function
```

Algorithm 15 Partition Opening Placement

```
1: function INSERTPARTITIONOPENING(grid, partition, wallGene)
2:   if  $\text{wallGene}.Orientation = Horizontal$  then
3:      $\text{wallLoc} \leftarrow (\text{partition}.X, \text{partition}.Y + \text{partition}.Height * \text{wallGene}.wallLoc)$ 
4:     if  $\text{partition}.X > 0$  then
5:        $\text{grid}[\text{partition}.X - 1, \text{wallLoc}.Y].EastWall \leftarrow false$   $\triangleright$  Opening on left
6:     else
7:        $\text{grid}[\text{partition}.Width - 1, \text{wallLoc}.Y].EastWall \leftarrow false$   $\triangleright$  Opening on right
8:     end if
9:   else
10:     $\text{wallLoc} \leftarrow (\text{partition}.X + \text{partition}.Width * \text{wallGene}.wallLoc, \text{partition}.Y)$ 
11:     $\text{passageLoc} \leftarrow (0, \text{wallLoc}.Y + \text{partition}.Height * \text{wallGene}.passageLoc)$ 
12:    if  $\text{partition}.X > 0$  then
13:       $\text{grid}[\text{partition}.X - 1, \text{passageLoc}.Y].EastWall \leftarrow false$   $\triangleright$  Left opening
14:    else
15:       $\text{grid}[\text{partition}.Width - 1, \text{passageLoc}.Y].EastWall \leftarrow false$   $\triangleright$  Right opening
16:    end if
17:  end if
18: end function
```

Algorithm 16 Partition Subdivision

```
1: function SUBDIVIDEPARTITION(grid, partition, wallGene)
2:   if wallGene.Orientation = Horizontal then
3:     wallLoc  $\leftarrow$  (partition.X, partition.Y + partition.Height * wallGene.wallLoc)
4:     passLoc  $\leftarrow$  (wallLoc.X + partition.Width * wallGene.passLoc, wallLoc.Y)
5:     wallLength  $\leftarrow$  partition.Width ▷ Wall spans full length of partition
6:     for position = 0 : wallLength do ▷ Mark horizontal wall and passage in maze grid
7:       if position  $\neq$  passLoc.X then
8:         grid[wallLoc.X + position, wallLoc.Y].SouthWall
9:       end if
10:    end for
11:    childPartition1.StartPoint  $\leftarrow$  (partition.X, partition.Y) ▷ Top partition
12:    childPartition1.Width  $\leftarrow$  partition.Width
13:    childPartition1.Height  $\leftarrow$  wallLoc.Y - partition.Y
14:    childPartition2.StartPoint  $\leftarrow$  (partition.X, wallLoc.Y + 1) ▷ Bottom partition
15:    childPartition2.Width  $\leftarrow$  partition.Width
16:    childPartition2.Height  $\leftarrow$  partition.Y + partition.Height - wallLoc.Y
17:  else
18:    wallLoc  $\leftarrow$  (partition.X + partition.Width * wallGene.wallLoc, partition.Y)
19:    passLoc  $\leftarrow$  (wallLoc.X, wallLoc.Y + partition.Height * wallGene.passLoc)
20:    wallLength  $\leftarrow$  partition.Height ▷ Wall spans full height of partition
21:    for position = 0 : wallLength do ▷ Mark vertical wall and passage in maze grid
22:      if position  $\neq$  passLoc.Y then
23:        grid[wallLoc.X, wallLoc.Y + position].EastWall
24:      end if
25:    end for
26:    childPartition1.StartPoint  $\leftarrow$  (partition.X, partition.Y) ▷ Left partition
27:    childPartition1.Width  $\leftarrow$  wallLoc.X - partition.X
28:    childPartition1.Height  $\leftarrow$  partition.Y
29:    childPartition2.StartPoint  $\leftarrow$  (wallLoc.X, partition.Y) ▷ Right partition
30:    childPartition2.Width  $\leftarrow$  partition.X + partition.Width - wallLoc.X
31:    childPartition2.Height  $\leftarrow$  partition.Height
32:  end if
33:  return < childPartition1, childPartition2 >
34: end function
```

APPENDIX B: EXPERIMENTAL PARAMETERS

All of the experiments in this dissertation utilize the NEAT algorithm. In the maze domain, NEAT evolves agent controllers, while in the body-brain coevolution domain, a variant of NEAT called *CPPN-NEAT* evolves CPPNs that generate material patterns in voxel bodies and weights for NNs that control active material deformation. Novelty search, which has an additional set of parameters, was used to bootstrap MCC with seed genomes in the maze domain as well as in the body-brain coevolution experiments. A description of each relevant NEAT parameter and novelty search parameter is provided below. Additionally, tables B.1 and B.2 enumerate the novelty search, NEAT and CPPN-NEAT parameter configurations for the maze navigation and body-brain coevolution experiments respectively (maze evolution parameters are exhaustively-detailed in chapter 5 sections 5.1.1 and 5.1.2).

NEAT parameter descriptions:

- **Population Size:** The number of NEAT genomes that compose the population.
- **Weight Mutation Probability:** The rate at which to apply mutations that modify a connection weight.
- **Add Connection Probability:** The rate at which to apply mutations that add a new connection.
- **Add Node Probability:** The rate at which to apply mutations that add a new node (neuron).
- **Delete Connection Probability:** The rate at which to apply mutations that remove an existing connection.

Novelty search parameter descriptions:

- **Behavior Characterization:** The behavioral trait used to characterize and compare individuals. For example, the trajectory end-point in a maze navigation domain could characterize

behavior.

- **Nearest Neighbors:** The number of individuals that are closest in behavior space against which to assess behavioral distance.
- **Archive Addition Threshold:** The minimum numeric novelty score that an individual must exceed to be entered into the novelty archive.
- **Max Batches with Archive Addition:** The maximum number of consecutive batches that can result in an archive addition without an increase in the archive addition threshold. Once this number is exceeded, the archive addition threshold is increased per the archive threshold increase multiplier.
- **Max Batches without Archive Addition:** The maximum number of consecutive batches that can be executed without an archive addition. After this number of batches has elapsed, the archive addition threshold is decreased per the archive threshold decrease multiplier.
- **Archive Threshold Increase Multiplier:** The proportion by which to increase the archive addition threshold.
- **Archive Threshold Decrease Multiplier:** The proportion by which to decrease the archive addition threshold.

Parameter	Bootstrap	MCC
Population Size	100	250
Weight Mutation Prob.	0.9	0.6
Add Connection Prob.	0.05	0.1
Add Node Prob.	0.005	0.01
Delete Connection Prob.	0.001	0.005
Behavior Characterization	End Point	N/A
Nearest Neighbors	15	N/A
Archive Addition Threshold	6	N/A
Max Batches w/Archive Addition	5	N/A
Max Batches wo/Archive Addition	10	N/A
Archive Threshold Inc. Multiplier	1.3	N/A
Archive Threshold Dec. Multiplier	0.95	N/A

Table B.1: **Maze navigation experiment parameter settings.** This table depicts the novelty search and NEAT parameter settings for the maze navigation experiments in chapters 4 and 5. Novelty search parameters apply only to the bootstrap.

Parameter	Bootstrap	MCC (Brains)	MCC (Bodies)
Population Size	100	150	30
Weight Mutation Prob.	0.9	0.6	0.6
Add Connection Prob.	0.05	0.1	0.1
Add Node Prob.	0.005	0.01	0.01
Delete Connection Prob.	0.001	0.005	0.005
Behavior Characterization	End Point	N/A	N/A
Nearest Neighbors	15	N/A	N/A
Archive Addition Threshold	6	N/A	N/A
Max Batches w/Archive Addition	5	N/A	N/A
Max Batches wo/Archive Addition	10	N/A	N/A
Archive Threshold Inc. Multiplier	1.3	N/A	N/A
Archive Threshold Dec. Multiplier	0.95	N/A	N/A

Table B.2: **Body-Brain coevolution experiment parameter settings.** This table depicts the novelty search and CPPN-NEAT parameter settings for the body-brain coevolution experiments in chapter 7. Novelty search parameters apply only to the bootstrap.

LIST OF REFERENCES

- [1] Christoph Adami, Charles Ofria, and Travis C. Collier. Evolution of biological complexity. *Proceedings of the National Academy of Sciences*, 97:4463–4468, 2000.
- [2] Sam Adams, Itmar Arel, Joscha Bach, Robert Coop, Rod Furlan, Ben Goertzel, J. Storrs Hall, Alexei Samsonovich, Matthias Scheutz, Matthew Schlesinger, et al. Mapping the landscape of human-level artificial general intelligence. *AI magazine*, 33(1):25–42, 2012.
- [3] Pere Alberch. Evolution of a developmental process: Irreversibility and redundancy in amphibian metamorphosis. In R. A. Raff and E. C. Raff, editors, *Development as an Evolutionary Process*, pages 23–40. Alan R. Liss, New York, 1987.
- [4] Mark A. Bedau. Artificial life: organization, adaptation and complexity from the bottom up. *Trends in cognitive sciences*, 7(11):505–512, 2003.
- [5] Jonathan I. Bloch and Doug M. Boyer. Grasping primate origins. *Science*, 298(5598): 1606–1610, 2002.
- [6] Margaret A. Boden. The philosophy of artificial life. *Minds Mach.*, 9(1):139–143, Feb 1999. ISSN 0924-6495. doi: 10.1023/A:1008373528631. URL <http://dx.doi.org/10.1023/A:1008373528631>.
- [7] Jonathan C. Brant and Kenneth O. Stanley. Minimal criterion coevolution: A new approach to open-ended search. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO ’17, page 67–74, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450349208. doi: 10.1145/3071178.3071186. URL <https://doi.org/10.1145/3071178.3071186>.

- [8] Jonathan C. Brant and Kenneth O. Stanley. Benchmarking open-endedness in minimal criterion coevolution. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '19, page 72–80, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450361118. doi: 10.1145/3321707.3321756. URL <https://doi.org/10.1145/3321707.3321756>.
- [9] Hans J. Bremermann. Optimization through evolution and recombination. *Self-organizing systems*, 93:106, 1962.
- [10] Hans J. Bremermann. Numerical optimization procedures derived from biological evolution processes. *Cybernetic problems in bionics*, pages 597–616, 1968.
- [11] J. J. Bull and E. L. Charnov. On irreversible evolution. *Evolution*, 39(5):1149–1155, 1985.
- [12] Larry Bull. On coevolutionary genetic algorithms. *Soft Computing*, 5(3):201–207, 2001.
- [13] Alastair D. Channon and Robert I. Damper. Towards the evolutionary emergence of increasingly complex advantageous behaviours. *International Journal of Systems Science*, 31(7):843–860, 2000.
- [14] Konstantinos Chatzilygeroudis, Antoine Cully, and Jean-Baptiste Mouret. Towards semi-episodic learning for robot damage recovery. *arXiv preprint arXiv:1610.01407*, 2016.
- [15] Nick Cheney, Robert MacCurdy, Jeff Clune, and Hod Lipson. Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2013)*, New York, NY, 2013. ACM Press.
- [16] Nick Cheney, Josh Bongard, and Hod Lipson. Evolving soft robots in tight spaces. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO '15, page 935–942, New York, NY, USA, 2015. Association for Computing Machinery.

- ing Machinery. ISBN 9781450334723. doi: 10.1145/2739480.2754662. URL <https://doi.org/10.1145/2739480.2754662>.
- [17] Nick Cheney, Josh Bongard, Vytas SunSpiral, and Hod Lipson. Scalable co-optimization of morphology and control in embodied machines. *Journal of The Royal Society Interface*, 15 (143):20170937, 2018.
 - [18] Jeff Clune. AI-GAs: AI-generating algorithms, an alternate paradigm for producing general artificial intelligence. *arXiv preprint arXiv:1905.10985*, 2019.
 - [19] Edoardo Conti, Vashisht Madhavan, Felipe P. Such, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems, NIPS’18*, pages 5032–5043, USA, 2018. Curran Associates Inc. URL <http://dl.acm.org/citation.cfm?id=3327345.3327410>.
 - [20] Tim F. Cooper and Charles Ofria. Evolution of stable ecosystems in populations of digital organisms. In *Proceedings of the Eighth International Conference on Artificial Life, ICAL 2003*, pages 227–232, Cambridge, MA, USA, 2003. MIT Press. ISBN 0-262-69281-3. URL <http://dl.acm.org/citation.cfm?id=860295.860334>.
 - [21] Francesco Corucci, Nick Cheney, Francesco Giorgio-Serchi, Josh Bongard, and Cecilia Laschi. Evolving soft locomotion in aquatic and terrestrial environments: Effects of material properties and environmental transitions. *Soft Robotics*, 5(4):475–495, 2018. doi: 10.1089/soro.2017.0055.
 - [22] Giuseppe Cuccu and Faustino Gomez. When novelty is not enough. In *European Conference on the Applications of Evolutionary Computation (EVOSTAR-2011)*, pages 234–243. Springer, 2011.

- [23] Antoine Cully and Jean-Baptiste Mouret. Behavioral repertoire learning in robotics. In *Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference*, GECCO '13, pages 175–182, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1963-8. doi: 10.1145/2463372.2463399. URL <http://doi.acm.org/10.1145/2463372.2463399>.
- [24] Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. Robots that can adapt like animals. *Nature*, 521:503–507, 2015. doi: 10.1038/nature14422.
- [25] Sylvain Cussat-Blanc, Jonathan Pascalie, Sébastien Mazac, Hervé Luga, and Yves Duthen. *A Synthesis of the Cell2Organ Developmental Model*, pages 353–381. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-33902-8. doi: 10.1007/978-3-642-33902-8_14. URL https://doi.org/10.1007/978-3-642-33902-8_14.
- [26] Sylvain Cussat-Blanc, Kyle Harrington, and Wolfgang Banzhaf. Artificial gene regulatory networks—a review. *Artificial Life*, 24(4):296–328, 2019. doi: 10.1162/artl_a_.00267. URL https://doi.org/10.1162/artl_a_00267. PMID: 30681915.
- [27] David B. D’Ambrosio and Kenneth O. Stanley. Scalable multiagent learning through indirect encoding of policy geometry. *Evolutionary Intelligence*, 6(1), 2013.
- [28] Paul Darwen and Xin Yao. Automatic modularization by speciation. In *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation*, pages 88–93. IEEE Computer Society Press, 1996.
- [29] Richard Dawkins. The evolution of evolvability. *On growth, form and computers*, pages 239–255, 2003.

- [30] Edwin D. De Jong. The incremental pareto-coevolution archive. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2004)*, Berlin, 2004. Springer Verlag. URL <http://www.cs.uu.nl/~dejong/publications/gecco04coev.pdf>.
- [31] Kalyanmoy Deb, Jeffrey Horn, and David E. Goldberg. Multimodal deceptive functions. *Complex Systems*, 7(2):131–154, 1993.
- [32] Antonio Della Cioppa, Angelo Marcelli, and Prisco Napoli. Speciation in evolutionary algorithms: Adaptive species discovery. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, GECCO '11, pages 1053–1060, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0557-0. doi: 10.1145/2001576.2001719. URL <http://doi.acm.org/10.1145/2001576.2001719>.
- [33] Frank Dellaert. Toward a biologically defensible model of development. Master's thesis, Case Western Reserve University, Cleveland, OH, 1995. URL <http://citeseer.nj.nec.com/dellaert95toward.html>.
- [34] P. Deloukas, G. D. Schuler, G. Gyapay, E. M. Beasley, C. Soderlund, P. Rodriguez-Tome, L. Hui, T. C. Matise, K. B. McKusick, J. S. Beckmann, S. Bentolila, M. Bihoreau, B. B. Birren, J. Browne, A. Butler, A. B. Castle, N. Chiannikulchai, C. Clee, P. J. Day, A. Dehejia, T. Dibling, N. Drouot, S. Duprat, C. Fizames, and D. R. Bentley. A physical map of 30,000 human genes. *Science*, 282(5389):744–746, October 23 1998.
- [35] Emily Dolson, Wolfgang Banzhaf, and Charles Ofria. Applying ecological principles to genetic programming. In Wolfgang Banzhaf, Randal S. Olson, William Tozier, and Rick Riolo, editors, *Genetic Programming Theory and Practice XV*, pages 73–88, Cham, 2018. Springer International Publishing. ISBN 978-3-319-90512-9.

- [36] Stephane Doncieux, Alban Laflaquière, and Alexandre Coninx. Novelty search: A theoretical perspective. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '19*, pages 99–106, New York, NY, USA, 2019. ACM. ISBN 978-1-4503-6111-8. doi: 10.1145/3321707.3321752. URL <http://doi.acm.org/10.1145/3321707.3321752>.
- [37] Miguel Duarte, Jorge Gomes, Sancho Moura Oliveira, and Anders L. Christensen. EvoRBC: evolutionary repertoire-based control for robots with arbitrary locomotion complexity. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, pages 93–100. ACM, 2016.
- [38] S.G. Ficici and J.B. Pollack. Challenges in coevolutionary learning: Arms-race dynamics, open-endedness, and mediocre stable states. *Artificial life VI*, page 238, 1998.
- [39] Dario Floreano, Peter Dürri, and Claudio Mattiussi. Neuroevolution: from architectures to learning. *Evolutionary Intelligence*, 1:47–62, 2008.
- [40] Lawrence J. Fogel. Artificial intelligence through a simulation of evolution. In *Proc. of the 2nd Cybernetics Science Symp., 1965*, 1965.
- [41] Lawrence J. Fogel, A. J. Owens, and M. J. Walsh. On the evolution of artificial intelligence. In *National Symposium on Human Factors in Electronics, 5th Edition, San Diego, California*, pages 63–76, 1964.
- [42] Martin Foltin. Automated maze generation and human interaction. *Brno: Masaryk University Faculty Of Informatics*, 2011.
- [43] Alex Fraser. Simulation of genetic systems by automatic digital computers. *Australian Journal of Biological Sciences*, 11(4):603–612, 1958.

- [44] Richard M. Friedberg. A learning machine: Part I. *IBM Journal of Research and Development*, 2(1):2–13, 1958.
- [45] Adam Gaier, Alexander Asteroth, and Jean-Baptiste Mouret. Aerodynamic design exploration through surrogate-assisted illumination. In *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, page 3330, 2017.
- [46] Adam Gaier, Alexander Asteroth, and Jean-Baptiste Mouret. Data-efficient exploration, optimization, and modeling of diverse designs through surrogate-assisted illumination. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 99–106. ACM, 2017.
- [47] Thomas Geijtenbeek, Michiel Van De Panne, and A. F. Van Der Stappen. Flexible muscle-based locomotion for bipedal creatures. *ACM Trans. Graph.*, 32(6):206:1–206:11, November 2013. ISSN 0730-0301. doi: 10.1145/2508363.2508399. URL <http://doi.acm.org/10.1145/2508363.2508399>.
- [48] Sherri Goings and Charles Ofria. Ecological approaches to diversity maintenance in evolutionary algorithms. In *2009 IEEE Symposium on Artificial Life*, pages 124–130, March 2009. doi: 10.1109/ALIFE.2009.4937703.
- [49] Sherri Goings, Heather Goldsby, Betty H.C. Cheng, and Charles Ofria. An ecology-based evolutionary algorithm to evolve solutions to complex problems. In *Artificial Life Conference Proceedings 2012*, pages 171–177. MIT Press, 07 2012. ISBN 9780262310505. doi: 10.7551/978-0-262-31050-5-ch024.
- [50] David E. Goldberg and Jon 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>.

- [51] Jorge Gomes, Paulo Urbano, and Anders L. Christensen. Progressive minimal criteria novelty search. In Juan Pavón, Néstor D. Duque-Méndez, and Rubén Fuentes-Fernández, editors, *Advances in Artificial Intelligence – IBERAMIA 2012*, pages 281–290, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-34654-5.
- [52] Jorge Gomes, Paulo Urbano, and Anders L. Christensen. Evolution of swarm robotics systems with novelty search. *Swarm Intelligence*, pages 1–30, 2013.
- [53] Jorge Gomes, Pedro Mariano, and Anders L. Christensen. Avoiding convergence in cooperative coevolution with novelty search. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems, AAMAS ’14*, pages 1149–1156, Richland, SC, 2014. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 978-1-4503-2738-1. URL <http://dl.acm.org/citation.cfm?id=2617388.2617428>.
- [54] Jorge Gomes, Pedro Mariano, and Anders L. Christensen. Novelty search in competitive coevolution. In *Parallel Problem Solving from Nature PPSN XIII*, pages 233–242. Springer, 2014.
- [55] Jorge Gomes, Pedro Mariano, and Anders L. Christensen. Cooperative coevolution of morphologically heterogeneous robots. *Artificial Life Conference Proceedings*, (27):312–319, 2015. doi: 10.1162/978-0-262-33027-5-ch059.
- [56] Stephen J. Gould. Dollo on dollo’s law: irreversibility and the status of evolutionary laws. *Journal of the History of Biology*, 3(2):189–212, 1970.
- [57] Robin Gras, Didier Devaurs, Adrianna Wozniak, and Adam Aspinall. An individual-based evolving predator-prey ecosystem simulation using a fuzzy cognitive map as the behavior model. *Artificial Life*, 2009. ISSN 10645462. doi: 10.1162/artl.2009.Gras.012.

- [58] Daniele Gravina, Antonios Liapis, and Georgios N. Yannakakis. Surprise search: Beyond objectives and novelty. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, GECCO '16, pages 677–684, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4206-3. doi: 10.1145/2908812.2908817. URL <http://doi.acm.org/10.1145/2908812.2908817>.
- [59] Daniele Gravina, Antonios Liapis, and Georgios N. Yannakakis. Surprise search for evolutionary divergence. *arXiv preprint arXiv:1706.02556*, 2017.
- [60] Daniele Gravina, Antonios Liapis, and Georgios N. Yannakakis. Coupling novelty and surprise for evolutionary divergence. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '17, pages 107–114, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4920-8. doi: 10.1145/3071178.3071179. URL <http://doi.acm.org/10.1145/3071178.3071179>.
- [61] Daniele Gravina, Antonios Liapis, and Georgios N. Yannakakis. Fusing novelty and surprise for evolving robot morphologies. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '18, pages 93–100, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5618-3. doi: 10.1145/3205455.3205503. URL <http://doi.acm.org/10.1145/3205455.3205503>.
- [62] Colin Green. SharpNEAT homepage. <http://sharpneat.sourceforge.net/>, 2003–2015.
- [63] Harry W. Greene. Natural history and evolutionary biology. *Predator-prey relationships: Perspectives and approaches from the study of lower vertebrates*, pages 99–108, 1986.
- [64] Inman Harvey, Ezequiel Di Paolo, Rachel Wood, Matt Quinn, and Elio Tuci. Evolutionary robotics: A new scientific tool for studying cognition. *Artificial life*, 11(1-2):79–98, 2005.

- [65] Jonathan Hiller and Hod Lipson. Dynamic simulation of soft multimaterial 3D-printed objects. *Soft Robotics*, 1(1):88–101, 2014. doi: 10.1089/soro.2013.0010.
- [66] John Holland. Adaptation in natural and artificial systems: an introductory analysis with application to biology. *Control and artificial intelligence*, 1975.
- [67] Gregory S. Hornby. ALPS: The age-layered population structure for reducing the problem of premature convergence. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, GECCO ’06, pages 815–822, New York, NY, USA, 2006. ACM. ISBN 1-59593-186-4. doi: 10.1145/1143997.1144142. URL <http://doi.acm.org/10.1145/1143997.1144142>.
- [68] Marcus Hutter and Shane Legg. Fitness uniform optimization. *IEEE Transactions on Evolutionary Computation*, 10(5):568–589, October 2006. ISSN 1089-778X. doi: 10.1109/TEVC.2005.863127. URL <http://dx.doi.org/10.1109/TEVC.2005.863127>.
- [69] Auke J. Ijspeert. Central pattern generators for locomotion control in animals and robots: A review. *Neural Networks*, 21(4):642 – 653, 2008. ISSN 0893-6080. doi: <https://doi.org/10.1016/j.neunet.2008.03.014>. URL <http://www.sciencedirect.com/science/article/pii/S0893608008000804>. Robotics and Neuroscience.
- [70] Nick Jakobi and Matthew Quinn. Some problems (and a few solutions) for open-ended evolutionary robotics. In *European Workshop on Evolutionary Robotics*, pages 108–122. Springer, 1998.
- [71] Sangbae Kim, Cecilia Laschi, and Barry Trimmer. Soft robotics: a bioinspired evolution in robotics. *Trends in biotechnology*, 31(5):287–294, 2013.
- [72] Steijn Kistemaker and Shimon Whiteson. Critical factors in the performance of novelty search. In *Proceedings of the 13th annual conference on Genetic and evolutionary com-*

- putation, 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>.
- [73] Aliona Kozlova, Joseph Alexander Brown, and Elizabeth Reading. Examination of representational expression in maze generation algorithms. In *Computational Intelligence and Games (CIG), 2015 IEEE Conference on*, pages 532–533. IEEE, 2015.
- [74] Sam Kriegman, Collin Cappelle, Francesco Corucci, Anton Bernatskiy, Nick Cheney, and Josh C. Bongard. Simulating the evolution of soft and rigid-body robots. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO '17*, pages 1117–1120, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4939-0. doi: 10.1145/3067695.3082051. URL <http://doi.acm.org/10.1145/3067695.3082051>.
- [75] Bakir Lacevic and Edoardo Amaldi. On population diversity measures in Euclidean space. In *IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE, 2010.
- [76] Joel Lehman and Kenneth O. Stanley. Exploiting open-endedness to solve problems through the search for novelty. In Seth Bullock, Jason Noble, Richard Watson, and Mark Bedau, editors, *Proceedings of the Eleventh International Conference on Artificial Life (Alife XI)*, Cambridge, MA, 2008. MIT Press. URL http://eplex.cs.ucf.edu/papers/lehman_alife08.pdf.
- [77] Joel Lehman and Kenneth 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. ISBN 978-1-4503-0072-8. doi: <http://doi.acm.org/10.1145/1830483.1830503>. URL <http://doi.acm.org/10.1145/1830483.1830503>.
- [78] Joel Lehman and Kenneth O. Stanley. Abandoning objectives: Evolution through the search

- for novelty alone. *Evolutionary Computation*, 19(2):189–223, 2011. URL http://www.mitpressjournals.org/doi/pdf/10.1162/EVCO_a_00025.
- [79] Joel Lehman and Kenneth O. Stanley. Evolving a diversity of virtual creatures through novelty search and local competition. In *GECCO '11: Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 211–218, Dublin, Ireland, 12–16 July 2011. ACM. ISBN 978-1-4503-0557-0. doi: doi:10.1145/2001576.2001606.
- [80] Joel Lehman and Kenneth O. Stanley. Novelty search and the problem with objectives. In *Genetic Programming Theory and Practice IX (GPTP 2011)*, New York, NY, 2011. Springer.
- [81] Joel Lehman and Kenneth O. Stanley. Beyond open-endedness: Quantifying impressiveness. In *Proceedings of the Thirteenth International Conference on Artificial Life (ALIFE XIII)*, Cambridge, MA, 2012. MIT Press.
- [82] Joel Lehman and Kenneth O. Stanley. Evolvability is inevitable: Increasing evolvability without the pressure to adapt. *PLoS ONE*, 8(4):e62186, 2013.
- [83] Joel Lehman, Kenneth O. Stanley, and Risto Miikkulainen. Effective diversity maintenance in deceptive domains. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pages 215–222. ACM, 2013.
- [84] Joel Lehman, Sebastian Risi, and Jeff Clune. Creative generation of 3d objects with deep learning and innovation engines. In *Proceedings of the International Conference on Computational Creativity (ICCC 2016)*, 2016.
- [85] Matthew A. Leibold. The niche concept revisited: Mechanistic models and community context. *Ecology*, 76(5):1371–1382, 1 1995. ISSN 0012-9658. doi: 10.2307/1938141.

- [86] Richard E Lenski, Charles Ofria, Robert T. Pennock, and Christoph Adami. The evolutionary origin of complex features. *Nature*, 423(6936):139–144, 2003.
- [87] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(1):1334–1373, January 2016. ISSN 1532-4435. URL <http://dl.acm.org/citation.cfm?id=2946645.2946684>.
- [88] Antonios Liapis, Héctor P. Martínez, Julian Togelius, and Georgios N. Yannakakis. Transforming exploratory creativity with DeLeNoX. In *Proceedings of the Fourth International Conference on Computational Creativity*, 2013.
- [89] Hod Lipson. Evolutionary robotics and open-ended design automation. In *Biomimetics*, pages 147–174. CRC Press, 2005.
- [90] Hod Lipson, Vytas Sunspirai, Josh Bongard, and Nicholas Cheney. On the difficulty of co-optimizing morphology and control in evolved virtual creatures. In *Proceedings of the European Conference on Artificial Life 13*, pages 226–233. MIT Press, 2016.
- [91] Michael Lynch. The frailty of adaptive hypotheses for the origins of organismal complexity. *Proceedings of the National Academy of Sciences*, 104(suppl 1):8597–8604, 2007.
- [92] Andrea Maesani, Pradeep R. Fernando, and Dario Floreano. Artificial evolution by viability rather than competition. *PloS one*, 9(1):e86831, 2014.
- [93] Andrea Maesani, Giovanni Iacca, and Dario Floreano. Memetic viability evolution for constrained optimization. *IEEE Transactions on Evolutionary Computation*, 20(1):125–144, 2016.
- [94] Samir W. Mahfoud. *Niching Methods for Genetic Algorithms*. PhD thesis, University of

- Illinois at Urbana-Champaign, Urbana, IL, May 1995. URL <http://citeseer.nj.nec.com/mahfoud95niching.html>.
- [95] Claudio Mattiussi and Dario Floreano. Viability Evolution: Elimination and Extinction in Evolutionary Computation. Technical report, EPFL, 2003.
- [96] Craig Mautner and Richard K. Belew. Evolving robot morphology and control. *Artificial Life and Robotics*, 4(3):130–136, Sep 2000. ISSN 1614-7456. doi: 10.1007/BF02481333.
- [97] Daniel W. McShea. Complexity and evolution: what everybody knows. *Biology and Philosophy*, 6(3):303–324, 1991.
- [98] Georgios Methenitis, Daniel Hennes, Dario Izzo, and Arnoud Visser. Novelty search for soft robotic space exploration. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO ’15, pages 193–200, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3472-3. doi: 10.1145/2739480.2754731. URL <http://doi.acm.org/10.1145/2739480.2754731>.
- [99] Elliot Meyerson, Joel Lehman, and Risto Miikkulainen. Learning behavior characterizations for novelty search. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, GECCO ’16, pages 149–156, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4206-3. doi: 10.1145/2908812.2908929. URL <http://doi.acm.org/10.1145/2908812.2908929>.
- [100] Thomas Miconi. Evosphere: evolutionary dynamics in a population of fighting virtual creatures. In *IEEE Congress on Evolutionary Computation*, pages 3066–3073. IEEE, 2008.
- [101] Laurent Mignonneau and Christa Sommerer. Creating artificial life for interactive art and entertainment. *Leonardo*, 34(4):303–307, 2001.

- [102] Nick Moran and Jordan Pollack. Evolving complexity in cooperative and competitive noisy prediction games. *Artificial Life*, 25(4):366–382, 2019. doi: 10.1162/artl_a_00302. URL https://doi.org/10.1162/artl_a_00302. PMID: 31697585.
- [103] Gregory Morse, Sebastian Risi, Charles R. Snyder, and Kenneth O. Stanley. Single-unit pattern generators for quadruped locomotion. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2013)*, New York, NY, USA, 2013. ACM.
- [104] Gregory Morse, L. B. Soros, and Kenneth O. Stanley. Additional stability for single-unit pattern generators. In *Workshop on Nature-inspired Techniques for Robotics at the 13th International Conference on Parallel Problem Solving from Nature (PPSN 2014)*, 2014.
- [105] Jean-Baptiste Mouret. Novelty-based multiobjectivization. In *New Horizons in Evolutionary Robotics*, pages 139–154. Springer, Berlin Heidelberg, 2011.
- [106] Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *ArXiv e-prints*, abs/1504.04909, 2015. URL <http://arxiv.org/abs/1504.04909>.
- [107] Jean-Baptiste Mouret and Stéphane Doncieux. Encouraging behavioral diversity in evolutionary robotics: An empirical study. *Evolutionary computation*, 20(1):91–133, 2012.
- [108] Anh Nguyen, Jason Yosinski, and Jeff 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.
- [109] Stefano Nolfi and Dario Floreano. Coevolving predator and prey robots: do “arms races” arise in artificial evolution? *Artificial life*, 4(4):311–335, 1998.
- [110] Stefano Nolfi, Josh Bongard, Phil Husbands, and Dario Floreano. Evolutionary robotics. In *Springer Handbook of Robotics*, pages 2035–2068. Springer, 2016.

- [111] Jørgen Nordmoen, Kai O. Ellefsen, and Kyrre Glette. Combining MAP-elites and incremental evolution to generate gaits for a mammalian quadruped robot. In *International Conference on the Applications of Evolutionary Computation*, pages 719–733. Springer, 2018.
- [112] Tønnes F. Nygaard, Charles P. Martin, Eivind Samuelsen, Jim Torresen, and Kyrre Glette. Real-world evolution adapts robot morphology and control to hardware limitations. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '18, page 125–132, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450356183. doi: 10.1145/3205455.3205567. URL <https://doi.org/10.1145/3205455.3205567>.
- [113] Charles Ofria and Claus O. Wilke. Avida: A software platform for research in computational evolutionary biology. *Artificial life*, 10(2):191–229, 2004.
- [114] Matthew R. Orr and Thomas B. Smith. Ecology and speciation. *Trends in Ecology & Evolution*, 13(12):502–506, 1998.
- [115] Steven L. Peck. Simulation as experiment: a philosophical reassessment for biological modeling. *Trends in Ecology & Evolution*, 19(10):530–534, 2004.
- [116] Rolf Pfeifer and Gabriel Gómez. Morphological computation—connecting brain, body, and environment. In *Creating brain-like intelligence*, pages 66–83. Springer, 2009.
- [117] Elena Popovici, Anthony Bucci, R. Paul Wiegand, and Edwin D. De Jong. *Coevolutionary Principles*, pages 987–1033. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-540-92910-9. doi: 10.1007/978-3-540-92910-9_31. URL http://dx.doi.org/10.1007/978-3-540-92910-9_31.
- [118] Mitchell A. Potter and Kenneth A. De Jong. Evolving neural networks with collaborative

- species. In *Proceedings of the 1995 Summer Computer Simulation Conference*, pages 340–345, 1995.
- [119] Mitchell A. Potter and Kenneth A. De Jong. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary computation*, 8(1):1–29, 2000.
- [120] Emmanuelle Pouydebat, Michel Laurin, Philippe Gorce, and Vincent Bels. Evolution of grasping among anthropoids. *Journal of evolutionary biology*, 21(6):1732–1743, 2008.
- [121] Joshua Powers, Sam Kriegman, and Josh Bongard. The effects of morphology and fitness on catastrophic interference. *Artificial Life*, pages 606–613, 2018. doi: 10.1162/isal_a_00111.
- [122] Jane Prophet. Sublime ecologies and artistic endeavors: Artificial life and interactivity in the online project technosphere. *Leonardo*, 29(5):339–344, 1996.
- [123] Justin K. Pugh, L. B. Soros, and Kenneth O. Stanley. Quality Diversity: A New Frontier for Evolutionary Computation. *Frontiers in Robotics and AI*, 3(40), 2016. ISSN 2296-9144. URL http://www.frontiersin.org/evolutionary_robotics/10.3389/frobt.2016.00040/abstract.
- [124] Thomas S. Ray. An approach to the synthesis of life. In *Artificial Life II*, pages 371–408. Addison-Wesley, 1991.
- [125] Ingo Rechenberg. Evolutionsstrategien. In *Simulationsmethoden in der Medizin und Biologie*, pages 83–114. Springer, 1978.
- [126] Torsten Reil and Phil Husbands. Evolution of central pattern generators for bipedal walking in a real-time physics environment. *IEEE Transactions on Evolutionary Computation*, 6(2):159–168, April 2002.

- [127] Daniel Richards and Martyn Amos. Evolving morphologies with CPPN-NEAT and a dynamic substrate. In *ALIFE 14: The Fourteenth Conference on the Synthesis and Simulation of Living Systems*, volume 14, pages 255–262, 2014.
- [128] John Rieffel, Davis Knox, Schuyler Smith, and Barry Trimmer. Growing and evolving soft robots. *Artificial Life*, 20(1):143–162, 2014. doi: 10.1162/ARTL_a_00101.
- [129] Sebastian Risi and Kenneth 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.
- [130] Sebastian Risi, Sandy D. Vanderbleek, Charles E. Hughes, and Kenneth 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.
- [131] Christopher D. Rosin and Richard K. Belew. New methods for competitive coevolution. *Evolutionary computation*, 5(1):1–29, 1997.
- [132] Günter Rudolph. Convergence of evolutionary algorithms in general search spaces. In *Proceedings of IEEE International Conference on Evolutionary Computation*, pages 50–54, 1996. doi: 10.1109/ICEC.1996.542332.
- [133] Kepa Ruiz-Mirazo, Juli Peretó, and Alvaro Moreno. A universal definition of life: autonomy and open-ended evolution. *Origins of Life and Evolution of the Biosphere*, 34(3):323–346, 2004.
- [134] Howard D. Rundle and Patrik Nosil. Ecological speciation. *Ecology letters*, 8(3):336–352, 2005.

- [135] Eivind Samuelsen and Kyrre Glette. Some distance measures for morphological diversification in generative evolutionary robotics. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2014)*, pages 721–728. ACM, 2014.
- [136] Peter T. Saunders and Mae-Wan Ho. On the increase in complexity in evolution. *Journal of Theoretical Biology*, 63(2):375–384, 1976.
- [137] Dolph Schluter. Ecological causes of adaptive radiation. *The American Naturalist*, 148: S40–S64, 11 1996. doi: 10.1086/285901.
- [138] Dolph Schluter. Ecology and the origin of species. *Trends in Ecology & Evolution*, 16: 372–380, 08 2001. doi: 10.1016/S0169-5347(01)02198-X.
- [139] Michael D. Schmidt and Hod Lipson. Age-fitness pareto optimization. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, GECCO ’10*, pages 543–544, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0072-8. doi: 10.1145/1830483.1830584. URL <http://doi.acm.org/10.1145/1830483.1830584>.
- [140] H. P. Schwefel. Kybernetische evolution als strategie der experimentellen forschung in der stromungstechnik. *Diploma thesis, Technical Univ. of Berlin*, 1965.
- [141] Jimmy Secretan, Nicholas Beato, David B. D’Ambrosio, Adelein Rodriguez, Adam Campbell, Jeremiah T. Folsom-Kovarik, and Kenneth O. Stanley. Picbreeder: A case study in collaborative evolutionary exploration of design space. *Evolutionary Computation*, 19(3):345–371, 2011. URL http://www.mitpressjournals.org/doi/pdf/10.1162/EVCO_a_00030.
- [142] Chad W. Seys and Randall D. Beer. Evolving walking: The anatomy of an evolutionary search. In S. Schaal, A. Ijspeert, A. Billard, S. Vijayakumar, J. Hallam, and J.-A. Meyer,

- editors, *From Animals to Animats 8: Proceedings of the Eight International Conference on Simulation of Adaptive Behavior*, pages 357–363. MIT Press, 2004.
- [143] Karl Sims. Evolving 3D morphology and behavior by competition. pages 28–39. MIT Press, Cambridge, MA, 1994. URL <http://www.mpi-sb.mpg.de/services/library/proceedings/contents/alife94.html>.
- [144] L.B. Soros and Kenneth O Stanley. Identifying necessary conditions for open-ended evolution through the artificial life world of chromaria. In *ALIFE 14: The Fourteenth International Conference on the Synthesis and Simulation of Living Systems*, pages 793–800, 2014.
- [145] L.B. Soros, Nick Cheney, and Kenneth O Stanley. How the strictness of the minimal criterion impacts open-ended evolution. In *ALIFE 15: The Fifteenth International Conference on the Synthesis and Simulation of Living Systems*, pages 208–215, 2016.
- [146] Russell K Standish. Open-ended artificial evolution. *International Journal of Computational Intelligence and Applications*, 3(02):167–175, 2003.
- [147] Kenneth O. Stanley. Comparing artificial phenotypes with natural biological patterns. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) Workshop Program*, New York, NY, 2006. ACM Press. URL http://http://eplex.cs.ucf.edu/index.php?option=com_content&task=view&id=14&Itemid=28#stanley.geccows06.
- [148] Kenneth 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.
- [149] Kenneth O. Stanley. Neuroevolution: A different kind of deep

- learning, 2017. URL <https://www.oreilly.com/ideas/neuroevolution-a-different-kind-of-deep-learning>.
- [150] Kenneth O. Stanley. Answering the call of open-endedness. In *Artificial Life Conference Proceedings*, pages 7–7. MIT Press, 2018.
- [151] Kenneth O. Stanley. Why open-endedness matters. *Artificial Life*, 2019. ISSN 15309185. doi: 10.1162/artl_a_00294.
- [152] Kenneth O. Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10:99–127, 2002. URL `stanley:ec02`.
- [153] Kenneth O. Stanley and Risto Miikkulainen. A taxonomy for artificial embryogeny. *Artificial Life*, 9(2):93–130, 2003. URL <http://nn.cs.utexas.edu/keyword?stanley:alife03>.
- [154] Kenneth O. Stanley and Risto Miikkulainen. Competitive coevolution through evolutionary complexification. *Journal of Artificial Intelligence Research (JAIR)*, 21:63–100, 2004. URL `stanley:jair04`.
- [155] Kenneth O. Stanley, Joel Lehman, and L. B. Soros. Open-endedness: The last grand challenge you’ve never heard of, 2017. URL <https://www.oreilly.com/ideas/open-endedness-the-last-grand-challenge-youve-never-heard-of>.
- [156] Christopher Stanton and Jeff Clune. Curiosity search: producing generalists by encouraging individuals to continually explore and acquire skills throughout their lifetime. *PloS one*, 11(9):e0162235, 2016.
- [157] Christopher Stanton and Jeff Clune. Deep Curiosity Search: Intra-Life Exploration Improves Performance on Challenging Deep Reinforcement Learning Problems. 2018. URL <http://arxiv.org/abs/1806.00553>.

- [158] Paul Szerlip and Kenneth 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.
- [159] Paul A. Szerlip, Gregory Morse, Justin K. Pugh, and Kenneth 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.
- [160] Jacopo Talamini, Eric Medvet, Alberto Bartoli, and Andrea De Lorenzo. Evolutionary synthesis of sensing controllers for voxel-based soft robots. *Artificial Life Conference Proceedings*, (31):574–581, 2019. doi: 10.1162/isal_a_00223.
- [161] Charles Taylor and David Jefferson. Artificial life as a tool for biological inquiry. *Artificial Life*, 1(1_2):1–13, 1993.
- [162] Tim Taylor, Mark Bedau, Alastair Channon, David Ackley, Wolfgang Banzhaf, Guillaume Beslon, Emily Dolson, Tom Froese, Simon Hickinbotham, Takashi Ikegami, Barry McMullin, Norman Packard, Steen Rasmussen, Nathaniel Virgo, Eran Agmon, Edward Clark, Simon McGregor, Charles Ofria, Glen Ropella, Lee Spector, Kenneth O. Stanley, Adam Stanton, Christopher Timperley, Anya Vostinar, and Michael Wiser. Open-ended evolution: Perspectives from the OEE workshop in york. *Artificial Life*, 22(3):408–423, 2016. doi: 10.1162/ARTL_a_00210. URL https://doi.org/10.1162/ARTL_a_00210. PMID: 27472417.
- [163] David Tilman. Resource competition and community structure. *Monographs in population biology*, 17:1–296, 1982.
- [164] David Tilman. Causes, consequences and ethics of biodiversity. *Nature*, 405:208–11, 06 2000. doi: 10.1038/35012217.

- [165] Julian Togelius, Georgios Yannakakis, Kenneth O. Stanley, and Cameron Browne. Search-based procedural content generation. In *Proceedings of the 2nd European event on Bio-inspired Algorithms in Games (EvoGAMES 2010)*, New York: NY, 2010. Springer.
- [166] Alan Turing. The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society B*, 237:37–72, 1952.
- [167] Sara Via. Natural selection in action during speciation. *Proceedings of the National Academy of Sciences*, 106(Supplement 1):9939–9946, 2009.
- [168] Darrell Whitley and Andrew M. Sutton. *Genetic Algorithms — A Survey of Models and Methods*, pages 637–671. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-540-92910-9. doi: 10.1007/978-3-540-92910-9_21. URL https://doi.org/10.1007/978-3-540-92910-9_21.
- [169] R. Paul Wiegand. *An Analysis of Cooperative Coevolutionary Algorithms*. PhD thesis, George Mason University, Fairfax, VA, USA, 2004. AAI3108645.
- [170] R. Paul Wiegand, William C. Liles, and Kenneth A. De Jong. An empirical analysis of collaboration methods in cooperative coevolutionary algorithms. In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation, GECCO’01*, pages 1235–1242, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1-55860-774-9. URL <http://dl.acm.org/citation.cfm?id=2955239.2955458>.
- [171] Lewis Wolpert, Cheryll Tickle, and Alfonso M. Arias. *Principles of development*. Oxford University Press, USA, 2015.
- [172] Larry Yaeger. Polyworld: Life in a new context. *Artificial Life III*, pages 263–298, 1993.
- [173] Xin Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, 1999.

- [174] Or Yogev and Erik K. Antonsson. Growth and development of continuous structures. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, GECCO '07*, pages 1064–1065, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-697-4. doi: 10.1145/1276958.1277169. URL <http://doi.acm.org/10.1145/1276958.1277169>.
- [175] Or Yogev, Andrew A. Shapiro, and Erik K. Antonsson. Computational evolutionary embryogeny. *IEEE Transactions on Evolutionary Computation*, 14(2):301–325, April 2010. ISSN 1089-778X. doi: 10.1109/TEVC.2009.2030438. URL <http://dx.doi.org/10.1109/TEVC.2009.2030438>.
- [176] Jinhong Zhang, Rasmus Tarnby, Antonios Liapis, and Sebastian Risi. Drawcompileevolve: Sparking interactive evolutionary art with human creations. In Colin Johnson, Adrian Carballal, and João Correia, editors, *Evolutionary and Biologically Inspired Music, Sound, Art and Design*, pages 261–273, Cham, 2015. Springer International Publishing. ISBN 978-3-319-16498-4.
- [177] Michael J. Zigmond, Floyd E. Bloom, Story C. Landis, James L. Roberts, and Larry R. Squire, editors. *Fundamental Neuroscience*. Academic Press, London, 1999.