2015

Worldwide Infrastructure for Neuroevolution: A Modular Library to Turn Any Evolutionary Domain into an Online Interactive Platform

Paul Szerlip
University of Central Florida

WORLDWIDE INFRASTRUCTURE FOR NEUROEVOLUTION: A MODULAR LIBRARY TO TURN ANY EVOLUTIONARY DOMAIN INTO AN ONLINE INTERACTIVE PLATFORM

by

PAUL A. SZERLIP
B.S. Tufts University, 2010
M.S. University of Central Florida, 2013

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

Summer Term
2015

Major Professor: Kenneth O. Stanley

ABSTRACT

Across many scientific disciplines, there has emerged an open opportunity to utilize the scale and reach of the Internet to collect scientific contributions from scientists and non-scientists alike. This process, called *citizen science*, has already shown great promise in the fields of biology and astronomy. Within the fields of artificial life (ALife) and evolutionary computation (EC) experiments in *collaborative interactive evolution* (CIE) have demonstrated the ability to collect thousands of experimental contributions from hundreds of users across the glob. However, such collaborative evolutionary systems can take nearly a year to build with a small team of researchers. This dissertation introduces a new developer framework enabling researchers to easily build fully persistent online collaborative experiments around almost any evolutionary domain, thereby reducing the time to create such systems to weeks for a single researcher. To add collaborative functionality to any potential domain, this framework, called Worldwide Infrastructure for Neuroevolution (WIN), exploits an important unifying principle among all evolutionary algorithms: regardless of the overall methods and parameters of the evolutionary experiment, every individual created has an explicit parent-child relationship, wherein one individual is considered the direct descendant of another. This principle alone is enough to capture and preserve the relationships and results for a wide variety of evolutionary experiments, while allowing multiple human users to meaningfully contribute. The WIN framework is first validated through two experimental domains, image evolution and a new two-dimensional virtual creature domain, Indirectly Encoded SodaRace (IESoR), that is shown to produce a visually diverse variety of ambulatory creatures. Finally, an Android application built with WIN, #filters, allows users to interactively evolve custom image effects to apply to personalized photographs, thereby introducing the first CIE application available for any mobile device. Together, these collaborative experiments and new mobile application establish a comprehensive new platform for evolutionary computation that can change how researchers design

and conduct citizen science online.

To my wonderful family and friends.

ACKNOWLEDGMENTS

I would like to thank my advisor Dr. Kenneth O. Stanley for his support through the writing of this dissertation. I consider myself fortunate to have had the opportunity to contribute to the excellent work being conducted at his research lab, the Evolutionary Complexity Research Group (EPlex). Dr. Stanley's formidable intellectual capacity and his compassion and concern for the students whom he advises has profoundly impacted my approach to problem solving, both academic or otherwise. The work in this dissertation is the most personally rewarding endeavor I've ever undertaken, and it could not have happened without Dr. Stanley. Thank you.

I would also like to thank my committee members for their guidance in focusing the questions and content of my research. Professor Annie Wu, Professor Joe LaViola, and Professor Joo Kim, your comments helped shape my intellectual work into a more robust dissertation. Moreover, I had the good fortune to work with Professor Joe LaViola on an independent study project early in my PhD that would form the underlying MaestroGenesis software discussed later in this manuscript.

Thanks to my good friend and colleague, Dr. Amy K. Hoover. On the very first day of classes before I knew anyone in Orlando, Amy invited me to join her and the rest of the EPlex members for a drink. It was a small gesture that speaks volumes about Amy's deep concern for the well-being of those around her. I am proud to call her my friend. Our work on the MaestroGenesis project helped me discover the type of software engineering I am passionate about, and I'll never forget our experiences attending conferences across the world.

The work in this dissertation has been supported with a UCF Hillman Award and NSF CreativeIT grant no. IIS-1002507. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

Finally, I'd like to thank my parents, Suzanne Barlow and Burt Szerlip.

Since as long as I can remember, my mother has always encouraged me to pursue my intellectual interests, wherever they may take me. I proudly consider myself a lifelong learner, and it is no doubt a direct result of my mother's passion for education. Thank you mom.

It is hard to put in words how fortunate I am to receive the unconditional support of my father, Burt. Without him, I would not be where I am. He is a good man, and a good father. Thank you dad.

Without the love and support of my parents, I could not have finished this dissertation. It is not a stretch to say that I have lived a privileged life, and I have both of your to thank for everything you've done for me.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

The fields of evolutionary computation (EC) and artificial life (ALife) are inspired by the products of natural evolution. Researchers in these fields hope to reproduce or one day exceed natural evolution's prodigious discoveries. Many experiments have focused on replicating specific influential aspects of natural evolution, e.g. *open-ended evolution* [8, 60] or evolving a diverse collection of virtual morphologies [2, 47, 56, 76, 83]. Excitingly, a new frontier has opened up at the intersection of ALife and the Internet, wherein researchers are no longer constrained by running experiments on a single computer [88].

Notably, with the abundance of cheap computational resources now available through cloud computing services, researchers can even provide a shared community platform for conducting hundreds of asynchronous ALife simulations, as demonstrated by the ALife Zoo [39]. Such a platform reflects recent community efforts towards "archiving, sharing, reproduction, and reuse of scientific experiments and platforms, for collaborative open science [88]."

As in other disciplines, to achieve this goal of collaborative open science, one promising research direction for evolutionary computation is *citizen science*, i.e. the act of aggregating scientific results from amateur or non-professional participants [74]. In fact, the idea that casual human users can aid serious scientific endeavors has gained credibility in recent years with a number of online citizen science projects in which users who often are not scientific experts are crowd-sourced to yield results that in some cases would be impossible to achieve in another way [20, 22, 74].

One particularly successful example is Fold It, a protein-folding game where players accurately predicted folded protein structures that even led to scientific publications crediting the 57,000 users [20]. Others include Phylo, an experimental game where players try to align nucleotide sequences [52], and Galaxy Zoo, where players help to classify the morphologies of large numbers

of galaxies [58]. Such citizen science projects often present scientific challenges to the user in a relatable "gamified" mechanic allowing non-scientists to contribute and even experience reciprocal educational benefits [40].

Interestingly, the evolutionary computation community already has an effective research tool for crowd-sourcing contributions from laypeople: interactive evolutionary computation (IEC; 24). In interactive evolution, the human user is repeatedly presented with a collection of potential choices within the given scientific domain and asked to iteratively select the choice that most appeals to him or her. In a similar way that human-driven dog breeding only requires aesthetic selections, experiments built with interactive evolution allow users to make scientific contributions through the simple act of iteratively choosing options that appeal to their own aesthetic or objective interests.

Previously, researchers have revealed the potential for IEC to aid in large-scale citizen science projects, successfully combining the power of IEC with the extensive reach of the Internet in a process called *collaborative interactive evolution* (CIE; 86). In such CIE experiments, potentially thousands of users from across the globe can implicitly contribute and collaborate through interactive evolution [11, 69, 72, 73, 86]. Note that such CIE experiments resonate with a strong community interest in simultaneously disseminating scientific results while promoting "public engagement and participation with A-Life research [88]."

Beyond the community goal of public engagement in science, crowd-sourcing human intuition has the added benefit of potentially improving algorithmic performance. Evidence in evolutionary robotics has already shown that amateur users operating a robot simulator can be harnessed for crowd-sourcing intricate robot controllers Bongard [6]. Similarly, recent experiments suggest that individual humans are capable of interleaving their own intuition with automated algorithms, yielding better results than the automated algorithms can alone [4, 92].

Demonstrating the scale of CIE experiments, one such application, Picbreeder (a genetic art pro-

gram for breeding pictures), empowers users to discover and collect a wide variety of images in a single browsable archive available at `http://picbreeder.org` [73]. Furthermore, the contributions of Picbreeder extend beyond scientific research as the largest repository of evolved content available online, collected from hundreds of users producing over 10,000 images in the last seven years. Most importantly, while the algorithms responsible for generating and modifying the images inside Picbreeder originate from artificial intelligence research [79, 82], the intuitive breeding process of interactive evolution frees users from the need for any scientific training or understanding to meaningfully extend existing results.

Fundamentally, Picbreeder is an evolutionary experiment that efficiently combines the efforts of multiple users, both academics and non-academic alike, to exploit the immense scale of the Internet to discover interesting solutions within a massive and complex search space. In fact, results generated within Picbreeder, a genetic art program, contributed to the creation of several new artificial intelligence algorithms including HyperNEAT and Novelty Search [32, 56] that have demonstrated strong performance in dozens of experimental domains [1, 3, 7, 12, 13, 15, 16, 27, 28, 32, 36, 50, 51, 83, 89].

Unfortunately, while CIE could be a significant source of scientific progress for the community, the process by which EC researchers can build and execute a system capable of conducting IEC across many simultaneous users is extremely resource intensive. That is, the construction of new CIE applications is dramatically limited by excessive resource and time investments. Even with a well-studied domain like genetic art, Picbreeder took a small team of six researchers over a year to construct [73].

Though web hosting technology has progressed in the years since Picbreeder was built, it would still take considerable effort to construct the same level of collaboration for any given experiment. Worse, due to a lack of community tools and resources, researchers are required to repeatedly

construct the same basic software structure for each CIE application. For example. at the time of writing, there are no open-source CIE experiments available. Thus, there is a significant untapped opportunity to crowd-source scientific results across a variety of evolutionary domains.

To unlock the potential of such evolutionary systems and enable an entirely new way to conduct research in EC, this dissertation introduces a framework for the EC and ALife community called Worldwide Infrastructure for Neuroevolution (WIN). WIN is a developer framework that significantly reduces the complexity of creating fully persistent, online, and interactive or automated evolutionary platforms around any evolutionary domain. Established from a body of published work [45, 83, 84], WIN enables researchers to create or effectively extend experimental domains with collaborative multi-user features even if the domain was not originally designed to be persistent or part of a CIE application.

The overarching **hypothesis** of this work is that the WIN framework is an effective platform-building tool for aiding evolutionary researchers in creating large-scale CIE experiments across a wide range of evolutionary domains. The hypothesis is supported through several major contributions.

First (1), the WIN framework emerged from extending work on an evolutionary music program I co-created with Amy K. Hoover called MaestroGenesis (`http://maestrogenesis.org`; 43, 44, 45, 46). Through the IEC framework I co-built in MaestroGenesis, the software enables novice musicians with little or no musical expertise to generate and discover musical voices to accompany existing musical compositions. Results from human studies conducted with MaestroGenesis validate that the IEC framework helps hide the domain complexity from non-scientific users, while the software represents a substantial experimental basis upon which the more ambitious WIN framework is built.

Though the domain of music generation is both creative and expressive, a new domain can help to

extend the ideas first explored in MaestroGenesis to a wider and more objective investigation of search in general. To aid this inquiry, a new artificial life domain was built to search and discover an extensive range of ambulating two-dimensional virtual creatures built from simples masses and springs, called sodaracers [62]. This new domain (2), named Indirectly Encoded SodaRace (IESoR), is both lightweight to simulate and can consistently produce distinct virtual creatures with visually diverse ambulation strategies [83]. In contrast to the creative domain of music, IESoR represents an experimental domain with measurable scientific objectives (distance traveled) based on a virtual world in which thousands of users previously showed interest through the Sodarace constructor [62].

Thus IESoR represents a viable candidate for a new collaborative interactive evolution experiment despite the domain not originally being constructed to handle collaboration. Through a prototype WIN framework (3), two CIE experiments built with WIN, win-Picbreeder and win-IESoR, simultaneously demonstrate the ability for the WIN library to replicate the functionality of existing CIE applications like Picbreeder and augment more objective-driven evolutionary domains like IESoR with collaborative features. The results show that the WIN framework can even handle interleaving human selections with automated evolutionary algorithms, a technique previously constrained to a single user on a single computer [92]. Such experimental results validate that WIN is a functional platform-building tool for multiple evolutionary domains.

To support the claim that WIN can help proliferate CIE experiments, a developer survey (4) was conducted in which participants were given the existing win-Picbreeder source code and asked to create a variant service. In support of the hypothesis, all survey participants were capable of creating win-Picbreeder experiments with small variations (now available online at `http://winark.org/variants`) in about four hours or less. In effect, each CIE experiment built with WIN and open-sourced to the community serves as a new building block for an entire collection of CIE applications.

To further facilitate WIN's effectiveness as a CIE platform-builder, the final contribution (5) of this work is a fully functional mobile application *#filters* built for the Android platform[1]. #filters enables users to take photos from their mobile device and apply aesthetically pleasing *image effects*, i.e. similar to the popular mobile application Instagram[2]. In contrast to the limited set of effects found in Instagram, #filters enables users to interactively discover and search for a near limit-less number of image filtering effects through collaborative interactive evolution. Excitingly, in addition to being the very first CIE application available on Android devices, the open-source #filters application serves as an instructional pillar for the community on how to build a CIE application for mobile devices with the WIN framework.

The aggregated results of the domains built with WIN for this dissertation are available to browse at `http://winark.org`.

The next chapter provides relevant background while chapters 3 and 4 review work in the evolutionary domains of music generation and two-dimensional creatures, respectively. The construction of the core WIN library and experimental results are discussed in chapter 5. From there, chapter 6 examines a human study exploring how developers create variant WIN services, while chapter 7 demonstrates a fully functioning Android application built with WIN. The resulting implications are discussed in chapter 8, and the dissertation concludes in chapter 9.

[1]Copyright Google 2015.

[2]Copyright Facebook 2015.

CHAPTER 2: BACKGROUND

Notably, WIN has the potential to augment the existing efforts and impact of individual research projects by adding collaborative features to new or existing evolutionary domains. Thus this chapter reviews previous efforts in both artificial evolution as well as domains relevant to the later demonstrations of WIN.

2.1 Evolutionary Computation

The field of *evolutionary computation* encompasses a range of search algorithms inspired by evolution and Darwinian natural selection [25]. Like in nature, evolutionary algorithms operate on a population of candidates where the most fit individuals selected to continue and reproduce have offspring that are slight variations of those individuals. As such, each evolutionary method must determine a number of important parameters including how to measure the level of fitness of an individual, called a *fitness function*, and what an individual is composed of genetically, called the *representation*. Figure 2.1 gives an intuitive sense of the process of artificial evolution.

An important distinction exists between methods for deciding how individuals are selected and ranked among the population: Predominantly, *automated evolution* is when the evolutionary algorithm employs a predefined fitness function, e.g. distance traveled by a robot in a maze, to assess and rank individuals for selection. In contrast, *interactive evolutionary computation* (IEC; [24]) refers to algorithms where a human, rather than an automated algorithm, rates individuals in the population for selection.

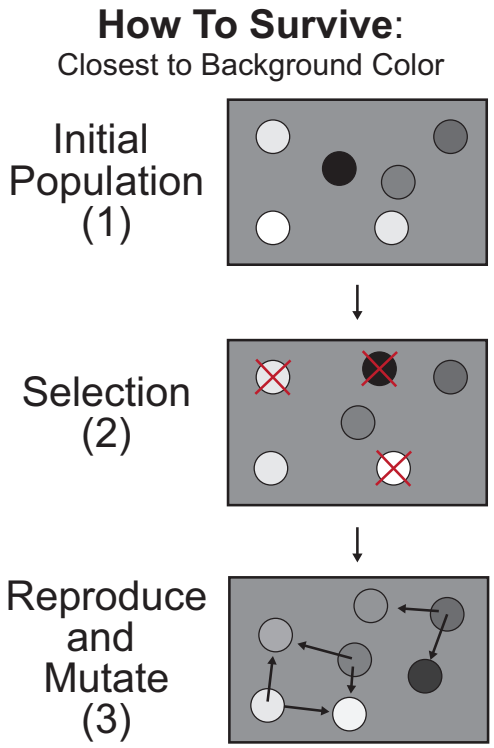


Figure 2.1: **Evolutionary Algorithm Example.** In this example, a simple domain illustrates how the process of artificial evolution progresses in practice. In this environment, all individuals are grayscale circles where those more closely matching the background color are more fit for selection. (1) Evolution is seeded with an initial population of circle individuals. (2) Traditionally, at each generation in evolution, the least fit individuals are removed from the population. (3) Finally, the remaining individuals are then allowed to reproduce individually or in pairs to produce offspring that are slightly mutated (indicated by the directional arrows from the parent circle to the child circle). Here it is evident that some offspring closely match the background while others are less fit than their parents. From there, the evolutionary algorithm returns to step (2), removing the least fit individuals and selecting parents for the next batch of offspring.

In practice, the process of interactive evolution is similar to how humans breed animals. For example, when generating music through IEC, a user is presented with a small collection (the population) of musical pieces created by the program. The user then selects his or her favorite song from the collection that serve as the parents for the next batch of musical pieces (the offspring), where each new child is a slight mutation of the parent. The process repeats until the user find a satisfactory musical piece. Much like the variety found from dog breeding, users experience small variations from parent to child, but over a large number of generations the differences between the current children and the initial population can be dramatic.

Deciding whether to include a human in the evolutionary loop depends on the goals of the researcher. IEC is a popular approach to facilitating creativity in non-experts in a variety of subjective domains, e.g. music or art generation [19, 45, 48, 75, 86, 87], where it can be difficult to define the exact fitness of an individual. Intuitively, it would be hard to numerically estimate how much one song is more fit than another. In contrast, automated evolution is traditionally employed in domains where the end-goal of evolution is well defined mathematically, e.g. a navigator needs to reach the end of a maze or a biped walker need to maximize its distance traveled [38, 55, 71].

In contrast to automated evolutionary searches, IEC experiments rely on sourcing human insight and intuition from users to drive the search process incrementally. While human insight is a powerful resource, it is also limited by the onset of *user fatigue* (when a given user grows tired during the search) [87]. In IEC, unlike automated evolution, each human contributing to the search will eventually experience user fatigue, though differences in individual dispositions will vary when user fatigue occurs.

2.2 Collaborative Interactive Evolution

Interestingly, there are existing evolutionary systems called *collaborative interactive evolution* (CIE) that try to minimize the negative impacts of user fatigue by allowing contributions from many simultaneous users over the course of an experiment [86]. This section explores such experimental systems.

2.2.1 Picbreeder

To specifically address user fatigue, one such CIE application, Picbreeder (a genetic art program for breeding pictures), empowers users through the processes of *branching and publishing* [73], which serve as a form of global checkpointing for the experiment. Through publishing, when users of Picbreeder discover an image that interests them, they can publish the image publicly. Later, the same user or another user can then select that published image as the starting point for his or her own interactive evolutionary search, in a process called branching.

As a result, every image discovered in Picbreeder is either a direct descendant of the efforts of another user or originates from a simple starting image. In effect, while any single user may fatigue at different rates across multiple sessions, the ability to publish any appealing image serves as a checkpoint recognizing every user's incremental progress for themselves and crucially to the public. Notably, these checkpoints mark a continually expanding frontier of images that, like natural evolution, can all be traced to common genetic origin and are preserved as potential stepping stones for future exploration by other users.

Importantly, CIE systems with branching, like Picbreeder (`http://picbreeder.org/`; [73]) for evolving pictures and Endless Forms (`http://endlessforms.org/`; [11]) for evolving three dimensional objects, have been available online for contribution for over four and seven years,

respectively. They continue to offer every picture or object ever discovered for further branching. Figure 2.2 shows a small sample of images found through Picbreeder over the lifetime of the ongoing experiment.

Figure 2.2: **Picbreeder Examples.** A small cross-section of images evolved by users of the Picbreeder service is shown [73]. Because users can branch from previously discovered images, each picture potentially represents the collective efforts of multiple users.

2.2.2 Tools for Collaboration

Though CIE systems like Picbreeder have produced a number of interesting results, engineering such a system is still complex and laborious. Before WIN, a library for assisting researchers in creating collaborative systems from scratch did not exist, but there are many existing tools enabling researchers to build their own experimental domains. Software packages like the Java-based ECJ suite [59] or the C++ framework SFERES [65] provide a collection of classes and data structures to solve common problems typically encountered when constructing new experiments (e.g. building

a user interface and visualizing results). ECJ and SFERES both represent significant contributions to the community, helping to proliferate new domains and advanced simulations.

Moreover, WIN is not the first attempt at collaboration within the evolutionary computation community. Within evolutionary robotics, Bongard [6] created a simulator, called Ludobots, that sets precedent for harnessing amateur users for crowd-sourcing intricate robot controllers (`http://www.uvm.edu/~ludobots/`). This work inspires the idea of exploiting advances in web technology to create distributed EC platforms. WIN builds on the same philosophy with the goal of bringing the benefits of crowd-sourcing to any evolutionary domain.

2.3 The Sodarace Domain

The key to enticing a community to develop for a platform like WIN is to show that new domains can be added easily and systematically. As such, WIN builds upon work across several different experimental domains to demonstrate the versatility of the platform. This section reviews previous such efforts in the artificial life world of Sodarace.

The Sodarace project is a simple two-dimensional physics world consisting entirely of masses, springs, and basic oscillatory muscles [63]. The goal in Sodarace is to create virtual robots and race them in different environments. Both the robots and the environments are usually hand-crafted by users. However, to aid in creating robots, a construction kit is provided to allow discovery and exploration by the community [62, 63]. Figure 2.3 shows a small collection of user-created sodaracers.

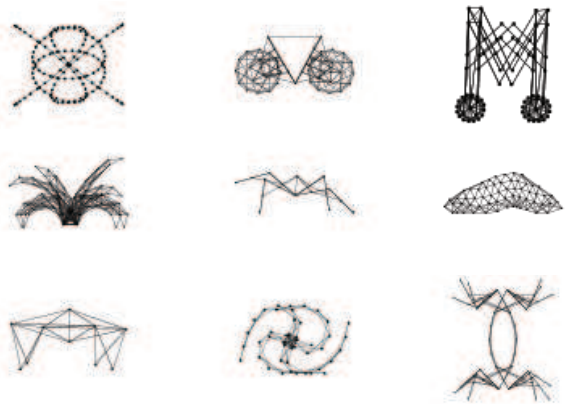


Figure 2.3: **Sodarace Example Creatures.** A collection of human-designed sodaracers [62] that are reproduced from the Sodarace homepage `http://sodaplay.com/`.

SodaRace serves as the chief inspiration for the work in chapter 4 in part because the variety of creature types found with the Sodaconstructor indicates the space of creatures is rich. Moreover, Sodarace is compelling in the variety of creatures users can create, yet lightweight overall, consisting of simple masses, springs, and muscles [63].

The Sodarace project was originally conceived as a type of online Olympics meant to test humans against machine intelligence at the task of designing robot racers. In fact, one redesign of the software includes an evolutionary algorithm that optimizes morphologies for racing. Reflecting the software's educational aspirations, an online repository of creatures and all relevant software packages are accessible in a centralized location [62]. At the peak of popularity, Sodaconstructor, the tool for creating the creatures, was played by about a million active users [63], suggesting its potential as a platform for exploration and discovery. Chapter 4 extends the Sodarace domain by creating a new domain for exploring new creature morphologies automatically called Indirectly Encoded SodaRace (IESoR), then chapter 5 details how WIN goes a step further to allow humans

to collaborate with this automated search process.

2.4 CPPNs, NEAT, and HyperNEAT

Though this dissertation discusses the WIN framework in relation to several distinct experimental domains (detailed in chapters 3, 4, 5, and 7), all the domains utilize a special type of function representation called a compositional pattern-producing network (CPPN; [82]). Notably, CPPNs have been employed as the underlying function representation within a variety of existing evolutionary domains [1, 3, 7, 12, 13, 15, 16, 27, 28, 36, 50, 72, 94], including music generation in NEAT Drummer [41] and MaestroGenesis (chapter 3), picture evolution in Picbreeder [73], and three-dimensional object evolution in Endless Forms [11].

In practice, a CPPN is a connected weighted graph similar to a neural network [82]. In contrast to a traditional neural network, internal nodes within a CPPN are not limited to the same activation function, e.g. a sigmoid function. That is the internal structure of a CPPN is a weighted network that denotes which functions are composed and in what order. The idea behind CPPNs is that sophisticated geometric patterns can be encoded as a composition of functions chosen to represent common regularities [32, 82]. CPPNs can effectively produce patterns in space (such as images or bodies) just as they can produce patterns in time (such as music), which contributes to their effectiveness in multiple domains discussed later in this dissertation.

To understand how a composition of functions could represent fundamental regularities even in multiple domains, consider the composition of internal nodes. Activation functions that are symmetric, e.g. Gaussian, enable the CPPN to create symmetric output, while periodic functions, e.g. sine, impart repetition and segmentation. Crucially, composing symmetric, periodic, and asymmetric functions can yield patterns of repetition with variation much like the motifs found within

nature, e.g. fingers of the human hand. Similar to neural networks, the topology of the network is critical in determining the properties of the pattern produced by the CPPN.

Typically, CPPNs are evolved through an algorithm called NeuroEvolution of Augmenting Topologies (NEAT; [81]), an evolutionary method for maintaining and evolving a population of neural networks over time. Importantly, NEAT begins the evolutionary search with minimal function compositions, and evolves additional complexity during the search, which means networks discovered by NEAT are not limited to a fixed size.

While MaestroGenesis employs NEAT to generate and modify the CPPNs responsible for creating musical accompaniment, IESoR employs a different method extending NEAT, called Hypercube-based NEAT (HyperNEAT) [32, 80], to build creature morphologies. Traditionally, HyperNEAT utilizes CPPNs to encode large neural network connectivity patterns with natural regularities, e.g. symmetry and repetition of structure. While such regularities are useful for neural networks, as chapter 4 demonstrates, they also benefit *bodies* made of connections and joints in a similar way. Similarly, Auerbach and Bongard [2] encoded the bodies of three-dimensional ambulating creatures with CPPNs. The versatility of CPPNs explains why the same underlying representation is chosen across four different domains in chapters 3 through 7.

The next chapter introduces MaestroGenesis [45], an interactive evolution program for novice musicians to evolve full harmonic musical accompaniments encoded by CPPNs. This chapter forms a foundation in IEC research and was a first step on the path to building the broader evolutionary framework of WIN.

CHAPTER 3: MAESTROGENESIS SOFTWARE

Co-created by myself and Amy K. Hoover, MaestroGenesis helps users create complete polyphonic songs through interactive evolution from as little as a single monophonic melody. Importantly, MaestroGenesis is a mature IEC application and a solid experimental basis upon which the larger evolutionary framework detailed in chapter 5 was built.

The capabilities of MaestroGensis are validated through several user studies, whereby MaestroGenesis is shown to create musical accompaniments that aesthetically improve over the course of interactive evolution. Furthermore, some compositions containing accompaniments generated by MaestroGenesis were indistinguishable from fully human-composed compositions, indicating the software can generate musical accompaniment that is human-plausible. A final self-assessment by users of MaestroGenesis reveals the software's potential for helping even experienced musicians explore new creative compositions by providing musical inspiration from evolution. Together, these assessments confirm MaestroGenesis potential as a foundation for the broader evolutionary framework of WIN.

This chapter gives an overview of the MaestroGenesis software and reviews the experimental results from three user studies conducted at the University of Central Florida (UCF).

3.1 MaestroGenesis Software

To assist users in creating musical accompaniments, MaestroGenesis starts by establishing a starting melody provided by the user that will form the rhythmic and harmonic seed of the final accompaniment. This initial melodic input, called a scaffold, provides a rhythmic and harmonic seed to the program. To encourage users to develop new potential compositions, MaestroGenesis employs

interactive evolution to guide search through the space of musical accompaniments. After providing the melodic scaffold, MaestroGenesis then generates ten candidate musical accompaniments for the user to choose from. Through IEC, as the user selects his or her favorite accompaniments, MaestroGenesis will continue creating new batches of candidates from user selections until a final accompaniment is chosen.

Recall from Section 2.4 that MaestroGenesis utilizes compositional pattern producing networks (CPPNs; [82]) to represent musical accompaniments as a composition of functions, and the NeuroEvolution of Augmenting Topologies (NEAT; [79]) method to evolve the CPPNs. Importantly, because each accompaniment is represented internally as a CPPN and the NEAT algorithm will increase CPPN complexity over time, user selections can meaningfully impact the musical complexity of the accompaniments over the course of evolution (as shown by user studies in the next section).

Specifically, my contribution to the MaestroGenesis project was a significant overhaul of the program structure, including an entirely new user interface (UI), to facilitate human-computer collaboration in exploring the musical search space. To help appreciate the magnitude of the change for the end user, figure 3.1 depicts a side-by-side comparison of the original software (NEAT Drummer; [42]) and the final interface for MaestroGenesis. Note that to reduce the learning curve for MaestroGenesis users, hard to read and complex interface elements were removed or hidden and instead replaced with larger and less cluttered buttons and sliders. Additionally, the musical notation was altered to reflect a more intuitive understanding of musical compositions without requiring the user to specifically know how to read staff notation.

NEAT Drummer
(1)

MaestroGenesis
(2)

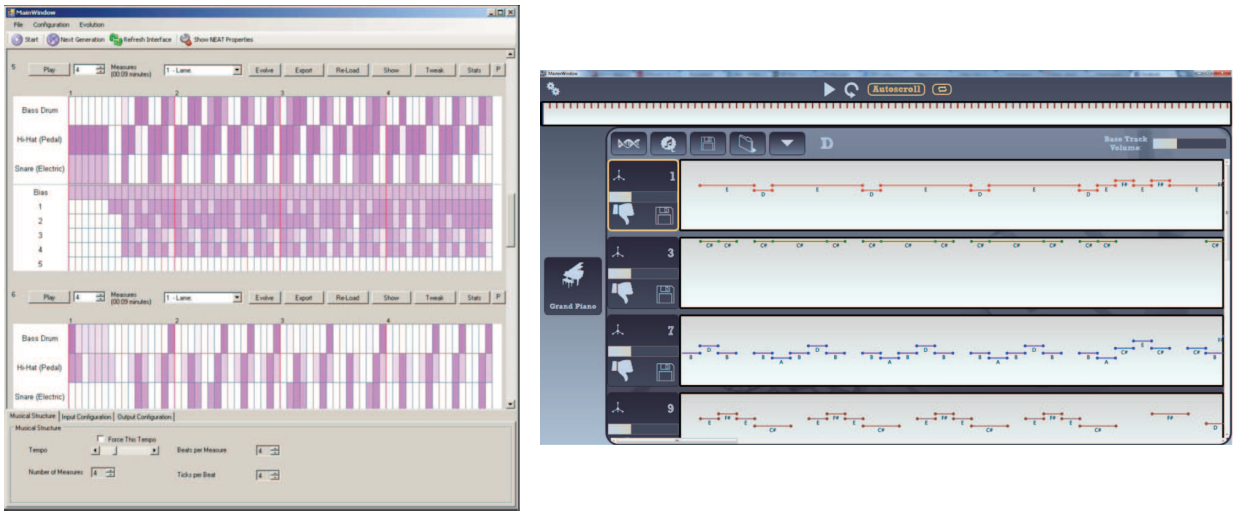


Figure 3.1: **NEAT Drummer to MaestroGenesis.** Built upon NEAT Drummer (1; [42]), the MaestroGenesis (2; [45]) software overhaul focused on improving how the user experienced musical evolution by removing unnecessary user interface complexity and creating a more intuitive musical notation.

3.1.1 Interactive Evolution Framework

Because there can be many appealing musical voices to accompany the original musical inputs, MaestroGenesis encourages users to develop accompaniments through interactive evolution. Once the melodic scaffold is provided to MaestroGenesis and a population of ten accompaniments is displayed, the user rates each candidate as good or bad by pressing the "thumbs-up" button (see figure 3.1). By rating favorable accompaniments higher than less appealing voices, the next batch of candidates tends to possess similar qualities to the well-liked parents. Through interactively evolving these accompaniments, the musical voices increasingly reflect the personal inclinations of the user. That is, interactive evolution enabled MaestroGenesis to better reflect the intent of the user while providing a variety of plausible new accompaniments to continually select. Importantly,

MaestroGenesis provides a strong interactive evolutionary foundation, which inspires both the design and construction of a more ambitious WIN prototype described in Chapter 5.

3.2 MaestroGenesis Experimental Results

With software contributions from both Amy K. Hoover and myself, the overhauled MaestroGenesis program allowed users to create new types of accompaniments. To assess the overall ability of the IEC software to assist in human composition, three primary human studies were conducted with the MaestroGenesis software. In effect, the studies address three questions in particular. What was the impact of interactive evolution on the quality of the accompaniment, can average listeners distinguish between musical pieces that are fully human-composed or partially computer-composed by MaestroGenesis, and can the software empower users to create multi-voice compositions from as little as a single starting melody? The experiments show that resulting accompaniments from MaestroGenesis are indeed musically plausible, improve from human selections during evolution, and can be constructed from as little as a single monophonic input.

To measure the effects of evolution on the resulting accompaniments, the first experiment explores the musical quality of several accompaniments over the course of evolution. Independent listeners sample the generated works and assign a rating to the quality of the accompaniments from the start, middle, and end of evolution. Similarly, to address the level of quality of completed accompaniments, the second experiment tests whether listeners can distinguish between two partially computer-composed and fully human-composed pieces. Finally, the third experiment examines if there is enough information in a single *monophonic* melody to scaffold an entire multipart piece. An additional study of user self-assessment provides a perspective on the users' own perceptions of their experience with MaestroGenesis.

3.2.1 Investigating Accompaniment Evolution

In this first experiment, the focus is on the evolution of the accompaniment. Therefore, the scaffold, i.e. music for which the accompaniment will be evolved, is chosen to meet an established level of quality. That way, it is possible to determine whether the accompaniment can maintain and complement the original quality in the scaffold. For this purpose, the well-known folk song Bad Girl's Lament is chosen, which was sequenced and provided with permission by musician Barry Taylor.

To explore the resulting space created by the MaestroGenesis software, an evolutionary progression of an instrumental accompaniment for Bad Girl's Lament between generations 1 and 12 is studied by highlighting important milestones at generations 1, 6, and 12. This 12-generation progression took about thirty minutes in total for the user to complete. Audio of the results is available at `http://cs.ucf.edu/~pszerlip/maestrogenesis`.

To demonstrate the musical changes user selection in MaestroGenesis can impart during evolution, figure 3.2 shows evolved accompaniment for measures 13 and 14 of Bad Girl's Lament in generations 1, 6, and 12. The pitches in measures 13 and 14 of the first generation differ significantly from those created for generations 6 and 12. Pitches in generation 1 ascend across notes A, B, and C#, followed by B, C#, and D in the next measure. However, in generations 6 and 12, the pattern more closely follows the harpsichord input from the scaffold, demonstrating the influence of the functional relationship on the evolved progressions. For example, they all travel from B to D and back to B in the first measure. However, in the second measure, generation 12 falls back to a C# rather than the B selected for generation 6. This variation imparts a progressive resolution that is missing in the thirteenth and fourteenth measures of generations 1 and 6.

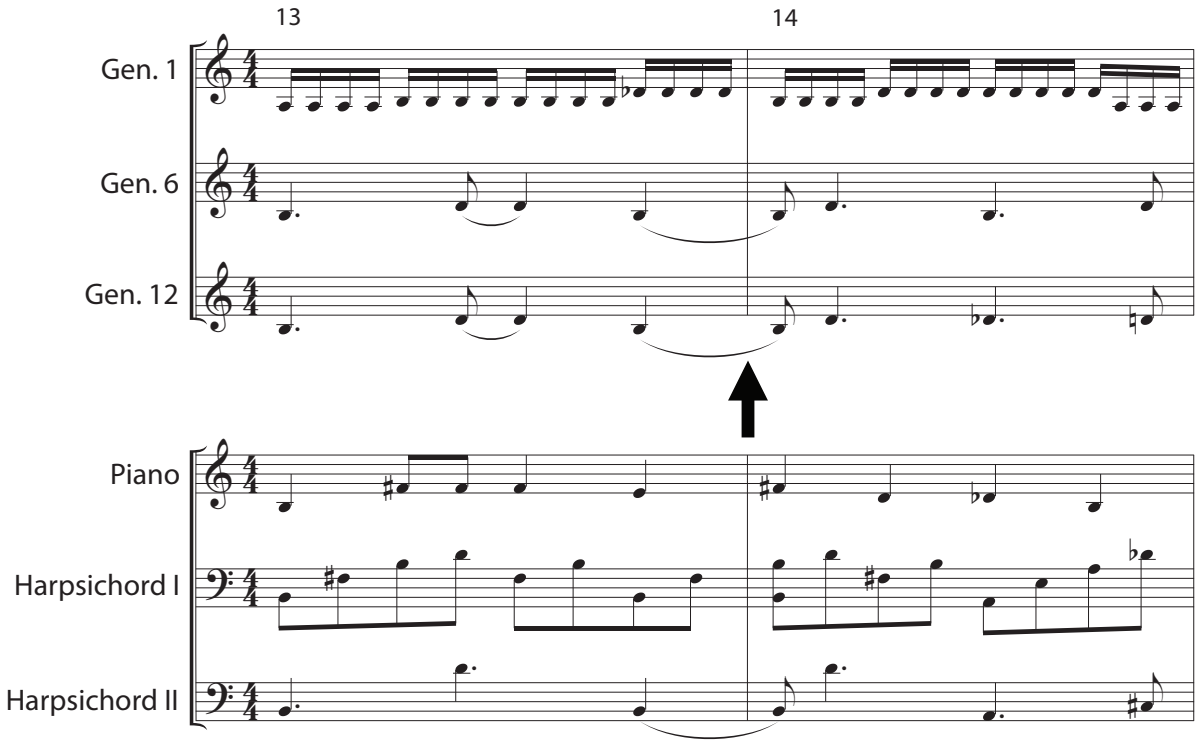


Figure 3.2: **Evolutionary Accompaniment Sequence for Bad Girl's Lament.** Evolved steel guitar accompaniment for generations 1, 6, and 12 of Bad Girl's Lament is shown at top, followed by the pitch and rhythm inputs to the CPPN from the scaffold.

Most importantly, while the three generations of Bad Girl's Lament have similar characteristics, a clear progression is observed over evolutionary time. For example, while generations 6 and 12 are rhythmically similar, generation 1 sounds significantly shorter notes. The pitch evolution progresses similarly to rhythm. From generation 1 to 6 many pitches change, but generations 6 and 12 differ in pitch by only a few choice notes.

To understand the effect of evolution on subjective appreciation, a total of 60 listeners, all of whom are students in a diversity of majors at the University of Central Florida, participated in a survey

after listening to the evolved variants of Bad Girl's Lament. In particular, without knowing which is which, they listened to (1) an intentionally poor-quality control with inappropriate accompaniment (which helps to establish that participants indeed generally agree on something subjective), (2) the original Bad Girl's Lament without accompaniment, (3) the song with accompaniment selected from the first generation of IEC, (4) the song with accompaniment selected from the sixth generation of IEC, and (5) the final selected song with accompaniment from generation 12. For each of the variants, the listener was asked: *Rate MIDI i on a scale of one to ten. (1 is the worst and 10 is the best)*, where i refers to one of the five variants, which are available for listening online at `http://cs.ucf.edu/~pszerlip/maestrogenesis/`.

By establishing the perceived quality of a respected composition, e.g. Bad Girls Lament, it becomes possible to estimate how well evolution compares to professional standards even though MaestroGenesis incorporates *no prior musical knowledge or expertise*. The results from the 60-person listener study, which focused on the same IEC-evolved accompaniments for Bad Girl's Lament from the previous section, are shown in table 3.1. As expected, the control is rated significantly worse than every other example in the survey (at least $p < 0.05$ for all pair-wise comparisons with Student's t-test). This result establishes that listeners likely understood the questions in the survey.

Table 3.1: **Perceived Quality by Survey Participants.** This table shows the average ratings and the mean and standard deviation for the control and four Bad Girl's Lament (BGL) MIDIs.

MIDI Name	**Mean**	**Std. Dev.**
Poor Control	4.35	1.93
BGL without Accompaniment	7.30	1.85
BGL, Generation 1	5.15	2.20
BGL, Generation 6	6.07	1.96
BGL, Generation 12	6.83	1.98

Importantly, generation 6 is judged significantly higher quality than generation 1 ($p < 0.05$) and generation 12 is judged significantly better than generation 6 ($p < 0.05$). Furthermore, although the original MIDI without accompaniment is judged significantly better than generation 6 ($p < 0.001$), it is *not* judged significantly better than generation 12. Thus evolution guided by the human user eventually achieves in a short number of generations a level of quality indistinguishable from that of the original, hinting that MaestroGenesis-generated parts can meet an acceptable level of quality.

3.2.2 Comparing MaestroGenesis to Fully Human Compositions

The aim of the second experiment is to explore whether accompaniments generated by MaestroGenesis can sound human. To explore this question, an additional accompaniment is generated for the folk song Nancy Whiskey, also originally arranged in MIDI format by Barry Taylor and redistributed with his permission. Then, the accompaniments for Nancy Whiskey and the final generation of Bad Girl's Lament from the previous section are included in a "musical Turing Test" to determine whether they are distinguishable from other completely human-composed pieces.

It is important to note that these pieces are chosen for this experiment because they exemplify entirely human compositions that meet a minimum standard of recognizable quality. That way, it is possible to discern whether the generated accompaniments reduce the human plausibility of the work, or whether they complement it successfully, as would be hoped for such an approach.

The main result, which is from only two generations of evolving accompaniment, can be seen and heard at `http://cs.ucf.edu/~pszerlip/maestrogenesis/`. Like the experiment in Section 3.2.1, the interactive evolutionary process was guided by the authors with the same experimental settings as in the previous section.

In the second listener study, anonymous participants were asked to rate examples with and without MaestroGenesis-created accompaniments. The focus in the study is if listeners can discern whether or not a computer is involved in generating the example compositions. Thus the survey is a kind of *musical Turing Test*.

For this study, a total of 66 listeners, all of whom were students in a diversity of majors at the University of Central Florida, participated in the study. The full survey, including the human compositions, is provided at `http://cs.ucf.edu/~pszerlip/maestrogenesis/`. Participants are asked to rate five different MIDIs by answering the following question:

> Based on your impression, how likely is it that any of the instrumental parts in the musical piece found at the following link, <link>, were composed by a computer? "Composed" means that the computer actually came up with the notes, i.e. both their pitch and duration, on its own. (1 means very unlikely and 10 means very likely).

The participants rated a total of five MIDIs: (1) an obviously computer-generated control[1] (which helps to establish that participants understand the question), (2) the version of Nancy Whiskey with computer-generated accompaniment, (3) fully human-composed Chief Douglas' Daughter, (4)

[1]Like in the previous experiment, the control also benefits from notes in the correct key.

fully human-composed Kilgary Mountain, and (5) the version of Bad Girl's Lament with computer-generated accompaniment from generation 12. Thus the main issue is whether participants judge piece 2 and piece 5, which have accompaniments evolved with FSMC, as distinguishable from piece 3 and piece 4, which are entirely composed by humans.

The complete results of this study are shown in table 3.2. On average, the 66 participants judge the intentionally-poor example as significantly more likely to be computer-generated than any other song in the survey ($p < 0.001$ according to Student's t-test). This difference indicates that participants understand the survey.

Table 3.2:**Survey Results** (lower means more human-like).

MIDI Name	**Mean**	**Std. Dev.**
Control	7.82	2.15
Nancy Whiskey with Accomp.	5.45	2.65
Chief Douglas' Daughter	4.32	2.61
Kilgary Mountain	4.86	2.39
Bad Girl's Lament with Accomp.	4.82	2.44

Although the accompanied Nancy Whiskey is judged significantly more likely ($p < 0.05$) to be computerized than the human song Chief Douglas' Daughter, it is not judged significantly more likely than Kilgary Mountain to be computerized. This result indicates that the accompanied Nancy Whiskey can pass the musical Turing test, i.e. people cannot distinguish it from a song that is entirely human-generated.

The Bad Girl's Lament accompaniment is even more difficult for participants to differentiate. It is not judged significantly more likely to be computer-assisted than either of the human pieces, i.e.

Chief Douglas' Daughter or Kilgary Mountain. In fact, on average, MaestroGenesis-accompanied Bad Girl's Lament scored slightly *less* likely to be computerized than the entirely human song Kilgary Mountain.

These results validate that evolved accompaniments through MaestroGenesis are at least plausible enough to fool human listeners into confusing partly computer-generated compositions with fully human-composed ones, even though the software requires almost no a priori musical knowledge to operate it.

3.2.3 Generating Polyphonic Accompaniment

The final experiment in this section is designed to show how users can generate multipart pieces from just a single monophonic melody with MaestroGenesis. A creative self-assessment from users of the program studies their experience of the process. This experiment explores an important issue in establishing the breadth of potential applications of MaestroGenesis, not simply adding a single additional accompaniment.

For this experiment, three undergraduate independent study students, (Marie E. Norton, Trevor A. Brindle, and Zachary Merritt) composed in total three monophonic melodies. From each of these user-composed melodies, a multipart accompaniment was generated through MaestroGenesis by the author of the originating melody. Two other multipart accompaniments were generated by one of the students for the folk song *Early One Morning*. The most important point is that no musical expertise was necessary to apply to the final creations beyond that needed to compose the initial monophonic melody in MIDI format. Thus, although results may sound consciously arranged it is important to bear in mind that all the polyphony you hear is entirely the output of MaestroGenesis. The original melodies, accompaniments, and CPPNs are available at `http://cs.ucf.edu/~pszerlip/maestrogenesis/`.

MaestroGenesis provides significant freedom to the user in how to accumulate the layers of a multipart piece. In general, the user has the ability to decide from which parts to generate other parts. For example, from the original melody, five additional parts could be generated at once. Or, instead, the user might accumulate layers incrementally, feeding each new part back into the software to evolve yet another layer. Some layers might depend on one previous layer, while others might depend on multiple previous layers. In effect, such decisions shape the subtle structural relationships and hence aesthetic of the final composition. For example, evolving all of the new parts from just the melody gives the melody a commanding influence over all of the accompaniment, while incrementally training each layer from the last induces a more delicate and complex set of harmonious partnerships. Overall, the student composers took advantage of this latitude in a variety of ways. Scores, audio, and the full details of the procedures followed in each case are at `http://cs.ucf.edu/~pszerlip/maestrogenesis/`.

Interestingly, despite inputting the same initial monophonic melody, Early One Morning (Song 1), a single user generated multiple accompaniments through MaestroGenesis that resulted in entirely different rhythmic and harmonic influences. Overall, the arrangements appear *composed* even though they are all evolved through distinct interactive breeding processes.

A key motivation for these polyphonic experiments is that they reflect a likely usage case for novice musicians whereby they input a single melodic line to MaestroGenesis and the software then helps generate a full multipart musical piece. Most importantly, in a self-assessment of the MaestroGenesis users responsible for creating the multipart compositions in this third experiment, participants indicated that MaestroGenesis provided satisfying ideas for composing creatively. For instance, when asked if "FSMC [MaestroGenesis] helped me explore a broader range of creative possibilities than I could before," each respondent indicated that MaestroGenesis helped them explore new areas of their creative search space. Surprisingly, one student noted that "FSMC [MaestroGenesis] freed me from my normal stylistic tendencies," while another stated that "I typically follow a sort

of pattern when I compose, but FSMC [MaestroGenesis] expanded my thinking."

Thus drawing from the polyphonic experimental results and user self-assessment, MaestroGenesis is shown to be an effective creative tool for composing full multi-part musical compositions.

3.3 Implications

Through experimental evidence, MaestroGenesis is capable of assisting users in creating musical accompaniments that sound plausible to other humans. More importantly, the MaestroGenesis software has the ability to facilitate the creativity of its users even in the absence of musical knowledge.

Despite the underlying complexity of the CPPN-based representation for each musical component, both the design of the software and the IEC framework play a critical role in enabling novice users to interact with a complex search space. That is, any user can grasp the intuitive process of selecting accompaniments they prefer while discarding the rest. The UI further enables the user to direct their search through the musical space with simplified interactive buttons and sliders. The result is an effective creativity enhancing tool.

An important implication is that the MaestroGenesis software represents a solid and validated experimental basis for extending such an IEC system into a more ambitious evolutionary platform. With similar functionality, this broader platform could enable almost any domain of interest to be situated within the same framework while allowing users with little domain knowledge to meaningfully explore within sophisticated design spaces.

CHAPTER 4: INDIRECTLY ENCODED SODARACE

While the last chapter discussed the MaestroGenesis software and successfully generating appealing accompaniments evolved by users, the domain of musical accompaniments presents significant challenges for assessing the final products of evolution. Importantly, though the user can discover musical accompaniments through interactive evolution, validating the quality of the results is an arduous process that often requires subjective human studies.

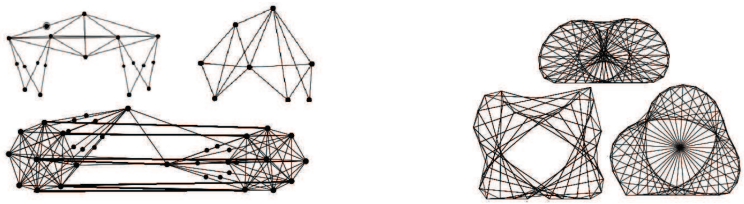
To more effectively build and test a broader evolutionary framework, a new domain can help to extend the ideas first investigated in MaestroGenesis to a wider and more objective investigation of collaborative search in general. Recall from Chapter 1 that there is an interest in the artificial life (ALife) community to conduct and promote open science [88], which serves as a guiding principle for the framework detailed in Chapter 5.

Notably, a considerable amount of research in ALife has investigated the evolution of artificial creatures. For example, visually compelling research in artificial creature evolution by Karl Sims [76] inspired many efforts to reproduce and extend such work, including experiments in open-ended evolution [8, 9, 60, 77, 78, 93], morphological innovation [5, 51, 54], and locomotion [14, 56, 64].

The work in this chapter addresses the need for a visually compelling domain for the WIN framework and builds upon previous work in ALife by introducing a new lightweight two-dimensional creature evolution domain called *Indirectly Encoded SodaRace* (IESoR). This new domain, IESoR, is inspired by a previously-existing project called Sodarace [62, 63] in which two-dimensional simulated creatures made of masses, springs, and muscles ambulate according to their particular morphological configuration. In the initial realization of Sodarace, humans designed the body morphologies by hand and then raced them together competitively. The program attracted nearly

one million contributions from users across the world [62]. The result was a broad collection of diverse human-designed morphologies (figure 4.1a) that resemble to some extent the output one might expect from an artificial life world.

However, IESoR is not the first attempt at automatically evolving Sodarace creature morphology. Later versions of the Sodarace software added an evolutionary module that could in fact evolve new morphologies without human assistance, but because the evolutionary component is relatively simple, these evolved morphologies capture only a small subset (figure 4.1b) of the diverse possibilities suggested by the many known human designs.



(a) (b)

Figure 4.1: **Sodarace Examples.** Human-designed racers (a) exhibit diverse strategies and morphologies for ambulation while those produced through the evolutionary optimizer (b) share an amoeba-like morphology and similar ambulation (reproduced from [62]).

In contrast, the key insight behind IESoR is utilizing compositional pattern producing networks (CPPNs; [82]) to encode regularities and symmetries within the morphological connectivity of Sodaracer bodies. For more background, see Section 2.4 for an overview of CPPNs.

The potential of this system is first demonstrated by successfully finding a collection of virtual creatures all proficient in walking that exhibit a large variety of morphological traits, e.g. short

and wide or tall and skinny. Further experiments with IESoR reveal that with small adjustments to the underlying search objectives, an entirely new set of morphologically-diverse creatures can be uncovered that are instead proficient in jumping. Such variety is achieved in both tasks through the *novelty search with local competition* [57] approach, which is designed to collect a diversity of solutions within a single evolutionary run.

The resulting variety establishes IESoR as a visually rich evolutionary domain, motivating its inclusion as an initial domain for collaborative evolution in the prototype WIN framework described in Chapter 5.

4.1 IESoR Approach

This section describes the implementation details of IESoR, and explains the variant of HyperNEAT that enables the creation and evolution of Sodarace body morphology.

4.1.1 IESoR

IESoR implements three primary properties derived from Sodarace (figure 4.2):

1. The environment is two-dimensional and creatures consist solely of masses, springs, physical joints, and muscles.
2. In creature bodies, masses are implemented by nodes and springs are connections attached at the joints.
3. Muscles manipulate the length of connections, leading to motion.

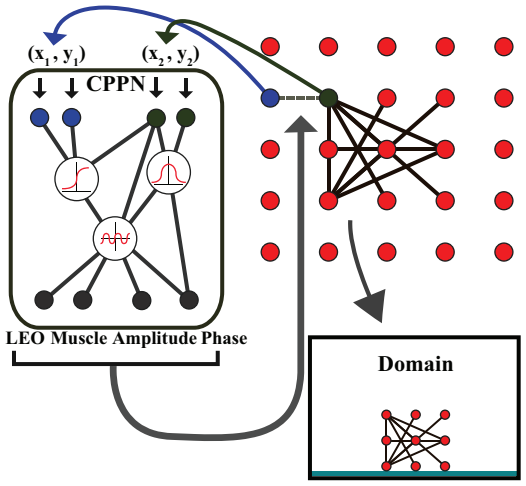


Figure 4.2: **Creating a Sodarace-like body using a HyperNEAT CPPN.** In regular HyperNEAT, the CPPN (left) would query the substrate (right) to determine the weights and presence (determined by the LEO output; [90]) of its connections. However, in IESoR the CPPN outputs the muscle, amplitude, and phase parameters for each queried connection instead of a connection weight. That way, the CPPN in effect describes the properties of a Sodarace body instead of a neural network, yet still with the same benefits of HyperNEAT as usual. The resultant creature is placed into a two-dimensional world where it attempts to ambulate.

In contrast to more complicated three-dimensional domains [2, 54, 57, 76], to support robust ALife evolution IESoR is designed to be simple to modify and inexpensive to simulate. In the spirit of accessibility and extensibility of the Sodarace project, IESoR implements a Sodarace-like simulator in JavaScript built on top of Box2D (box2d.org), an open-source two-dimensional rigid body physics engine. There is a small performance hit for programming in a scripting language, but JavaScript allows the domain to be accessible through the browser for most modern computing devices, from phones to tablets to more traditional PCs. In addition, Box2D physics enables rich environments for testing creature morphologies. Finally, Box2D has been ported to most popular

programming languages, which means IESoR could be ported without significant effort.

4.1.2 Encoding Morphologies with HyperNEAT

Bodies inside of IESoR consist of masses with variable or fixed length constraints. Each constraint, or connection, is represented by a distance joint in Box2D (i.e. a constraint on the length between two masses) and has three distinct properties:

1. The joint is either variable or fixed length (i.e. a muscle or a bone).
2. The change in distance during muscle contraction is the muscle amplitude.
3. The phase shift of the sinusoidal function controlling muscle length is the muscle phase.

Fixed length connections, or bones, do not receive a magnitude or phase from the CPPN.

Recall from Section 2.4, that HyperNEAT traditionally utilizes compositional pattern producing networks (CPPNs) to encode large connectivity patterns for neural networks, where each CPPN is simply a composition of mathematical functions. Formally, CPPNs in HyperNEAT are functions of geometry (i.e. locations in space) that output connectivity patterns whose nodes are situated in n dimensions, where n is the number of dimensions in a Cartesian space.

Consider a CPPN that takes four inputs labeled x_1, y_1, x_2, and y_2. This point in four-dimensional space also denotes the connection between the two-dimensional points (x_1, y_1) and (x_2, y_2), and the output of the CPPN for that input thereby represents the weight of that connection (figure 4.2). By querying every possible connection among a pre-chosen set of points in this manner, a CPPN can produce a connectivity pattern, wherein each queried point is a node position. Because the connections are produced by a function of their endpoints, the final structure is produced with knowledge of its geometry. In effect, the CPPN paints a pattern on the inside of a four-dimensional

hypercube that is interpreted as the isomorphic connectivity pattern, which is the origin of the name hypercube-based NEAT (HyperNEAT). Connectivity patterns produced by a CPPN in this way are called substrates so that they can be verbally distinguished from the CPPN itself, which has its own topology. While the substrate in the original HyperNEAT is interpreted as an ANN, in IESoR the substrate is a creature's body.

Note that HyperNEAT paints a four-dimensional pattern across the weights of a network by querying the CPPN for every pair of nodes in the substrate. The insight is to take this concept of a substrate and extend it to two-dimensional morphologies. Instead of painting a pattern of weights across the substrate, the CPPN encodes both what joint constraints should exist between masses on a two-dimensional plane and their three virtual properties (i.e. bone or muscle, amplitude, and phase). For this purpose, the CPPN requires four outputs (as shown in figure 4.2).

Before clarifying how a HyperNEAT substrate can be used to represent a morphology, it is important to consider the placement of bones and muscles in natural body plans. The skeletal system is crucial to mobility at a fundamental level. Equally important to where bones are placed in a body plan is the concept of where bones *are not* placed. If a rough representation of the human body was drawn on a small grid of dots, the principle of symmetry is as important as the fact that there is no bone connecting the tip of the foot to the top of the skull. Morphologies generated in IESoR ideally also should usually respect this simple principle of locality.

Conveniently for this purpose, HyperNEAT can be expanded with a special Link Expression Output (LEO) [90] to generate an *expression pattern* that controls whether connections are expressed at different locations independently of other CPPN outputs. In Verbancsics and Stanley [90], HyperNEAT with LEO was seeded with a bias towards favoring locality although evolution could adjust this bias during search.

To generate a morphology using an n-by-n grid of nodes as the substrate, for each node location in the substrate (figure 4.2), the CPPN queries all other node positions. The (x, y) coordinate of nodes *i* and *j* are denoted as (x_i, y_i) and (x_j, y_j), respectively. The input into the CPPN is thus x_i, y_i, x_j, y_j, and there are four outputs. First, the LEO output (which is a step function) is checked for a positive value. If LEO is positive, a connection is placed between nodes *i* and *j* from (x_i, y_i) to (x_j, y_j). Then the output that determines whether the connection is a bone or a muscle is queried. If the output value is below a pre-defined *muscle threshold*, the connection becomes a fixed-length constraint. Otherwise, the constraint is a muscle, and the amplitude and phase of the muscle contraction are read from the remaining two CPPN outputs. Finally, to further reduce complexity in the resultant morphologies and keep computational costs low, pairs of points greater than a third of the diagonal length of the substrate are not queried while constructing the two-dimensional creatures. An example of a fully constructed morphology is shown in the lower right of figure 4.2.

After assembling the masses and joints, the bodies are placed in a simple Box2D environment consisting of the ground, gravity, and friction. As the world is simulated, muscles oscillate according to the amplitude and phase values defined by the CPPN, while bones remain a fixed length. Creatures occupy distinct Box2D environments, and nodes cannot collide with each other.

4.2 Experiments

Though its creatures are mainly hand-crafted, Sodarace shows that the space of possible two-dimensional body types is likely filled with creatures capable of movement. As noted in the Background section, the Sodarace Kiosk went on to create an automated approach to generating creatures, but resulted in a highly restricted space of bodies. In contrast, two experiments described in this section are constructed with IESoR to find sodaracers capable of both walking and jumping

behavior. These experiments are designed to show that not only is an automated approach capable of designing two-dimensional walkers and jumpers, but the method can also produce a *wide variety* of different means for locomotion, thereby giving hope for further application of Sodarace-like creatures in artificial life.

4.2.1 Novelty Search with Local Competition

To best demonstrate the morphological diversity possible in IESoR, Pareto multiobjective search (based on NSGA II) [26] including both novelty and local competition [57] is implemented to explore the space of body types. The original experiments utilizing Pareto multiobjective search with novelty and local competition yielded a diverse group of ambulating three-dimensional morphologies all within a *single run* of evolution [57]. Maintaining and exploiting diversity across evolution is both an impressive and important part of validating the potential for future artificial life research with IESoR and its inclusion in the larger WIN platform.

The first of the three objectives that make up novelty search plus local competition is novelty search, which was introduced by Lehman and Stanley [55, 56] to avoid the common pitfall of evolution prematurely converging on a deceptive objective. Novelty search aligns well with the aims of these experiments because the hope is to find a diversity of novel creatures. Joachimczak and Wrobel [51] have shown before that novelty search can be effective for this purpose. The characterization of creature novelty for the novelty search component can significantly impact evolution and strongly bias the resulting creatures discovered. In these experiments, novelty search characterizes creatures by their width, height, and mass (as measured by the number of nodes and the sum of the connection lengths) at the first time-step of the simulation, which should lead to a visually diverse population. The novelty metric is the squared Euclidean distance separating two individuals in this characterization space, and thus the novelty of a creature is proportional to how

different its starting morphology is from that of other creatures currently in the population. Such a characterization space especially encourages creatures with varying widths, heights, and masses.

The second objective, local competition, forces individuals to compete *only* with those who are characterized as similar [57]. The idea is that within novelty search it is possible to push individuals who are similar with respect to the behavior characterization to compete locally to be the best of their type. That way, globally novelty search probes a wide variety of possibilities, but locally individuals optimize to be the best they can. In IESoR, creatures who are locally close share similar widths, heights, and masses, ideally indicating a similar morphology. Local competition is the mechanism for pressuring individuals with related morphologies towards more effective locomotion.

Importantly, by changing the local competition objective, the search can be heavily biased to find morphologies capable of fundamentally different movement strategies. Therefore, the local competition component is the primary source of difference between the two experiments described in this chapter. For the first experiment, the local competition objective is the horizontal distance traveled by an individual. The second experiment measures instead the maximum vertical distance above the physical ground achieved during the individual's simulation. That is, the first experiment encourages walking, while the second experiment encourages jumping. Note that by modifying only this second objective, these experiments allow the direct comparison between morphologies generated by both experiments to best isolate the effects of local competition during search.

As in [57], the Pareto multi-objective search has three objectives: novelty, local fitness, and finally *genotypic diversity*. The genotypic diversity objective encourages exploring innovative genotypes by assigning higher values to more novel genotypes. That way, new genotypes created by HyperNEAT are not initially penalized and thereby have a chance to optimize to reach their potential. This genetic diversity objective is in effect a multiobjective-compatible substitute for the usual spe-

ciation mechanism in NEAT, which serves the same purpose. Additionally, the genotypic diversity objective is also localized within the characterization space; similar in motivation to that of local competition, local genetic diversity ensures that genotypic diversity is not only exploited in those characterization niches in which such diversity is incidentally most easily expressed.

In all setups, the distribution of individuals in behavioral space as well as their overall performance is recorded. The idea is to quantify how much morphological diversity is discovered and maintained and how well each behavioral niche is being exploited overall throughout a run.

4.2.2 Experimental Parameters

The overarching multiobjective algorithm is based on NSGA II [26]. The population size is 120 for all runs, and the walking experiment ran for 1,200 generations while the jumping experiment went for 300 generations. The nearest-neighbor size for novelty search and local competition is 20. The three morphology dimensions used to characterize novelty (i.e. width, height, and mass) are rescaled so that their values fill the range between zero and three. The selection method for NSGA II was tournament selection (with tournament size two), and other parameters followed precedent [57], which in turn used the parameters of [54].

4.3 Results

The intent of these experiments is to demonstrate that a wide variety of walkers exists in the encoding space defined by IESoR, thereby establishing the viability of IESoR for future ALife and IEC research. Thus, as opposed to machine learning experiments aimed at demonstrating optimality, the aim in these experiments is to show both diversity and competence. Recall also that novelty search plus local competition is designed to return a significant coverage of possible solutions from a sin-

gle long run. There is precedent for demonstrating the diversity that results from such a search. For example, Lehman and Stanley [57] measured the height and mass of three-dimensional morphologies from novelty search plus local competition to show the breadth of morphologies discovered by evolution, while Joachimczak and Wrobel [51] used principal component analysis (PCA) to demonstrate coverage across morphological space after novelty search. Following this precedent, to quantify IESoR's ability to create diverse walkers, PCA is run across characterizations in both experiments of all generated creatures during evolution to create a visualization of the resultant diversity.

In particular, to characterize morphological diversity in IESoR for the purpose of visualization, three dimensions that describe gross creature characteristics (i.e. width, height, and mass) are projected into a two-dimensional space by the PCA algorithm. However, while PCA with this information can reveal the diversity across the morphological space, the goal of this analysis is also to give a sense of the competence of such creatures as well. That way it becomes possible to observe the *diversity of competent creatures* instead of just diversity overall. Therefore, in the visualization of the PCA output in both figures 4.3 and 4.5, to ensure each graph shows the diversity of only competent walkers or jumpers, only points for walkers that ambulate beyond 200 units and jumpers that launch higher than 10 units off the ground are displayed. Furthermore, the size of each point's radius is proportional to the absolute performance within the task, either walking or jumping.

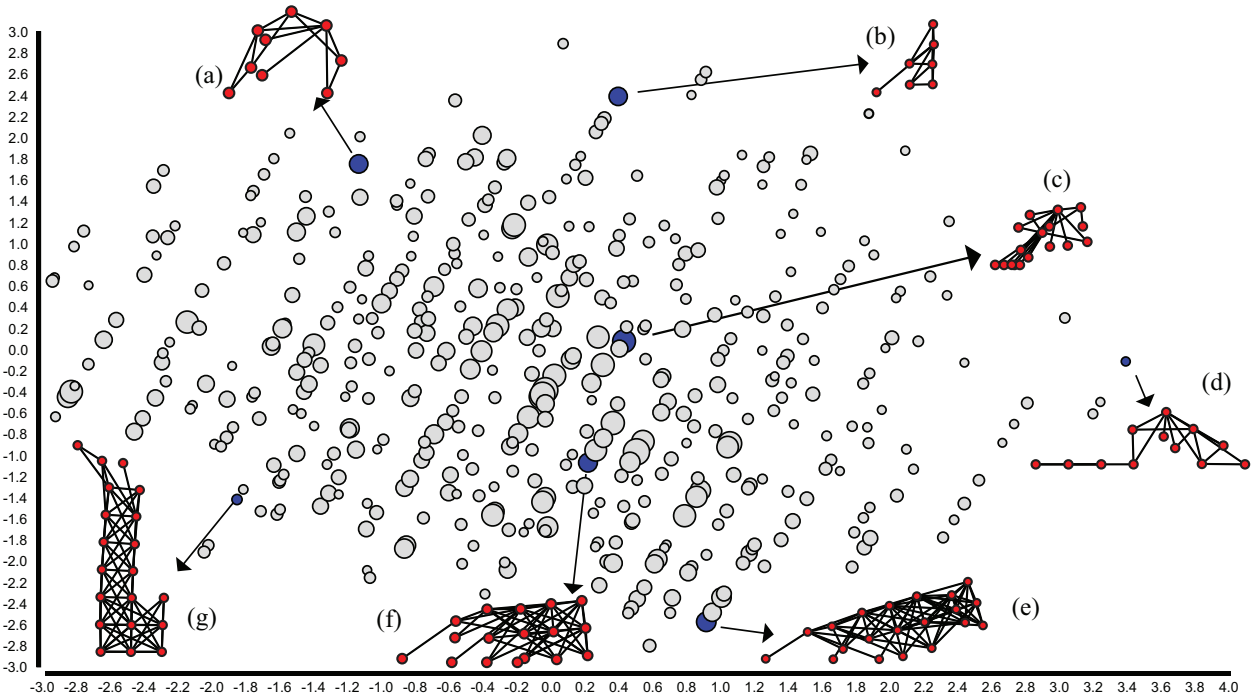


Figure 4.3: **PCA-based Visualization of Morphological Diversity and Movement Distance Performance.** The location of each point represents its respective creature morphology, while the size indicates the absolute fitness. All points shown are for creatures able to walk at least 200 units. A total of 28.1% of all 1,600 possible bins are filled with competent walkers, suggesting the diversity of ambulation methods. Furthermore, several creatures are shown to give a sense of qualitative diversity. The creatures for every point in this visualization can also be viewed in motion at `http://eplex.cs.ucf.edu/iesor/live/walk`.

Because thousands of points result, the visualization is further refined to reduce clutter and ensure that each point represents a genuinely unique individual. For this purpose, the plane is discretized into 40×40 equally sized "bins." Creatures are placed into bins according to the coordinate assigned by the PCA process. Conceptually, each bin thus represents a similar area in morphological

space, and the creature assigned to a bin that performed the best among all in that bin is chosen as the representative of that bin. That way, the circles in figure 4.3 and 4.5 show the best performance for the morphological class represented by its respective bin, and each circle represents a distinct morphological class. Any bin without a representative is shown as empty space in both figures.

For the ambulation experiment, of the 1,600 possible bins, 450 are filled with individuals who can ambulate the minimum distance, covering in total 28.1% of all possible bins. Furthermore, the visualization in figure 4.3 exhibits the breadth of coverage of competent morphologies. In effect, IESoR with novelty search plus local competition uncovered hundreds of unique and effective ambulation methods covering a significant breadth of conceivable strategies.

Additionally, for the jumping experiment, of the 1,600 possible bins, 500 are filled with individuals who launch off the ground to a minimum height of 10 units, covering 31.25% of all possible bins. Figure 4.5 depicts a range of morphologies capable of making at least one jump above 10 units.

Equally important as this quantitative perspective is a qualitative analysis of the breadth of behaviors. It is important to note that every behavior in figures 4.3 and 4.5 can be viewed at `http://eplex.cs.ucf.edu/iesor/live/walk/` and `http://eplex.cs.ucf.edu/iesor/live/jump/` respectively, through a special online interface where the user can click on any point and see the corresponding creature behavior. This fast interactive visualization of hundreds of creatures is possible in part due to the lightweight, inexpensive nature of Sodarace-like creatures, which is one of their potential advantages for researchers in artificial life. Figures 4.3 and 4.5 both show a sampling of morphologies, while figures 4.4 and 4.6 show a subset of those at different stages of motion.

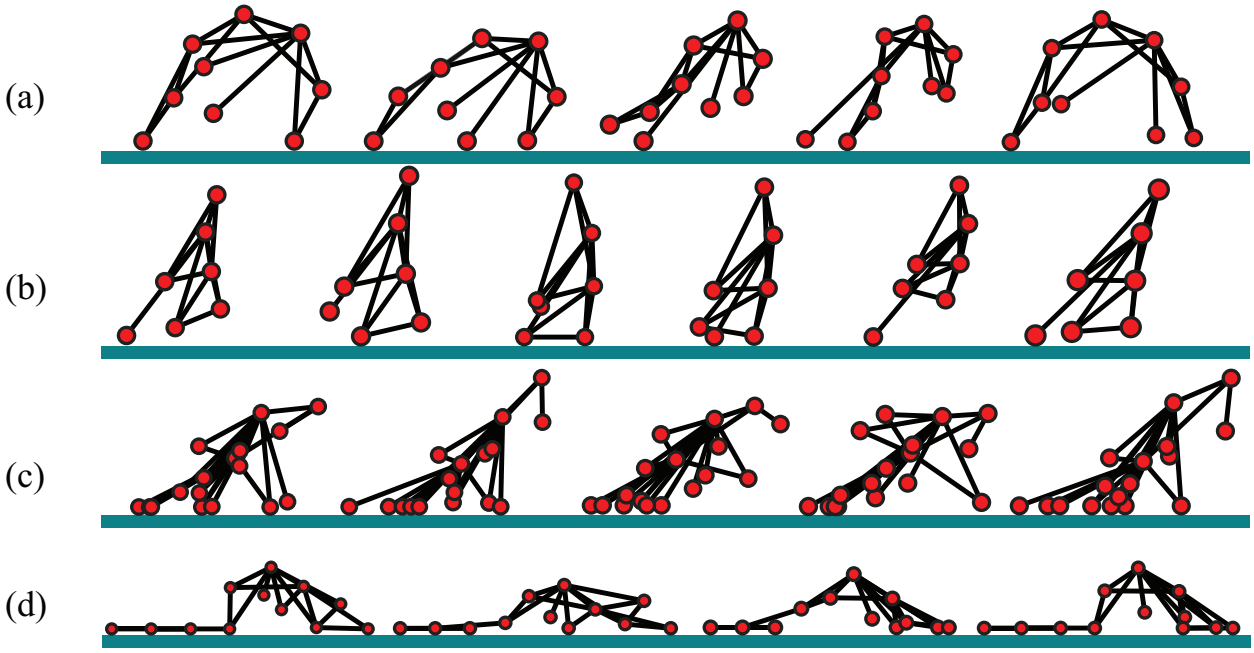


Figure 4.4: **Creature Motion Over Time.** The motion (from left to right over time) of a small sample of successful creatures evolved in IESoR is shown. The letters (a)–(d) correspond to those in figure 4.3.

An additional important further qualitative observation is the significantly broader diversity seen in IESoR compared to the original Sodarace evolver's amoeba-like creatures shown in figure 4.1b. Among those that can be observed are gaits based on loping (degrading into pushing) (figures 4.3a/4.4a), pogo-stick hopping (4.3b/4.4b), multiple cascading octopus legs (4.3c/4.4c), dragging (4.3d/4.4d), complex galloping (4.3e), sliding and pumping (4.3f), and bouncing into a long dive (4.3g). Some strategies depend on an initial burst of propulsion, while others rely and stable and consistent ambulation. Some of the very best gaits (largest circles in figure 4.3) involve galloping or hopping, though even among the very best the diversity of approaches is significant.

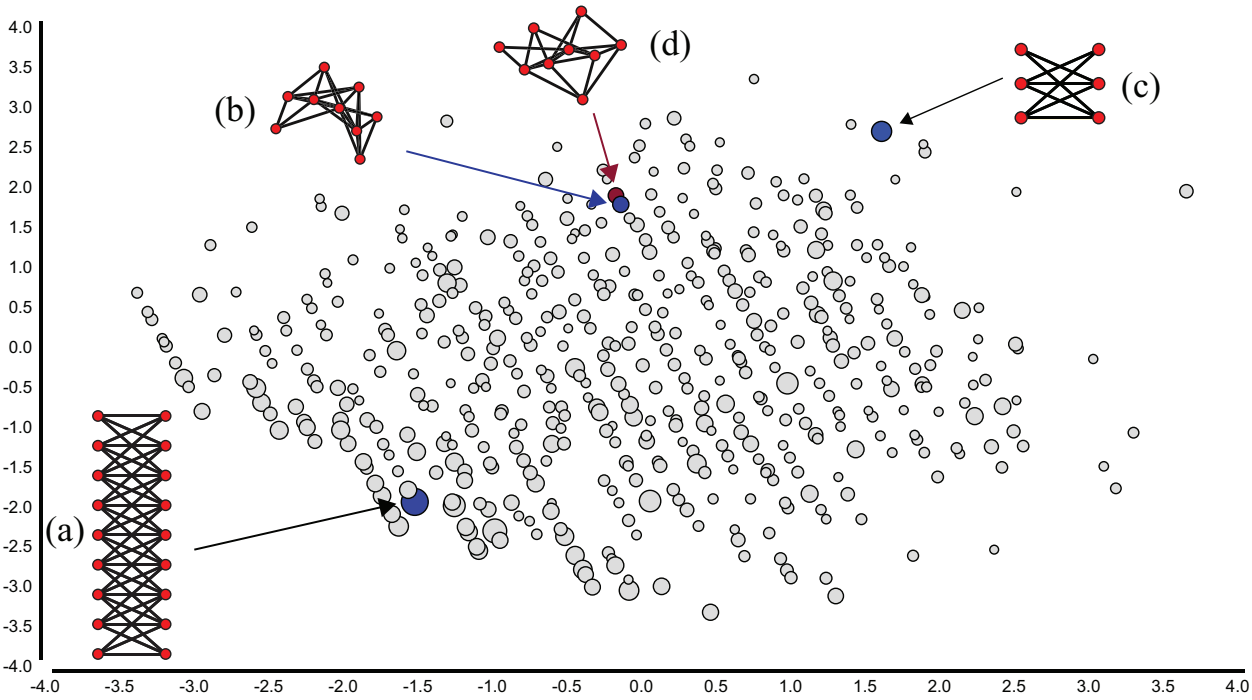


Figure 4.5: **PCA-based Visualization of Morphological Diversity and Jump Height Performance.** The location of each point represents its respective creature morphology, while the size indicates the absolute fitness. All points shown are for creatures able to jump at least 10 units. A total of 31.25% of all 1,600 possible bins are filled with competent jumpers. The creatures for every point in this visualization are viewable in motion online at `http://eplex.cs.ucf.edu/iesor/live/jump/`.

Note that by setting the local fitness objective to be jump height, IESoR demonstrates a capability to find sodaracers with behaviors not originally anticipated by the Sodaconstructor. As the name implies, sodaracers are designed for racing, but IESoR is able to discover a breadth and variety of jumping sodaracers just as effectively as walking sodaracers. As with walking sodaracers evolved by IESoR, creatures found during the jumping search employ a variety of behaviors to excel at the

local fitness objective. Some jumping behaviors rely heavily on symmetric bodies and symmetric muscle contractions (figures 4.5a/4.6a) to launch into the air, while other creatures engage multiple muscles at a single node to spring upwards (4.5c/4.6c). As is the case with creatures from figures 4.5b and 4.5d, even if two individuals have nearly identical initial morphologies and are therefore located in the same location of the PCA space, the method for propelling the body into the air can be entirely distinct (4.6b and 4.6d).

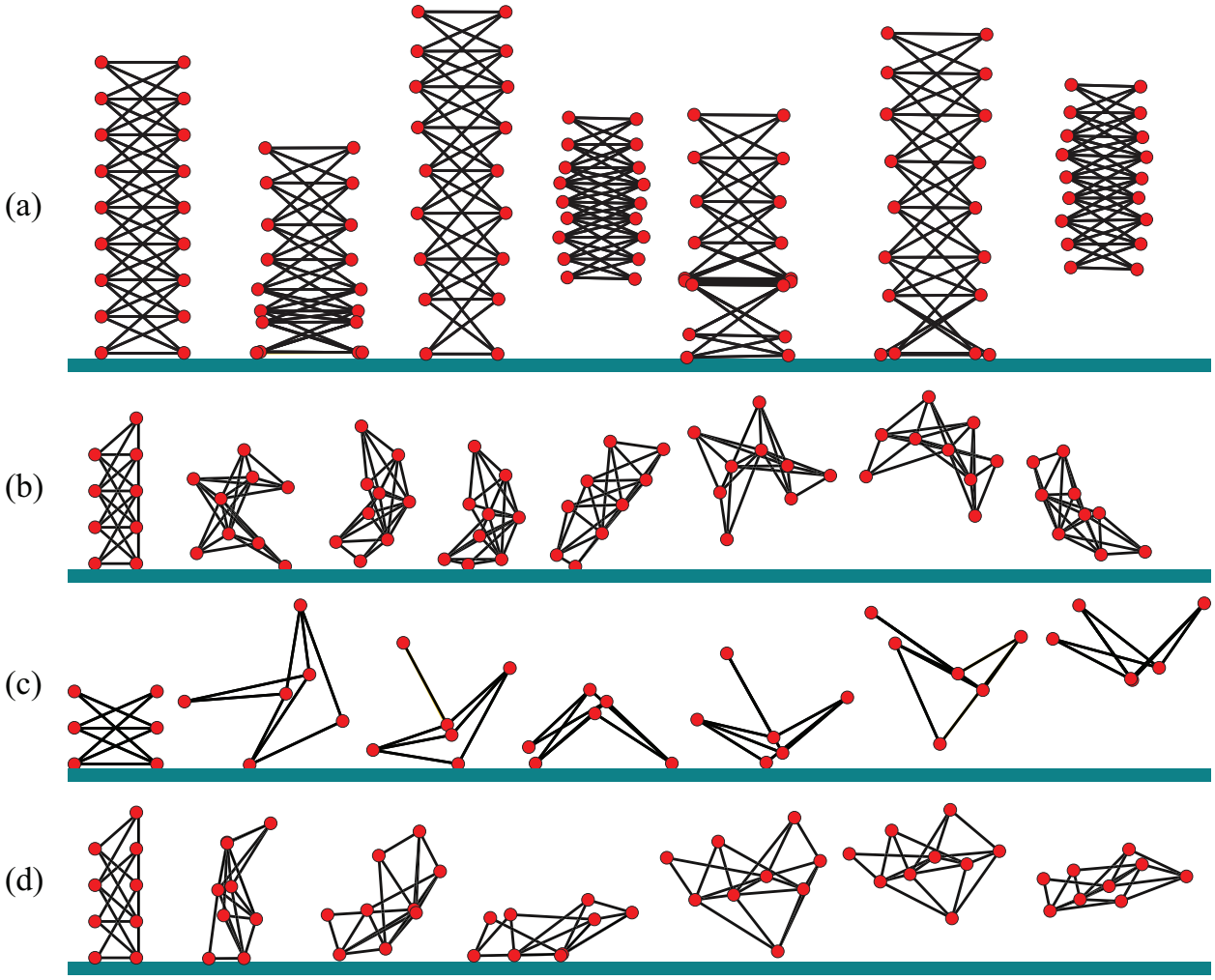


Figure 4.6: **Creature Motion Over Time.** The motion (from left to right over time) of a small sample of successful jumping creatures evolved in IESoR is shown. The letters (a)–(d) correspond to those in figure 4.5.

4.4 Implications

Experimental results from the PCA-organized map of discovered creatures in figures 4.3 and 4.5 provide a qualitative view of the breadth and variety of creatures IESoR can collect in the space of Sodarace bodies. Such results validate that IESoR can consistently produce a visually distinct and morphologically-diverse collection of walking and jumping gates in a limited computational environment. That is, IESoR is a promising new domain for artificial life research, and a lightweight domain for exploring a potentially vast and complicated design space.

Recall that Sodarace, the domain that inspired IESoR, was well-received by the public, which led to thousands of user-created Sodaracer contributions [62]. Given the variety of automated results produced by IESoR in this chapter and the public interest in the Sodarace domain, the IESoR domain provides a compelling initial domain for validating the WIN framework.

CHAPTER 5: WORLDWIDE INFRASTRUCTURE FOR NEUROEVOLUTION

An open problem in the fields of artificial life (ALife) and evolutionary computation (EC) is how to effectively leverage the modern Internet's infrastructure to facilitate and enhance research with user contributions from across the globe [88].

Recall from Chapter 2, a promising technique for crowd-sourcing scientific results across the Internet is collaborative interactive evolution (CIE). Powered by interactive evolution, CIE enables contributions from multiple users over the course of the experiment [86]. In particular, existing CIE systems like Picbreeder [73] and Endless Forms [11] benefit from contributions by multiple users online through a process called branching, whereby users can seed evolution with previously discovered results.

Yet despite the potential benefits of human-computer collaboration through CIE, engineering such a system is still complex and laborious. Notably, Picbreeder alone took several researchers over a year to construct [73].

While existing web technologies provide the path to enable large-scale CIE applications, there are limited community resources or programming libraries to assist researchers in building such evolutionary applications. With the rise of cloud computing and advancements in the development of new web technologies, e.g. HTML5, JavaScript, and Node.js [23], there is an untapped opportunity to create a new set of tools that can enhance the presence of ALife and EC communities online while harnessing the power of humans to aid search on a massive scale.

To address the difficulty of creating online collaborative evolutionary systems, this chapter presents a prototype library called Worldwide Infrastructure for Neuroevolution (WIN) and its accompany-

ing site WIN Online (`http://winark.org/`). The WIN library is a collection of software packages built on top of Node.js that reduce the complexity of creating fully persistent, online, and interactive (or automated) evolutionary platforms around any domain. WIN Online is the public interface for WIN, providing an online collection of domains built with the WIN library that lets novice and expert users browse and meaningfully contribute to ongoing experiments. The long term goal of WIN is to make it trivial to connect any platform to the world, providing both a stream of online users, and archives of data and discoveries for later extension by humans or automated algorithms.

5.1 WIN Library

There are existing efforts in the ALife and EC community aimed at enhancing a communal pool of resources for collecting and sharing experimental results. For example, the ALife Zoo aims to take advantage of cloud computing to host a shared platform for running ALife simulations [39]. Similarly, the Virtual Complexity Lab (VLab) is an online resource that aggregates a variety of ALife simulations to stimulate interest in the field [35].

Together, these platforms represent a significant contribution to the community, helping to encourage advanced simulations and to share improved results. However, there remains an open opportunity in ALife for additional platforms geared towards enhancing the contributions of laymen as well as academics.

There is thus a need for software that goes beyond organizing domains, simulations, or resources in the community. Such software could explicitly minimize the developer effort required to add collaborative interactive features and multi-user support to already existing experimental domains or simulations.

5.1.1 WIN Architecture

To fully address the needs of the community, WIN serves as a minimally invasive library for storing and cataloging evolutionary artifacts to support collaborative interactive evolution. By design, WIN is conceived as both a platform and a service. The platform is a set of libraries and tools to assist in enabling any domain to allow access to its data online by algorithms or users, while the service aggregates a growing collection of ongoing experiments built with the WIN platform. To aid development, the platform is designed to significantly reduce the programming burden of making experimental data available, and provides methods for attaching any domain to the worldwide repository of experiments. On the other hand, the service is the public interface to all the collected domains, allowing interested users to freely browse available experiments linked by the WIN platform. Together, the WIN platform and service aim to amplify the collective effort of the community by reducing the work required to open any search space to both academics and laymen alike.

In more detail, the WIN platform is built in the model of Picbreeder but with an eye towards more general applications. Recall that one of WIN's goals is to integrate easily with *any domain*. As such, to prevent being too unwieldy to gracefully integrate existing domains, WIN's architecture is highly modular. Built on top of the JavaScript library Node.js [23], WIN is a lightweight and expandable collection of Node.js packages that are optionally included for any domain. Keeping the core WIN library minimal but extensible, WIN avoids becoming a one-size-fits-all package. Overall, the WIN platform is a small set of libraries for handling storage, retrieval, and cataloging of complex chains of research artifacts similar to how genotypes are stored and presented by the Picbreeder web service [73].

5.1.2 Modules

It is important to note that WIN is not confined only to organizing research data. To enable more functionality, a driving concept of the platform is the ability to create *WIN modules*, which are event-driven Node.js packages that plug in to the WIN framework. For example, the two domains demonstrated later in this paper extend WIN to include modules for user interface elements and for managing automated evolutionary searches. A benefit of these modules is that they can be reused by future researchers, which is later validated in the next chapter through a developer study conducted with the WIN framework.

Borrowing from the event-driven design paradigm of Node.js, each WIN module specifies the events to which it responds as well as any events required from other modules. The overall WIN framework handles passing these generic messages and events between modules, and routing the responses to the appropriate places. While the exact technical details of how WIN handles message passing are described later in this chapter and available to study within the open-source repository `https://github.com/OptimusLime/win-backbone`, it is important to understand the implications of constructing the framework as a generic message passer.

There are two significant advantages to building WIN as a collection of event-driven modules. Primarily, any researcher in the community can bootstrap their own experimental work by extending any relevant WIN modules. For example, researchers may be interested in modifying the module responsible for handling the genetic encoding for a given experiment. Because each module only responds to a select set of events, augmenting a module does not require understanding the implementation details, but rather a higher level understanding of how to combine those events to add new functionality.

Second, the design enables the concept of module swapping. Due to the structure of Node.js,

modules may request an event response without knowing *a priori* what module will respond. Researchers can take advantage of this feature by building modules that perform the same overall function powered by entirely different algorithms.

5.1.3 Saving Data in WIN

However, simply making WIN modular and event-driven is not a panacea for integrating WIN with any research domain. It is possible for each evolutionary experiment to have a distinct genetic encoding sometimes paired with custom methods to mutate or cross-over genotypes. Furthermore, special algorithms for exploring the search space can create additional domain complexity. Creating software that can support the wide range of existing domains while covering future research directions presents a unique engineering concern.

To directly address these software challenges, the key insight behind WIN's architecture is to be agnostic to the processes generating the data, and instead focus on the form the experiment's data will take. In effect, WIN inverts the typical relationship between experiments and the data produced. As researchers, the primary focus often revolves around what algorithms and encodings produce the collected data. In contrast, WIN is primarily concerned with how the data is structured, which is defined a priori by the researcher for each experiment.

Formally, to maintain data with varying attributes and sizes, WIN enlists the JSON format [21], as well as the JSON Schema specification for data validation [31]. JSON describes a data format for building complex data objects as the composition of a smaller set of universal data structures (e.g. strings, numbers, arrays and dictionaries). Similarly, the JSON Schema definition specifies a template language to describe the structure of JSON data for facilitating data validation. Essentially, each JSON schema outlines a contract describing the format that data can take, which enables an application to validate that incoming data objects match the same format.

Inside WIN, the JSON schema defined by the researcher dictates the internal structure of the data being stored. Before permanent storage, all data being saved by WIN is validated against the expected format. Currently, WIN employs the NoSQL document-database MongoDB for long term storage and retrieval, which is a natural fit for storing JSON data [70]. Note that because WIN utilizes Node.js and JavaScript for its underlying functionality, it is more convenient to select JSON for data formatting over other storage types, e.g. XML. Regardless of the specific storage format, the key required property is the ability to represent a wide variety of data configurations, which enhances the ease of use when integrating new domains onto the platform.

Importantly, WIN can store research artifacts while remaining encoding-agnostic. Figure 5.1 shows how different genetic encodings can be represented as simple JSON schema, all of which can be tracked and saved by the WIN framework. By default, a JSON schema is required for saving objects in WIN, but the schema specification is a simple and extensible template for saving any type of data. Interestingly, by design of the JSON format, data with almost any conceivable form can be stored by WIN, potentially opening the platform to support most genetic encodings actively researched by the community.

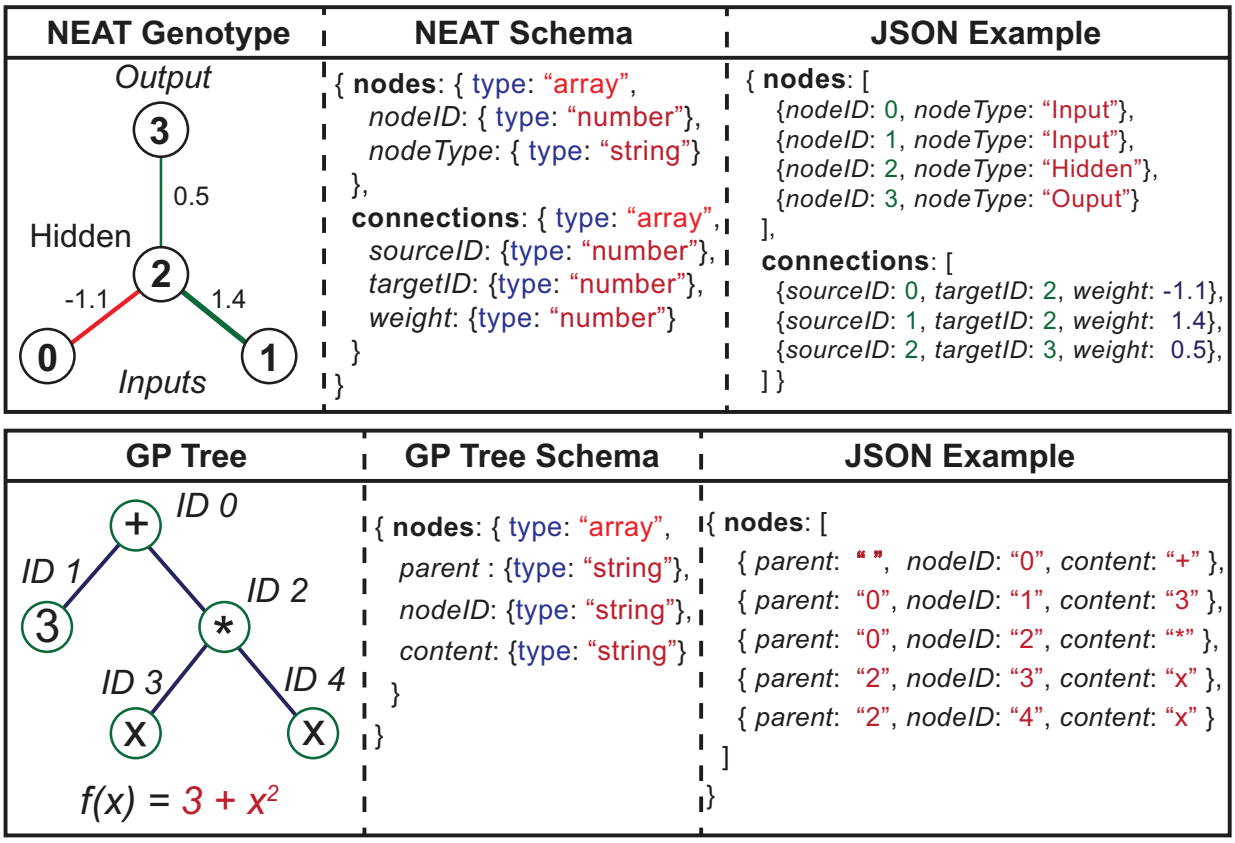


Figure 5.1: **Example Schema in WIN.** Shown here are examples of two potential encodings and the corresponding JSON format for saving inside WIN. At top, a NEAT Genotype describes a *compositional pattern producing network* (CPPN) with four nodes and three connections [82]. Below, a GP-Tree [53] representing the function $f(x) = 3 + x^2$ is shown. For both figures, the middle column describes the expected composition of the data sent to WIN for the purpose of validation.

5.1.4 Phylogenies

Ultimately, when collecting research artifacts from evolutionary domains, the relationships among the data can tell an important story about how a domain solution was discovered. In evolutionary computation, experimental data commonly has a parent-child relationship, wherein one object is considered the direct descendant of another. By chaining a collection of objects and their relatives, an artificial phylogeny can be constructed. Previously, Woolley and Stanley [91] investigated the Picbreeder phylogeny to understand why fitness-based automated evolution was having significant difficulty attempting to recreate images already evolved through interactive evolution by the users of Picbreeder.

Crucially, the Picbreeder phylogeny is likely not the only phylogeny capable of informing scientific research, but it is the only phylogeny available online. Of all the previous evolutionary experiments conducted, representing thousands of published papers, the lost phylogenies of those experiments may have contained potential treasure troves of information.

To account for this potential, WIN not only stores artifacts created by evolution, but simultaneously tracks the relationships among the data as well. Practically, tracking relationships in the data thereby requires a minimal amount of additional work by the researcher. By default, WIN attaches a unique identifier to each object saved internally. Therefore, to enable tracking connections in the data, each research artifact being saved must provide an accompanying list of identifiers representing the object's parents. This additional parental list is enough to create a map of the ancestry across all artifacts.

Notably, preserving artifact lineages is an important prerequisite for augmenting any evolutionary domain with collaborative interactive evolution support. The processes of branching and publishing, a critical underpinning of CIE experiments like Picbreeder [73] and Endless Forms [11],

rely on the historical relationships among artifacts. Without maintaining such relationships, there would be no way to contextualize the progression of the search during the experiment.

5.2 WIN Technical Details

The modular scaffolding of WIN described in the last section is a core design principle of the framework. Thus its technical details are helpful for assisting developers to utilize WIN to create CIE applications. To understand how the modules of WIN work together, figures 5.2 to 5.5 depict several WIN modules working together to power a generic CIE experiment in a sample domain. This depiction illuminates how the modules of WIN can support a broader collection of experimental domains, while the next section will detail two example CIE domains to demonstrate the WIN framework in practice.

For clarity, all figures separate the local and global components of the CIE experiment because WIN is structured as a client-server architecture. The local components (shown on the left of each figure), i.e. the client, are executed on the end user's computer, while the global components of WIN (show on the right of each figure) are run on dedicated servers. Note that developers employing WIN must host their own dedicated servers with the WIN framework installed.

Broadly, the local components of WIN enable interactive evolution on the client, while providing the appropriate mechanics to send and receive properly structured data to the WIN server. The primary purpose of the global components of WIN are to preserve, organize, and retrieve evolutionary artifacts.

To illustrate WIN functionality, figures 5.2 to 5.5 review the set of user-driven actions a typical CIE experiment, e.g. Picbreeder [73] or Endless Forms [11], supports, including local interactive evolution, publishing artifacts, a home screen of recently published artifacts, and displaying the

genetic relationships among artifacts (phylogenies).

More specifically, figures 5.2 and 5.3 show how the WIN framework supports interactive evolution on the client. At the heart of each CIE experiment is the interactive evolution client driving the search for new artifacts. During interactive evolution, WIN employs three local modules to assist in the creation of new artifacts: win-iec, win-gen, and win-NEAT.

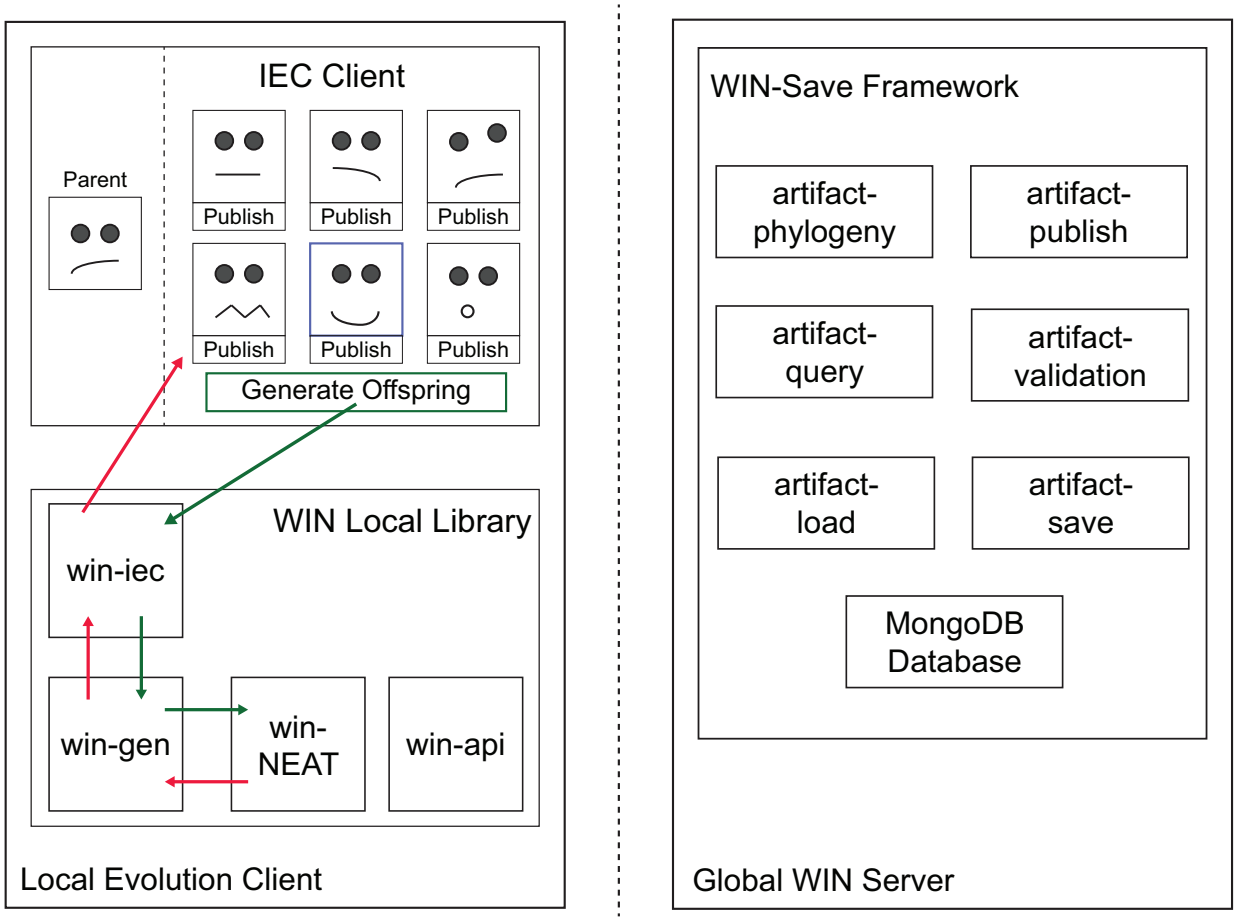


Figure 5.2: **Generating Artifacts in WIN.** Shown here is a high-level overview of how WIN modules work together to generate new offspring within a CIE experiment. Creating offspring happens on the local client, and there is no need for this purpose to contact the WIN server depicted on the right of the figure.

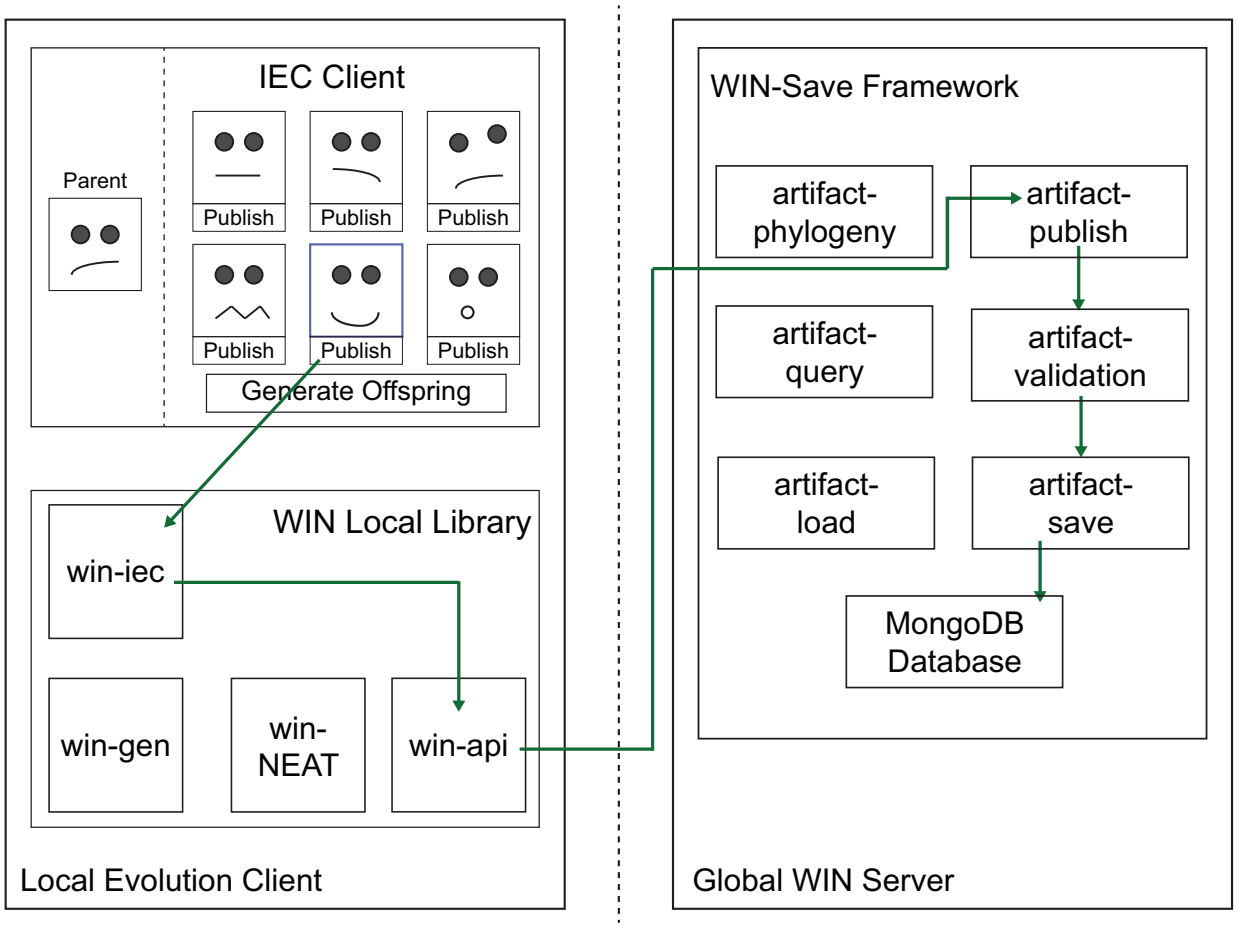


Figure 5.3: **Publishing Artifacts in WIN.** Above, the flow of data through the WIN framework depicts how CIE experiments built with WIN can utilize the framework to publish and save artifacts generated during interactive evolution. In this example, the user requests to publish an artifact, which sends a message through the local win-iec and win-api modules before being handled by the global WIN server.

The win-iec module supports basic interactive evolution with mechanisms for selecting/de-selecting parent artifacts and triggering the creation of new offspring from the selected parents. Importantly, the win-iec module is agnostic to the objects stored internally, and outsources the generation of

offspring to the win-gen module. The win-gen module aids in creating new offspring matching a predefined data schema, offloading the specifics of asexual or sexual reproduction to the relevant encoding module. In the example domain within figures 5.2 to 5.5, the encoding module is win-NEAT, which handles neural network encodings (e.g. CPPNs; [82]).

In detail, figure 5.2 shows the flow of data through the local WIN framework when the user clicks the "Generate Offspring" button, where the green arrow indicates the forward flow of WIN messages and the red arrows indicate the return of data back to the user interface (UI). First, the win-iec module receives a message from the UI requesting new offspring. From there, the selected parents tracked by the win-iec module are sent to the win-gen module, which according to the schema structure will relay the relevant genomes to the win-NEAT module for handling NEAT-specific offspring creation.

Although the module structure may appear to add complexity to the local WIN framework, its design confers a number of desirable benefits to WIN developers. In particular, for developers to swap the encoding within a CIE experiment built with WIN, there is only a single module that must be modified: win-NEAT. For example, developers interested in experiments with genetic programming (GP) encodings would only need to create a win-GP module to handle offspring creation, while the remaining interactive evolution structure remains unchanged.

For user-driven actions that require contacting the global server, the local win-api module handles the logistics of formatting the server requests. As shown in figure 5.3, when the user chooses to publish an artifact, a message is passed to the win-iec module, which in turn forwards the relevant artifact information to the win-api module. From there, an HTTP request is sent to the global WIN server with the attached artifact data.

Depending on the server HTTP endpoint, an internal message on the server is generated and the data is directed to the appropriate modules. When publishing artifacts, there are four important

modules: win-publish, win-validation, artifact-save, and the MongoDB database. Accordingly, figure 5.3 shows the flow of data into the database with a green arrow. The only data returned to the local client is whether the artifact is successfully published.

The win-publish module on the server receives requests for publishing, and organizes the data into batches for processing. The batch of artifacts is then sent to the win-validation module that ensures the data being sent strictly matches the predefined JSON schema (Section 5.1.3 details the schema). The win-validation module breaks each object into its constituent JSON components, and verifies that the data matches the expected type for each individual JSON component. Additionally, the win-validation module checks and removes any artifacts that are duplicates in the database. After validation, the artifacts are sent to the artifact-save module that converts the JSON information into MongoDB objects according to the JSON schema. Finally, the MongoDB objects are asynchronously saved to the database.

The data-driven design within the WIN server means that the same publishing framework can be utilized regardless of the schema structure. That is, the only change in program structure for the server-side of WIN is to define the artifact schema for the evolutionary experiment. In fact, the image breeding and creature-evolution domains built with WIN in the next section employ the exact same server-side program despite being distinctly different evolutionary experiments.

To enable functionality similar to Picbreeder [73], the WIN framework also supports a "home screen," of recently published artifacts (as shown in figure 5.4). When the home screen is loaded, a message is sent to the local win-api module to fetch recently published artifacts from the WIN server. From there, three WIN modules work together on the server-side to load the relevant artifacts: artifact-query, artifact-load, and the MongoDB database. The flow of data from the local client to the global server and back is shown through the green and red arrows, where green signifies the movement of requests and red the movement of responses through the framework.

When the HTTP request is sent to the WIN server, the artifact-query module responds by passing a message to the artifact-load module. The artifact-load module is responsible for formatting a request in the proper MongoDB query language and structure, taking into account the schema defined by the experimenter. After the artifacts are queried in the database, the JSON data is returned to the local client and then displayed on the home screen UI.

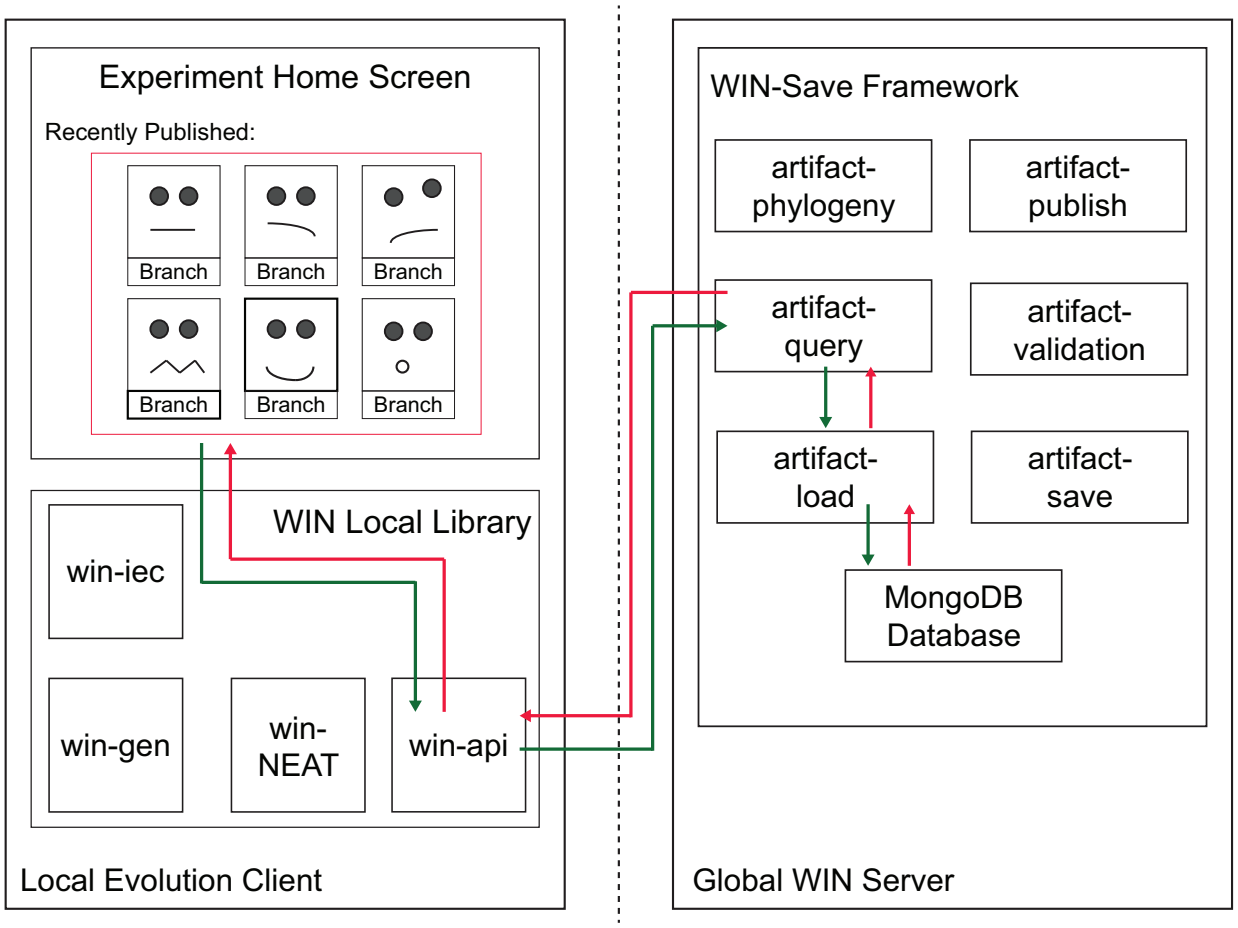


Figure 5.4: **Example WIN Home Screen.** Similar to the Picbreeder [73] experiment, CIE applications built with WIN support a "home screen" of recently discovered artifacts. To fetch the correct information, the client sends an HTTP request through the local win-api module, which triggers a series of messages on the global WIN server. The result is a collection of artifacts that are sorted by date published that are then displayed on the user interface.

To help illuminate the benefits of the modular structure on the WIN server, figure 5.5 portrays a server request to display a portion of the artificial phylogeny for the CIE experiment. These artificial phylogenies, like the ones discovered within Picbreeder and in the two domains later in

this chapter, represent a key functionality of CIE experiments, and are visual representations of the connections and relationships among the artifacts. Interestingly, the flow of data to the server to recover the artificial phylogeny is almost identical to the flow of data for retrieving recently published artifacts. However, to recover the phylogeny, an initial artifact identifier is sent to the WIN server, and the artifact-phylogeny module recursively requests the children of each artifact up to a certain depth. In this way, the phylogeny is unraveled level by level through the WIN framework. The next section demonstrates all four of these WIN mechanics in practice within functioning CIE experiments.

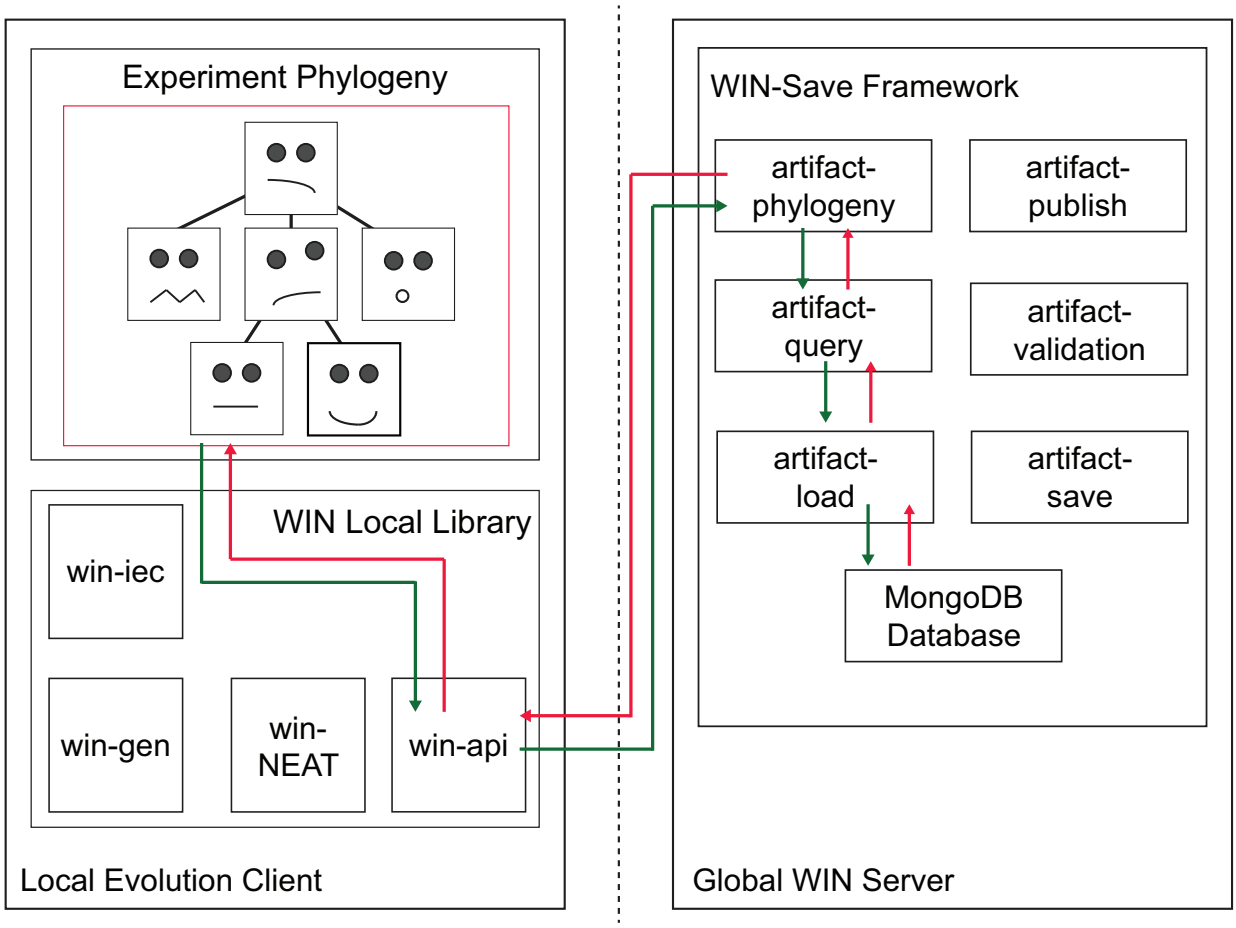
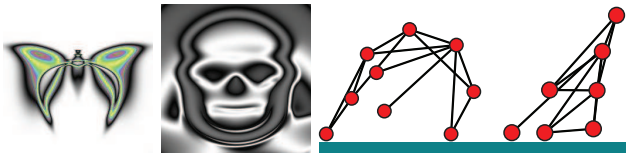


Figure 5.5: **Displaying WIN Phylogeny.** The flow of data is shown through the local and global WIN framework to return the artificial phylogenies stored within the MongoDB database powering the CIE experiment. Similarly to figure 5.4, the artifact-query and artifact-load modules are critical in returning properly structured artifacts, which are then organized for display to the end user.

5.3 WIN Domains

Beyond software functionality, the key to enticing a community to develop for a platform like WIN is to show that new domains can be added easily and systematically. By demonstrating this point through two domains that would otherwise be highly challenging to put online without WIN, this section (and its corresponding demonstrations online) not only shows the possibilities that WIN creates but also provides a reference for future developers aiming to extend WIN with more domains.

The first is an HTML/JavaScript clone of Picbreeder [73]. The second is IESoR, the Sodarace-inspired [63] domain for evolving two-dimensional morphologies capable of ambulation [83] described in the previous chapter. Some resulting phenotypes from both Picbreeder and IESoR are shown in figure 5.6. To avoid confusion, the version of Picbreeder integrated with WIN will be called win-Picbreeder, and the IESoR variant will be called win-IESoR. Both examples are currently accessible through WIN Online at `http://winark.org/`. Together, the Picbreeder and IESoR domains aim to cover a broad range of interests inside the ALife community. Picbreeder is a proxy for domains that are more suited towards Interactive Evolutionary Computation (IEC) [87] and difficult to optimize. In contrast, IESoR is a control-based domain that runs multiobjective search to discover new ambulatory creatures, i.e. it is designed for automated optimization.



(a) Picbreeder (b) IESoR

Figure 5.6: **Products of Evolution.** Picbreeder artifacts (a) are n x n pixel images where the RGB values are constructed from the outputs of a CPPN [82]. (b) A pair of two-dimensional ambulating creatures are shown from the IESoR domain [83].

To accommodate the two example domains, several new WIN modules were built to augment the features described in the *WIN Architecture* section. Importantly, both Picbreeder and IESoR depend on two new modules, win-home and win-gen. Win-home is a user interface (UI) module that mimics the web portion of Picbreeder, which is intended to provide a simple user interface for domains hosted on WIN. The module includes a homepage template similar to Picbreeder for displaying new or interesting artifacts published by users. The win-gen module aids in creating new objects matching a predefined data format and in particular for generating the underlying indirect encoding (CPPNs) powering both domains [82]. An exhaustive list of modules created so far is available online at `http://winark.org/modules`, and the exact programming details of these modules are all accessible and open-source. Developers who are new to WIN will have the benefit of the already-included domains and modules built for both demonstrations that thereby serve as templates for new experiments in WIN.

5.3.1 Picbreeder

Originally, the Picbreeder website was written in PHP, while the Picbreeder evolution client responsible for generating the images was written in Java. For a more modern approach, win-Picbreeder is written in HTML5 and JavaScript, which has the added benefit of running in any browser without plugins. Because WIN is conceived in the mold of Picbreeder, it follows that win-Picbreeder is among the easiest projects to integrate with WIN.

To match the major features of the original Picbreeder, win-Picbreeder needs a homepage, an IEC user interface, and the ability to store, retrieve, and generate image artifacts. As described in section 5.1.3, WIN offers an encoding-agnostic method to store, retrieve, and generate custom schema. In the case of Picbreeder, the schema simply contains a NEAT genotype, i.e. the CPPN, conforming to the structure depicted in figure 5.1a, along with user tags describing the final phenotype image in a few keywords. Therefore, the main issue for creating win-Picbreeder is how to connect the IEC user interface to the existing WIN framework for saving artifacts.

In WIN, the solution is fairly simple. Recall that WIN's main architecture is event-driven, and the process of generating and saving artifacts is displayed for a generic experiment in figures 5.2 and 5.3. Similarly, as the user explores the domain through the IEC interface in win-Picbreeder, whenever an artifact must be displayed, an event is passed to the local win-iec module to generate a new artifact from the currently selected parents. WIN routes this message to the win-gen module described above that is responsible for creating new artifacts, and a new win-NEAT module creates NEAT genotypes and the corresponding CPPNs by combining the provided parent genotypes. After the IEC interface receives the new artifact object(s), domain-specific Picbreeder code converts the NEAT genotype in the artifact to the displayed picture in the interface through WebGL (an HTML rendering framework found within most modern browsers).

The majority of complexity in Picbreeder is in the storage and retrieval of evolutionary data. Because WIN is designed specifically to handle the data management task, the rest of the win-Picbreeder application is lightweight compared to the original Picbreeder's code. Altogether, win-Picbreeder effectively demonstrates a simple WIN application; the code is available online at `https://github.com/OptimusLime/win-Picbreeder`.

The exciting point about adding this domain is that numerous IEC-based domains can easily be constructed and put online simply by deriving them from the win-Picbreeder code. That is, with minimal effort researchers can create services like Genetic-Programming-Picbreeder, L-Systems-Picbreeder, or any such conceivable variant. In fact, in a developer study detailed in Chapter 6, participants managed to create and launch several win-Picbreeder variant services within four hours.

5.3.2 IESoR

Recall from Chapter 4 that in the original IESoR, an automated NSGA-II multiobjective search [26] evolved functional two-dimensional ambulating morphologies similar to the creatures depicted in figure 5.6b. In the win-IESoR application, the same multiobjective search algorithm operates with one important difference: the human user is included in the loop. Instead of running an automated algorithm for a fixed period of time and collecting the results, win-IESoR interleaves occasional choices by the user with shortened multiobjective searches and returns the most promising individuals from which the user can further evolve. Woolley and Stanley [92] demonstrated recently the ability of human choices interleaved with search to outperform even automated algorithms. WIN is uniquely positioned to make this kind of interleaved search easy to integrate into any domain.

Because interactive evolution was not part of the original IESoR domain, it was not necessary to build win-IESoR with user interaction. However, explicitly including user interaction with win-

IESoR helps to highlight a deeper purpose behind the infrastructure of WIN. If humans have the ability to add insight and utility to search and WIN makes the process of including a human in the loop relatively painless, then the hope is that researchers in the future will not need to hesitate to take advantage of human insight and collaborative functionality whenever it is appropriate.

To enable user interaction with an automated search, a new local WIN module named win-NSGA was created to handle the complexity of this interleaving search. Note that the win-NSGA module replaces the win-iec module displayed in figures 5.2 to 5.5, and handles automated evolution. Following precedent in Woolley and Stanley [92], after a certain number of viable candidates are found through automated evolution, the search returns the results to the user interface for human selection. The main takeaway is that win-IESoR demonstrates that the WIN platform is capable of executing a search process that involves both an automated algorithm and a human.

In win-IESoR, the data saved by WIN includes additional components beyond those saved by win-Picbreeder. The data contains both a NEAT genotype to define the creature morphology as well as the parameters and objects inside of the two-dimensional physics engine (e.g. the ground, gravity, and friction). Along with the win-home user interface templates, a new UI module for interleaving search was created inspired by the user interface elements from Woolley and Stanley [92]. While the interface for interactive evolution in win-IESoR is still under construction, the results of an initial set of evolutionary experiments within win-IESoR are browsable through the WIN Online service.

A major benefit of WIN is that there is no need to write and re-write the same code when someone in the community has already written it. Thus all the infrastructure written for win-IESoR that allows integrating multiobjective searches with interleaved human selection can now be applied to any other ALife domain, which should help to accelerate research in such systems significantly.

5.4 Artificial Life Community: WIN Online

While the WIN platform is designed to be minimally intrusive, WIN as a service, or WIN Online, aims for a larger role in the ALife and EC community. Accordingly, WIN Online is the public face to a worldwide repository of ongoing experiments for any evolutionary or ALife domain integrated with the WIN platform. Recall that the data saved by WIN represents potential starting points for new interactive or automated searches. Thus WIN Online becomes a place where users who are interested in the ALife and EC communities can participate in academic domains without prerequisite domain knowledge and immediately start a fresh evolutionary branch from existing results. The two examples demonstrated in the next section provide an example of what the WIN Online user experience is like for domains that operate both in and out of the browser.

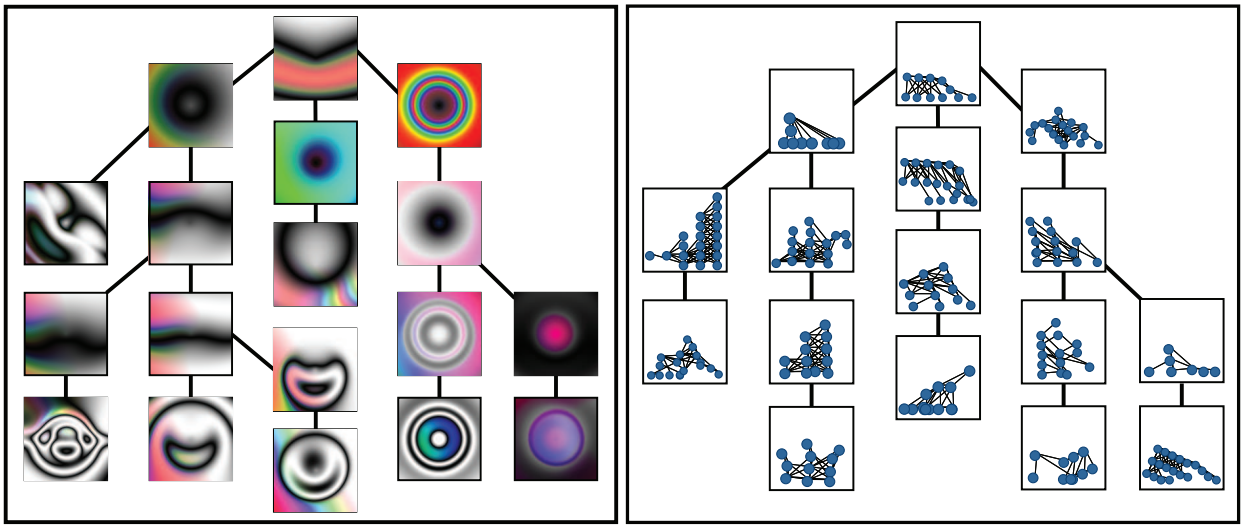
For researchers, WIN Online acts as a potential resource for attracting users and crowd-sourcing new domain solutions. Note that researchers who prefer not to allow WIN Online to host their research data may host data on their own servers and supply an external link for WIN Online. To assist in creating an online collection of experiments, any domain built with WIN can integrate into the online repository with minimal additional effort. As discussed in 5.1.3, internally, WIN utilizes the MongoDB database for storage. Taking advantage of a MongoDB feature, domains linked to WIN Online are given a separate database within the global MongoDB database hosted by WIN Online. Additionally, WIN Online handles configuration of the REST API required to access the newly created domain database and its contents.

Utilizing these features, WIN Online can provide a catalog of cutting edge ALife research with the potential to provide a steady stream of users to new research domains. A prototype for WIN Online that includes links to the two domains described next is accessible at `http://winark.org/`.

5.4.1 WIN Phylogenies

An important functionality included within win-Picbreeder and win-IESoR is the ability for the user to decide when an artifact should be saved for the public record, i.e. published. Recall that WIN always tracks parent-child relationships among published artifacts. As users interact with a domain, each published object adds a single branch to the tree of artifacts maintained within the database. The exact publishing process via the WIN framework is depicted in figure 5.3, the same flow of data that occurs within win-Picbreeder and win-IESoR. At any point during the ongoing experiment, researchers may compile part or all of the published objects into an artificial phylogeny. With the help of the artifact-phylogeny module on the WIN server (shown in figure 5.5), WIN assists in constructing phylogenies by providing methods for retrieving the full parent and children database elements for any artifact within the database. Researchers can repeatedly query these methods to unravel a chain of research artifacts and construct a phylogeny of any depth.

To highlight this WIN feature, phylogenies generated with the artifact-phylogeny module for win-Picbreeder and win-IESoR are shown in figure 5.7a and 5.7b, respectively. For both applications, the data represent a cross-section of artifacts generated by a single researcher. It is important to note that these artificial phylogenies do not represent static aggregated results, but rather collections of potential stepping stones that are all available to branch from currently in the ongoing experiments. Both win-Picbreeder and win-IESoR are open for browsing artifacts through WIN Online at `http://winark.org/`. Win-Picbreeder already supports contributing new artifacts, while win-IESoR is a prototype that allows browsing artifacts generated during initial experiments.



(a) win-Picbreeder (b) win-IESoR

Figure 5.7: **WIN Phylogenies.** The tree of artifacts in (a) and (b) represent artificial phylogenies resulting from the efforts of a single researcher. In (a), each square represents a published image inside win-Picbreeder, while each connection represents a direct relationship between the images. Each image in (b) illustrates the starting morphology of a two-dimensional ambulating creature evolved by win-IESoR. Though images are linked by a single connection, there may have been multiple human selections and potentially hundreds or thousands of automated evaluations in each branch of the tree. Both phylogenies contain artifacts that are currently available for browsing in win-Picbreeder and win-IESoR by visiting WIN Online at `http://winark.org/`.

5.5 Implications

Building a tool that can organize and unlock the data within any arbitrary evolutionary domain is ambitious, but the WIN prototype hints at how it becomes possible. The win-Picbreeder and win-IESoR domains represent a milestone on the path towards building an infrastructure to give

researchers from evolutionary computation and artificial life the ability to quickly make their research available to users online, who can then genuinely contribute.

Moreover, this initial WIN prototype forms one part of the larger framework constructed within this dissertation that includes support for both web browsers and mobile devices (e.g. smart phones and tablets). To help achieve goals for collaborative open science set forth by the ALife community [88], the lightweight design of the WIN framework aspires to create a new type of research collection full of accessible domains assembled together for easy and perpetual access across the world via WIN Online.

CHAPTER 6: WIN HUMAN STUDY

The previous chapter demonstrated the technological feasibility of WIN as a platform creation tool for multiple evolutionary domains, win-Picbreeder and win-IESoR. However, a key motivation for WIN is to enable researchers to build rapidly upon previous efforts in the community to create new collaborative interactive evolution (CIE) systems. Thus to evaluate WIN's ability to facilitate the creation of new CIE systems from old ones, this chapter describes a survey of three developers who each utilized WIN to construct new win-Picbreeder variations (available online at `http://winark.org/variants`). The results reveal strengths in WIN's ease of use and quick deployment, while also uncovering weaker elements of WIN such as gaps in the documentation, which have been addressed directly with improvements in the library.

6.1 Survey Design

To begin to study the capabilities of WIN as a platform builder, it is helpful to analyze how developers interact with the library in practice. Recall from chapter 5 that the WIN design emphasizes a lightweight implementation for minimal burden on potential developers. Thus the aim of this study is to validate that the WIN library enables quick creation of CIE variants while also providing a qualitative understanding of how the library is most naturally utilized by developers.

The study was conducted with three researchers from the Evolutionary Complexity Research Group at University of Central Florida (UCF) that all had prior exposure to both neuroevolution techniques and the original Picbreeder service. Each participant was instructed to download the open-source win-Picreeder repository from GitHub (`https://github.com/OptimusLime/win-Picbreeder`) and make a change to the win-Picbreeder code to create a new variant ser-

vice. A small tutorial for installing and launching win-Picbreeder was also provided. Participants were asked several quantitative questions to numerically assess WIN on its ease of setup, modification, and deployment, including the clarity of the code and the tutorials. The survey concluded with several more open-ended questions about strengths and weaknesses of the experience building with WIN.

6.2 Results

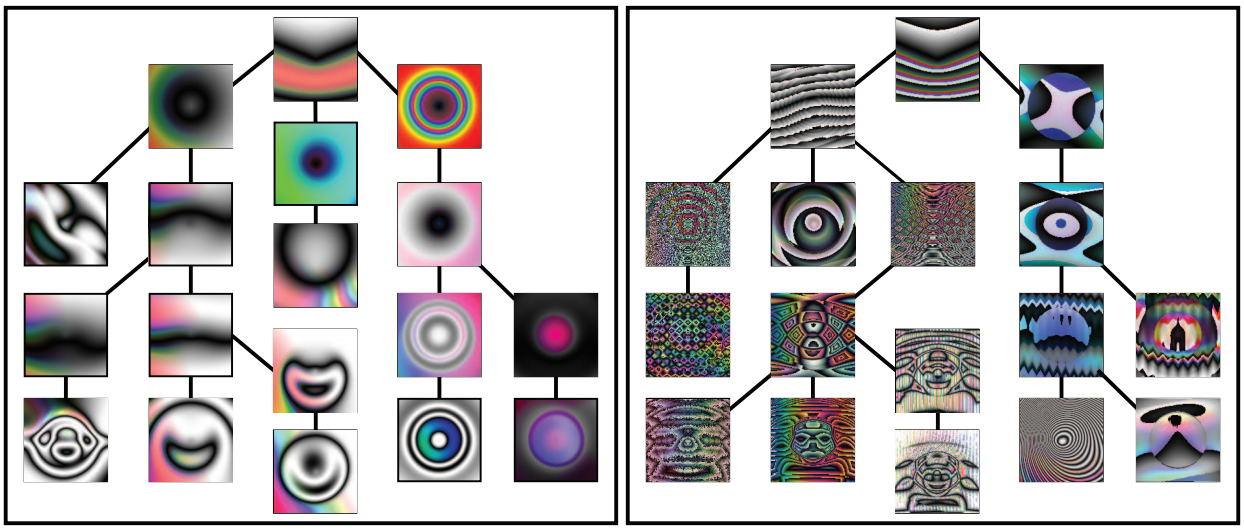
Quantitative results suggest that the WIN framework is rated highly for the software's ability to be easily set up, modified, and deployed for the purpose of building CIE variations from existing application code. Though one participant ran into some difficulty initializing WIN and another had an issue with documentation clarity, both participant concerns are addressed through modifications to the framework detailed in the next section. The full set of quantitative survey questions and answers are shown in table 6.1.

Table 6.1: **Perceived Quality of WIN by Survey Participants**. This table shows the average ratings for five quantitative assessments of the WIN library from ease of use to quality of the documentation, where each score is out of 10 and higher values indicate a better experience with the WIN library.

Survey Questions	**Dev 1**	**Dev 2**	**Dev 3**	**Mean**
Rate ease of initial setup of the WIN interactive evolution system	6	8	9	7.67
Rate ease of modification of your WIN interactive evolution system	9	7	8	8.00
Rate ease of local deployment of your WIN interactive evolution system	7	10	9	8.67
Rate the clarity of the existing WIN code in helping you build your service	8	7	9	8.00
Rate the quality of the WIN tutorials in helping you get started	10	5	9	8.00

Participant responses to the qualitative questions indicate that all three participants could create win-Picbreeder variations in less than four hours even without having any previous exposure to the WIN library or the win-Picbreeder code. One participant chose to adjust how the win-Picbreeder service performs selection and crossover during interactive evolution. The other study participants created win-Picbreeder variations that altered the activation functions inside the compositional pattern producing networks (CPPNs) generating the win-Picbreeder images. One such participant-created variant replaced almost all the traditional activation functions found within CPPNs with "triangle wave, square wave, and two sizes of sawtooth waves in order to produce psychedelic

art." Figure 6.1 depicts the phylogenies tracked by WIN for both win-Picbreeder and the variant with the altered activation functions. The resulting images discovered in the variant service are distinctly different than the original win-Picbreeder discoveries described in chapter 5.



(a) win-Picbreeder (b) win-Picbreeder-variant

Figure 6.1: **Dual WIN Phylogenies.** The tree of artifacts in (a) and (b) represent artificial phylogenies resulting from the efforts of a single researcher. In (a), each square represents a published image inside win-Picbreeder, while each connection represents a direct relationship between the images. Each image in (b) illustrates a published image inside a win-Picbreeder variant with altered activation functions developed by a survey participant. Both phylogenies contain artifacts that are currently available for evolving in win-Picbreeder at `http://winark.org/apps/win-Picbreeder/` and the win-Picbreeder variant at `http://winark.org/apps/win-Picbreeder-g/`.

Note that while the sample size of the survey is small, these results highlight an essential design goal of WIN wherein developers do not need to understand the more complex server-side elements

of WIN to still gain the benefits of being able to create online interactive experiments. That is, the survey confirms that future practitioners of WIN need not be concerned with all aspects of the platform, but instead only the particular pieces that are directly relevant to their research. As one participant notes, "With WIN, it is not necessary to have a working knowledge of all components of such a system. Evolutionary researchers, for example, could answer evolution-related questions without having to know much about working with databases or interfacing with users through web pages."

Notably, while two participants estimated that creating the same variant service from scratch could take anywhere from "at least 40-80 hours" to "several hundred hours," one participants suggests that the experience with WIN had positively impacted the participant's thoughts about creating CIE experiments, remarking "While I have not previously considered using interactive evolution in any projects previously, the ease that WIN offers for implementing such a system may encourage me to consider that option in the future."

Importantly, the results validate the idea that a single open-source domain built with WIN can proliferate many new variations that are all valid CIE experiments. Specifically, the survey demonstrates the ability of three researchers with little exposure to the original program to create three win-Picbreeder variants in several hours each. Interestingly, to put this result in perspective, despite the seemingly small number of participants involved in this experiment, the new services created in this experiment increase the number of available CIE experiments online by 60%, from five to eight at the time of writing including Picbreeder [73], win-Picbreeder (chapter 5), Endless Forms [11], Petalz [72], and BrainCrafter [69]).

6.3 Modifications

The survey responses also reveal areas for improving the usability of the WIN library. Examining table 6.1, while the average rating for the ease of initial setup and quality of documentation for WIN is high, in each category there is a developer who rated the service lower than other participants, likely indicating a specific issue encountered with WIN. For example, one developer noted that only a small portion of time was required to make the win-Picbreeder changes, yet the initial setup involved "additional hours spent working through dependency issues that were eventually corrected by the software's developer [Paul Szerlip]." Though the library was designed to be cross platform, this developer uncovered support issues for Windows that have since been corrected.

Another developer did not understand how the WIN platform was initialized and required assistance refreshing the database elements of WIN after making changes. This frustration with the documentation is likely reflected in the lowest score given to the quality rating of the WIN documentation. Both issues yielded practical and clarifying add-ons to the documentation of the WIN library as well as fixes for the build tools of WIN.

6.4 Discussion

Notably, traditional scientific libraries like ECJ [59] and SFERES [65] allow researchers to create multiple experiments with small algorithmic differences to produce a set of publishable scientific results, e.g. comparing different search algorithms like novelty search and MAP-Elites [66]. Similarly, the survey conducted with WIN demonstrates the ability to quickly produce multiple win-Picbreeder variants all with small algorithmic modifications, yielding significant differences in the artificial phylogeny of images observed in figure 6.1. In effect, WIN enables the fundamental building blocks for scientific exploration even for CIE experiments that have previously taken

multiple researchers considerable effort to construct. With WIN, each open-source CIE application published represents a new stepping stone to an entire collection of easily created variant services.

Fundamentally, WIN's support for multiple evolutionary domains demonstrated in chapter 5 and the successful construction of several win-Picbreeder variations revealed in this chapter validate the WIN library as an effective research tool to aid developers in significantly expanding scientific crowd-sourcing efforts within the field of evolutionary computation.

CHAPTER 7: WIN FOR ANDROID

In both chapters 5 and 6, prototypical domains and extensions based on WIN successfully demonstrated the promise of the platform in the web browser. Designing for the browser enabled anyone with a computer and an Internet connection to potentially access WIN Online. However, a valid question remains about how the WIN platform can handle domains not programmed in JavaScript, and how well the platform is suited for more comprehensive applications that go beyond research prototypes.

To directly address these concerns, this chapter discusses the development of *#filters*, a complete WIN application built for Android, the largest mobile operating system in the world [49]. The application, #filters, allows users to take photos and apply custom image filters discovered during interactive evolution, much like the filters provided in the popular mobile application Instagram[1]. Surprisingly, #filters represents the very first collaborative interactive evolution (CIE) application available on any mobile platform (Android, iOS, or Windows), which collectively represent millions of mobile applications. Without WIN, constructing such a unique CIE platform would be prohibitively complex for a single programmer.

The next section reviews previous evolutionary (non-CIE) efforts on mobile devices. Then, the following section addresses the image filtering domain and corresponding encoding within #filters. An overview of the adjustments required to make WIN effective in a mobile environment are briefly examined. Finally, the chapter ends by discussing the aggregated results from the #filters, as well as the impact and potential future work for #filters.

[1]Copyright Facebook 2015

7.1 Background: Evolution on Mobile

Excitingly, shifting WIN to mobile opens up entirely new paradigms for both interactive evolution and artifact discovery. Previously, win-Picbreeder and win-IESoR were designed principally for desktops, which allowed the experiments to take advantage of much larger screen resolutions. For example, both the original Picbreeder and win-Picbreeder could rely on large screen spaces to display grids of images from which the user can select quickly. In contrast, the limited screen resolutions present a significant problem for search and discovery on mobile.

Notably, the constraints imposed by mobile are not simply a software engineering problem, but more deeply a research question for the field. Across the world smartphones are becoming increasingly common, in both developed and developing countries [34]. In 2014, three in four people in the US owned a smartphone, and the trend is towards even higher ownership percentages [18]. Moreover, a majority of the top applications across all mobile operating systems were social media applications, implying mobile users are concerned about applications structured around their social networks. At the time of writing, there are only a small handful of mobile applications that employ evolutionary computation (EC) or artificial life (ALife) at all, none of which were integrated with any social network support [19, 30, 48]. Worse, there is a distinct lack of automated or interactive evolutionary research being conducted on any mobile operating system.

Given both the trend of smartphone proliferation and the dearth of such research, there is a significant risk to the future relevance of both EC and ALife on an increasingly important and popular computing platform. In direct contrast, as noted in chapter 3, interactive evolutionary computation (IEC) can reduce complex domains to a simple process of selecting the user's favorite choices. That is, the simplicity and personalization of IEC makes it a natural fit for a mobile environment where most users engage with personalized applications for shorter time frames [18]. Despite the potential of IEC applications, with little to no community infrastructure that can support multiple

users across many devices, EC and ALife have a significant uphill battle to stay relevant on mobile devices.

Fortunately, the WIN framework presents a unique opportunity to address the very core of this research problem on Android. Unlike other EC libraries, WIN is built to asynchronously aggregate and amplify the efforts of multiple users by default, a necessity for any connected mobile application. Thus, by building the fully functional Android WIN application #filters, the WIN library simultaneously demonstrates the robustness of the WIN platform across multiple programming languages and provides a pathway for other researchers to create a similar caliber of connected evolutionary applications on Android. In effect, #filters helps unlock new low hanging fruit in an underdeveloped research direction.

7.2 Evolution of Image Filters

To capture a mobile audience, a research domain with public appeal is helpful for demonstrating the potential of CIE on mobile. One such domain is image filtering. Digital image processing has taken many forms in both the scientific community and the wider consumer culture over the last few decades. Traditionally, the computer vision community has employed image processing techniques to aid in manipulating image or video streams for the purposes of e.g. edge detection or other high-level feature decomposition [33]. More recently, the success of consumer technologies like Instagram have demonstrated significant public interest in image filtering for aesthetic purposes.

Within Instagram, users can capture an image with their mobile device's camera and apply a series of photo effects, or filters, to adjust the original image before sharing the altered image on their preferred social network. In contrast to academic research, the image effects produced within Instagram must satisfy the personal tastes of users instead of solving a broader computer vision

task. Instagram thus faces the challenge of defining a finite set of image filters that can appeal to a wide variety of user preferences, a difficult obstacle for a service with over three hundred million users [29].

Interestingly, evolution may be a more effective tool for collecting diverse image filters. Previous experiments in EC (both automated and interactive) have demonstrated that the creation of a multitude of image filters is indeed feasible [17, 37, 61]. However, these experiments all employed evolution to reach a specific type of filter that solved an underlying computer vision task, e.g. noise reduction [37], image enhancement [17], or removing facial blemishes [61]. Importantly, as is the case for user preferences, there is no definitive optimal image filter, i.e. it is difficult or impossible to define the concrete value of an image filter in the highly subjective domain of aesthetic tastes.

Yet Picbreeder's effectiveness in generating thousands of varied two-dimensional images with CIE suggests that image filtering may be an attractive test case for the WIN library. Curiously, the original Picbreeder revealed that while there was a core of committed users, a majority of visitors to the site were not dedicated to evolving content [73]. However, if the content being produced had been more personal, e.g. a filter applied to a user's favorite image, Picbreeder may have drawn an audience more interested in evolution. Thus, an image filtering application for Android simultaneously demonstrates the potential value of CIE to the general public, while addressing the robustness of the WIN platform in a domain that is valuable to a broader consumer community. That is, image filtering on mobile can serve as a killer application for WIN.

Therefore, in an effort to maximally engage users of #filters, the application focuses on personalizing evolutionary content by encoding plausible image effects with compositional pattern producing networks (CPPNs; [82]) and more fine-grained evolutionary control via filter effect combinations.

7.2.1 CPPN Filters

To most effectively generate a variety of filters relevant to users, it is possible to draw on existing image processing efforts to guide CIE. Within the field of Digital Image Processing, there are many techniques for filtering images [33]. One such technique, convolution filtering, is achieved by dragging a small n×n pixel window across an image and performing a weighted sum of pixels within the window to generate a new color value for the pixel in the center [33]. A small matrix of weights within the convolution kernel determines how to calculate the weighted sum that influences the final effect on the image, e.g. blur or image sharpening effects. Figure 7.1 depicts example convolution filter weights and the final images produced, highlighting the importance of the relative values within a convolution filter. Convolution kernels are capable of many different effects with a small adjustment to the matrix of weights, and thereby serve as a good starting place for creating a potential encoding within the #filters application.

Figure 7.1: **Convolution Filters.** The effects are shown of applying multiple 3×3 convolution filters to alter the original image (1). The convolution grid is dragged across every pixel of the original input (1) producing the final images (2), (3), and (4). Note that the 3×3 grid next to images (2), (3), and (4) indicates the values of the convolution matrix for the three filter variants. Importantly, varying the relative values within the convolution matrix can result in a variety of filters with different overall effects and magnitudes.

Given the variety of effects produced by simple modifications to the convolution matrix, one way to encode image filters is by selecting convolution matrix values via CPPN outputs. To understand how this approach works, figure 7.2 examines how the CPPN-based encoding constructs the con-

volution matrix for each pixel region. The key insight is that the 3×3 convolution filter can be represented by a CPPN queried at the nine corresponding pixel coordinates required to generate the full matrix of convolution values.

In effect, by inputing a region of pixels into the CPPN and encoding the convolution matrix as outputs of the CPPN, each convolution matrix applied to the original image is unique. As the filter is dragged across the original image, the CPPN can decide how to adjust the convolution matrix at each pixel region. Such an encoding potentially allows for selective effects, wherein the CPPN only applies an effect for a specific input region combination (e.g. blurring certain pixel regions).

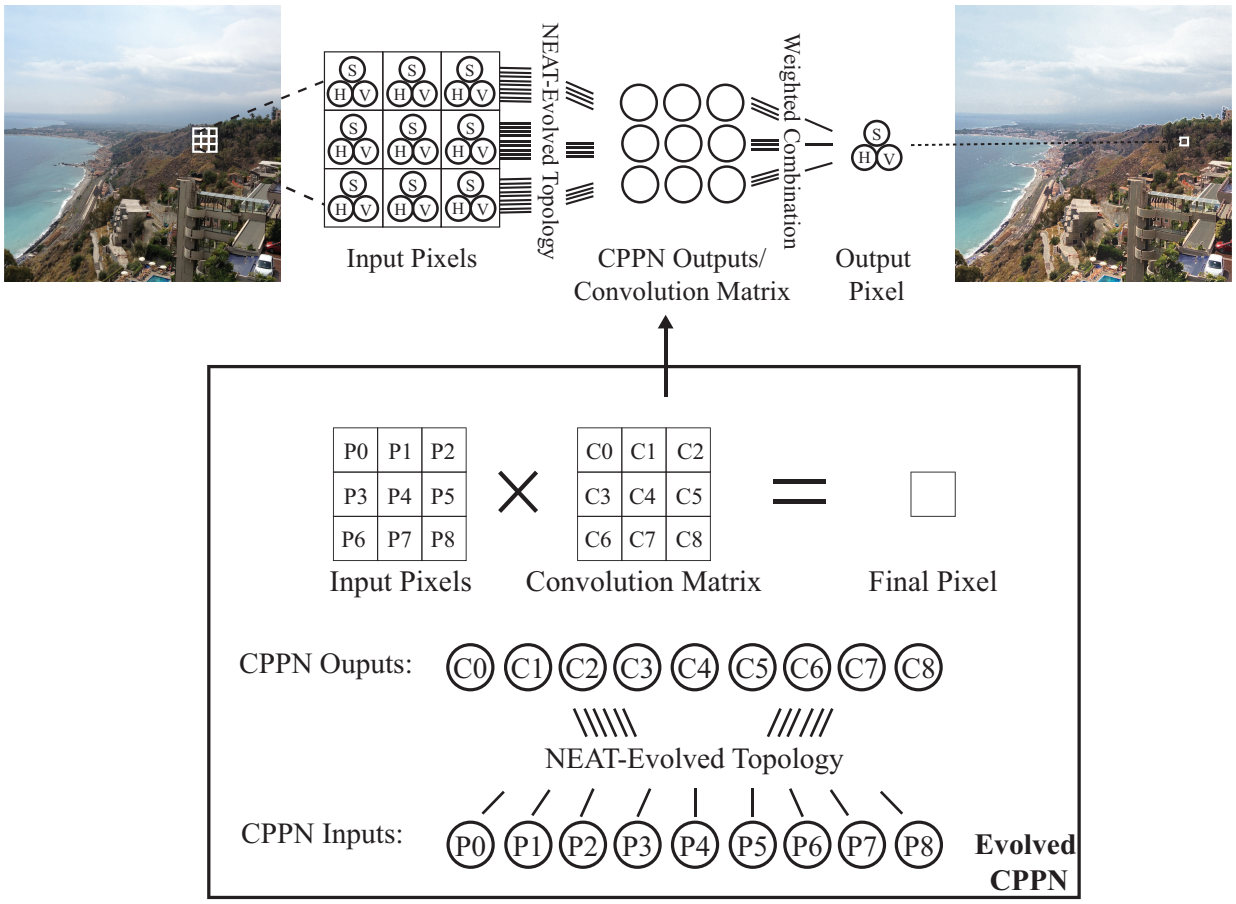


Figure 7.2: **CPPN Filters.**

Depicted above, an image filter is applied to the original image on the left to produce a filtered image on the right. Each filter evolved within #filters is encoded with a compositional pattern producing networks (CPPN; [82]). To filter an image, the CPPN is fed a 3×3 region of pixels, and outputs a corresponding convolution matrix for that region (for more details on CPPNs, see Section 2.4). The convolution matrix is then convolved with the original pixels to produce the color value for the center pixel. This grid-based filter is then dragged across the entire image on the left, region by region to produce the final image seen on the right.

7.2.2 Filter Combinations

Crucially, though CPPNs are capable of producing a wide variety of filters, users may wish to have more than one effect applied to different parts of a single image. That is, because there are multiple potential subjects within a given photograph, users may wish to apply different filters for distinctly different subjects, e.g. a dedicated filter to enhance the blue hues of the sky and another to subtly mute the skin tone of any person in the image. Similarly, users may desire fine-grained control over a whole set of filters applied in succession to produce the final filtered image within #filters, resulting in nearly limitless user customization for any given image.

Allowing an arbitrary number of CPPN-based image effects to be contained within a single evolutionary artifact presents a complicated encoding challenge. Fortunately, as described in chapter 5, the generic design of WIN already supports the storage and cataloging of artifacts containing an arbitrary-length array of inner encodings. In fact, for this reason, no additional changes to the WIN library are required to enable artifacts in #filters with multiple CPPN-based image effects.

Interestingly, once the #filters artifact is designed as a collection of filter effects instead of a single effect, including non-evolutionary filters within the application is a natural extension. One investigation into constructing common filter effects similar to those found within Instagram revealed the important and deceptively simple technique of overlaying a predefined background image on top of an existing image to produce basic vignette (darkening towards the edges of the image) or grainy texturing effects [10]. While not technically sophisticated, overlaying static images underpins a common set of desired effects within the Instagram community. As such, #filters permits the addition of image overlay effects, which blend user images with a predefined background image selected from a small set provided within #filters.

To avoid confusion, it is important to note that a single filter artifact in #filters may actually contain

an entire array of statically defined or CPPN-encoded filter effects. Consequently, the final product of #filters is the image produced by applying *all* filter effects defined by the artifact in succession to the original user-supplied image. Importantly, the ability to include complex artifacts in #filters without modification to the underlying WIN infrastructure is a non-trivial feature of the WIN library, reinforcing the value of the framework for future potential experimental domains not yet conceived.

Similarly to win-Picbreeder and win-IESoR, when a user in #filters decides the final filtered image matches his or her aesthetic preferences, the user can caption and publish the filtered image to the user's preferred social network while both the unfiltered image and the entire filter artifact containing possibly many filter effects are stored to the #filters database powered by WIN. Other users can then explore images published on #filters and see both the filtered and unfiltered versions to isolate the changes made by any given filter artifact. The following section analyzes how the image filter evolutionary search progresses and the way users discover new filters within #filters.

7.3 Search and Discovery on Mobile

To help address future EC and ALife research with WIN on mobile, there are several additions to WIN required to be more effective across a range of experimental domains beyond the image filtering domain of #filters. Specifically, the primary problem posed by mobile is the search and discovery of artifacts produced by evolution. That is, constrained by mobile, how do users effectively find artifacts within a domain aided by interactive evolution, and what common mechanic allows one user to discover artifacts that were produced by other users within the application?

At heart, both issues revolve around circumventing the impact of user fatigue in IEC [87]. First, the problem of evolutionary search with limited screen real estate critically impacts the effectiveness

of the mobile user, and without a new IEC mobile paradigm users may fatigue quickly. Second, user discovery of new artifacts and innovations within the mobile application is critical to encouraging and facilitating user branching, the primary functionality underlying CIE applications like Picbreeder, which can thereby exploit the efforts of all users despite individual fatigue.

To solve these fundamental problems at the heart of mobile, #filters employs *infinite feed IEC* and *hashtag discovery* to reorganize both search and discovery around common paradigms with which mobile users are already familiar.

7.3.1 Infinite Feed IEC

Within any IEC experiment, evolution can generate as many offspring as can be reasonably fit into the existing screen resolution. For instance, Picbreeder generates fifteen options for every user selection by displaying all offspring in a 5×3 grid of images. Practically, the same grid design would be difficult to replicate on a mobile device without severely restricting the size of each image. Instead, an alternate method is required to expose the user to an uncountably large set of possible offspring within a constrained screen size.

Fortunately, Internet users are already accustomed to a number of existing design paradigms for scrolling through a large collection of content. Particularly relevant to mobile applications is the concept of an *infinite feed*, which works as follows: First, the user is shown a small set of existing content, e.g. social media posts or news headlines. As the user scrolls down the list and approaches the bottom of the content displayed on screen, the application fetches more data in the background and appends the new content to the bottom of the list. Effectively, as the user consumes content, it is continually replenished and appended by the application giving the illusion of a screen that never ends, i.e. an infinite feed.

By harnessing the same design principles, users within #filters can similarly scroll through an endless set of image filters generated by interactive evolution. Figure 7.3 visually depicts this process of IEC within #filters. Note that #filters modifies the traditional infinite feed to instead focus on one user-selected image at a time, while providing an infinite horizontal feed of image filter previews created through evolution that the user can then investigate one by one. In practice, the key insight is that #filters exemplifies one potential solution to the problem of IEC for restricted screen sizes by employing well-known mobile design paradigms.

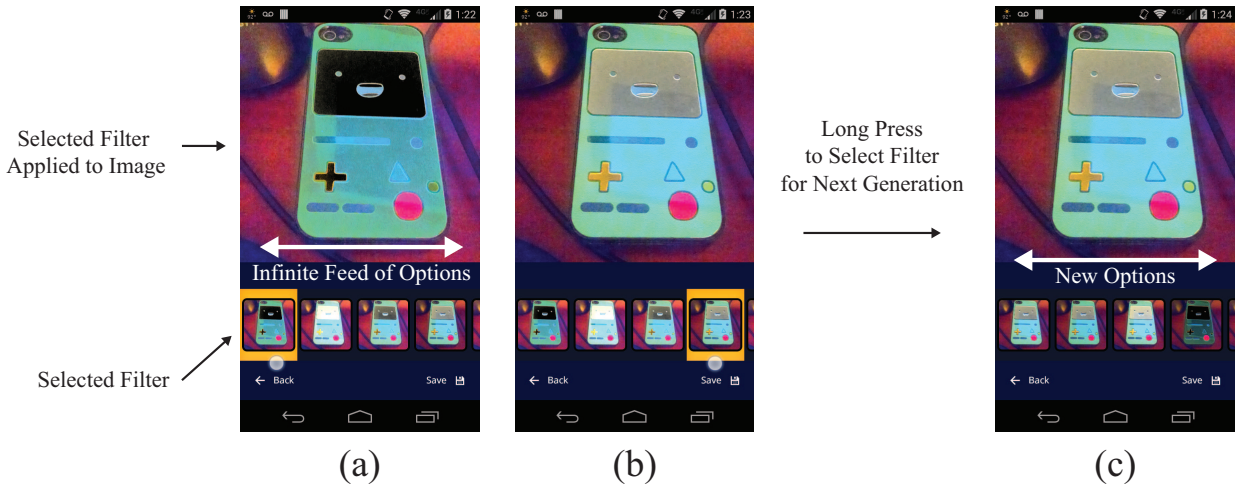


Figure 7.3: **IEC Design Pattern in #filters.** To handle interactive evolution on limited screen sizes, #filters appropriates an existing mobile design pattern *infinite feed* to display a limitless number of image filters to the user. In (a), the user selects a filter from the infinite feed to see the effect at a larger resolution displayed at the top. As a user scrolls through the options (b), he or she may select other filters to examine in more detail. Finally, to see children of the selected filter the user long presses on the chosen filter (c) before being shown a new stream of filters for browsing.

7.3.2 Hashtag Discovery

Within #filters, *hashtag discovery* enables users to search for image filters through all existing *hashtags*. Originally, hashtags were an informal way for early users of the social networking service Twitter to categorize messages (tweets) into specific topics and themes [68]. To create a hashtag, a user simply precedes the desired hashtag with a # symbol anywhere in the message. There are no predefined limitations or strict rules for hashtag placement, e.g. acceptable hashtags include #Science, #TwoWords, and #AnythingYouWant. Twitter would then aggregate hashtags across all tweets, allowing users to search for tweets directly by hashtag. Recently, the fundamental concept behind hashtags has spread more broadly across Internet culture as a way for users to categorize types of content other than tweets, e.g. pictures, and express individual feelings about a particular topic [68].

Importantly, despite being a recent social phenomena, hashtags represent a common organizing principle that orients data within a service relative to the individuals or groups that are most interested via a decentralized user mechanic. Therefore, supporting hashtags within WIN better enables individualization of evolutionary content by organizing generated filters around relevant user interests. Hashtagging is similar in practice to keyword tags employed by other services including Picbreeder [73]. However, unlike more traditional tagging, hashtags encourage content personalization organically and can often capture higher level concepts, e.g. sentiment, irony, or emotional expression. In particular, hashtags are meant to be naturally interspersed with descriptions or captions created by the user. For instance, an Instagram user may post an image of their current meal accompanied by a caption that reads "This is #delicious," where #delicious is a hashtag expressing the user's feelings about the image while not directly describing the content of the image.

Given the advantages of hashtagging for personalization, hashtag support is included in both #filters and the WIN framework, including support for arbitrary hashtags.

Together, infinite feed IEC and hashtag discovery play an important role in reinforcing the utility of user personalization with evolutionary content in familiar ways on mobile. Recall that while Picbreeder had a core collection of enthusiastic users, a majority of visitors were not dedicated to the evolved content [73]. Hashtag discovery aims to more directly align evolutionary content produced by WIN towards the interests of its users. As such, #filters focuses heavily on artifact browsing and discovery through hashtags, underscoring the derivation of the #filters hashtag name.

Building on these new mobile CIE paradigms, the next section investigates the results from the #filters experiment.

7.4 Results

It is important to note that the #filters application is now available for public download at `http://winark.org/filters`. In fact, #filters is the first CIE application available on an Android device, and represents a successful demonstration of the WIN framework's generic tracking and storage capabilities on mobile devices. Similar to applications previously developed for the WIN framework, win-Picbreeder and win-IESoR (described in chapter 5), the #filters application supports user-driven branching and publishing functionality, a key feature of the framework.

7.4.1 Many-to-One Image Effects

Note that unlike win-Picbreeder and win-IESoR, the evolutionary products of the image filtering domain can generate more than one final image. That is, #filters' users are directly interested in the single personalized and filtered images being produced by the application, while only indirectly concerned with the image filters that can be utilized to generate other filtered images. Interestingly, this generative aspect of image filter effects enables the construction of a entirely different set of

results, wherein a single image effect discovered in #filters can be applied to many different images without requiring further interactive evolution.

Figures 7.4 and 7.5 depict two unique image effects discovered by the #filters application being applied to the original image that the user chose to filter, as well as several other user-contributed images collected elsewhere within the experiment. Importantly, as described in the previous section, #filters encourages such serendipitous application of image effects through the mechanism of hashtag discovery.

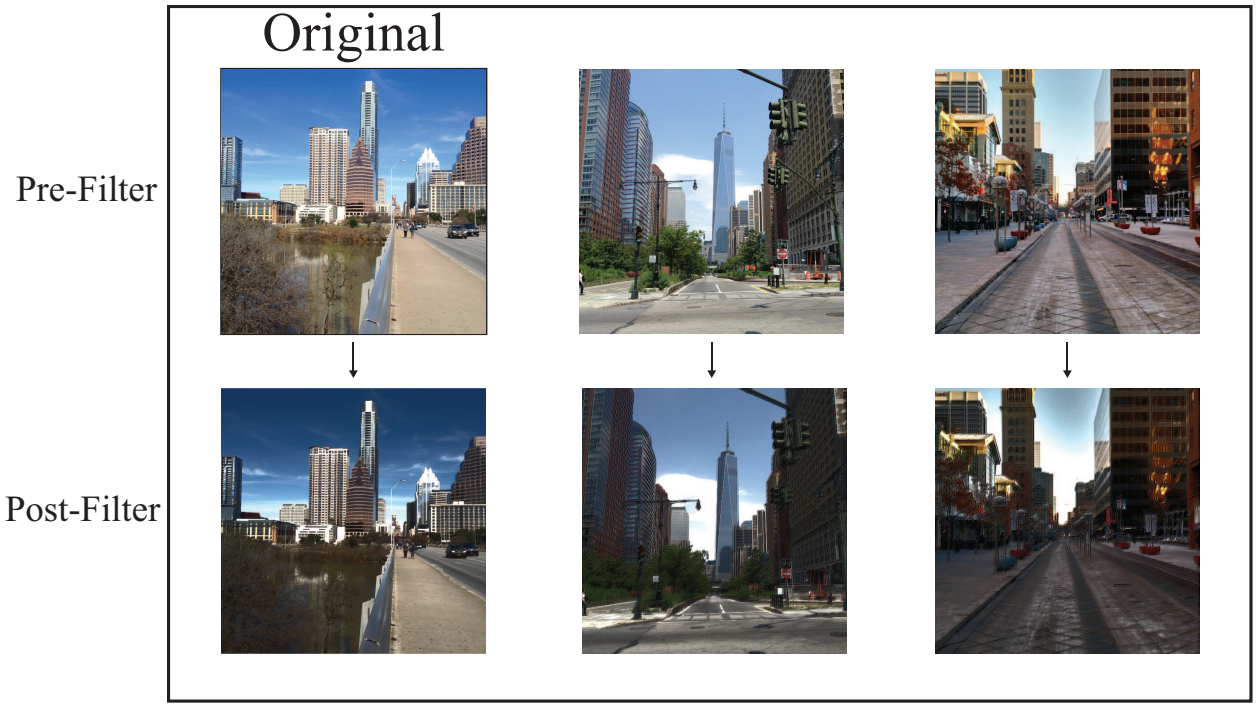


Figure 7.4: **Filter application to multiple images.** An example filter discovered by the #filters application is examined in closer detail. The image without any image processing, marked "Pre-filter", is displayed above, while the image after a filter is applied, denoted "Post-filter", is shown on the bottom. The original image employed during evolutionary search is marked, while the other images were collected elsewhere within the application. In the original image, the filter applies a darkening effect on the horizon, while highlighting the building outlines. When applied to other city skylines, the filter behaves similarly.

Figure 7.5: **Additional filter application to multiple images.** As in figure 7.4, an example filter found within the #filters application is applied to multiple images, with the pre-filter image above, and the post-filter image below. On the far left, the filter was originally applied to a picture of dogs in the snow at a park with the effect of highlighting the animals in the scene. Interestingly, when applied to other animal pictures, the filter also has a highlighting effect despite the difference in hues. On the far right, the fur coloring of a sleepy kitten is made prominent by the same filter.

7.4.2 Filter Phylogenies

Recall from chapter 5, the parent-child relationships being tracked within the WIN framework can be composed into an artificial phylogeny, a hierarchical display of all or part of the evolutionary

lineages found during search. Similar to the phylogenies constructed for win-Picbreeder and win-IESoR, figure 7.6 reveals a cross-section of artifacts generated by several users of #filters. Notably, the discovered image effects displayed in figure 7.6 are openly accessible for extension by users of the #filters application, a direct benefit of the WIN framework.

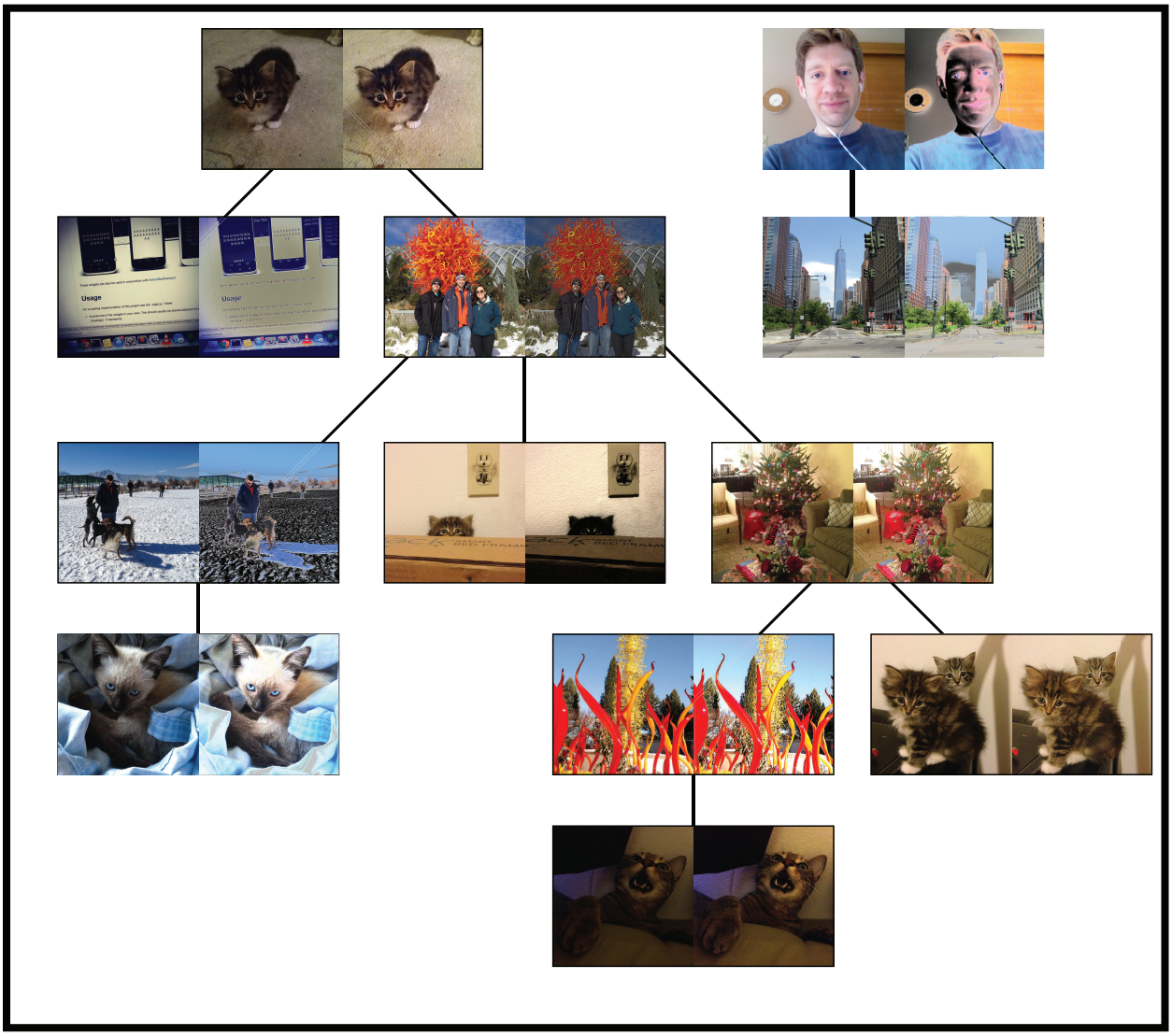


Figure 7.6: **WIN Phylogeny for #filters.** The tree of artifacts collected from the #filters Android application resulting from the efforts of several users. Each square represents a published image filter applied to a user-supplied photograph. The connection between squares represents the direct genetic relationship between the image filters, though there is no implied relationship among the images chosen by different users. This artificial phylogeny is a small cross-section of an ongoing experiment with users of the #filters application actively expanding the discovered set of image filters through the Android app available publicly at `http://winark.org/filters`.

7.5 Implications

The #filters app serves as a capstone application for the WIN framework demonstrating what one example of a fully realized WIN application looks like. Furthermore, #filters represents a significant scientific contribution. Not only is #filters the first collaborative interactive evolution experiment for any mobile platform, but it is also the first experiment to employ a CPPN-based encoding for image filtering.

However, the hope for #filters extends beyond a single evolutionary domain or a one-off Android application. Instead it aspires to illuminate a path for more researchers to follow suit and build entirely new CIE applications for Android. The immense power of collaborative interactive evolution to conceal the complexities of sophisticated domains from end users is largely untapped across all mobile operating systems. Where once there was a dearth of community infrastructure, #filters now provides a non-trivial example and blueprint for what CIE assisted by WIN can accomplish on Android.

In isolation, #filters is a scientific contribution that traditionally might have taken several researchers a year of effort to construct. With WIN, the same effort took half the time with only a single researcher. Excitingly, future researchers utilizing WIN for Android will not have to imagine new mobile design paradigms for IEC or extend WIN infrastructure to support user hashtags, instead only focusing on the underlying experimental domain and encoding development. In this way, #filters leads the way to a generation of new CIE mobile applications with more personalized evolutionary content.

CHAPTER 8: DISCUSSION AND FUTURE OPPORTUNITIES

While many tools exist to assist researchers in constructing or running experimental domains in Artificial Life (ALife) and Evolutionary Computation (EC), e.g. the Java ECJ suite [59] or the ALife Zoo simulation platform [39], WIN represents a unique effort to explicitly enable researchers to build or extend domains with collaborative branching and persistent storage of research artifacts and relationships. Such effort reflects a community-wide interest in supporting open science and public engagement through the Internet [88].

The next section discusses the importance of the WIN framework for augmenting the EC community's ability to contribute to citizen science. The following section then explores the significance of the WIN framework's support for collaborative interactive evolution (CIE) on mobile devices.

8.1 Tool for Citizen Science

Notably, previous CIE applications such as Picbreeder [73] and Endless Forms [11] established the potential for evolutionary computation to aid in serious citizen science. In fact, Picbreeder successfully solicited thousands of scientific contributions from hundreds of users [73], and contributed directly to the creation of new evolutionary search techniques, e.g. HyperNEAT [32] and Novelty Search [56].

Demonstrated through the construction of win-Picbreeder and win-IESoR in chapter 5, WIN greatly reduces the engineering effort required to add CIE functionality to existing evolutionary domains. Similarly, chapter 6 reveals the same framework can enable single researchers to quickly modify existing CIE applications and then launch variant CIE services, and the previous chapter detailed the first CIE application on a mobile device. Most important, all of these efforts have been made

publicly available through open-source code and documentation so that future researchers can build upon previously completed work.

In effect, the WIN library has consistently shown promise for improving the efficiency of a single researcher in collaborative evolution. Originally, the Picbreeder service took six people a full year to construct [73]. In contrast, win-Picbreeder, a clone of the Picbreeder service built on WIN, took one researcher several months, while three researchers not familiar with the program managed to build several variations in under four hours. In fact, unlike the original Picbreeder, the source code for win-Picbreeder is fully open-source meaning any software developer can create and launch a replica of the win-Picbreeder service on a local computer within ten minutes [85].

The WIN platform thereby offers a substantial reduction in time required to open a domain to online access and contribution, a crucial stepping stone towards proliferating citizen science projects powered by evolution. That is, the intersection of collaborative evolution and citizen science is a rich opportunity to draw public attention to the work being done in evolutionary computation as well as extracting meaningful scientific contributions from crowd-sourcing infrastructure. Enabled by the WIN framework, the fields of evolutionary computation and artificial life are uniquely positioned to proliferate experiments that can aggregate the efforts of an entire community, with the potential to change the way in which research is conducted.

8.2 CIE on Mobile Devices

Highlighting the potential created by WIN for new scientific directions, the #filters Android application reviewed in chapter 7 is the first CIE application constructed for any mobile device. Interestingly, recent data suggests that consumers in the United States access the Internet more often from their smartphones than their desktop PCs, hinting at the increasing importance of mobile

devices as a computational platform [67]. Such user statistics convey the potential value of WIN in positioning the field of evolutionary computation to reap the rewards of the global reach of mobile devices to further our scientific objectives.

Furthermore, #filters is open-source software and a blueprint to future researchers for proliferating such CIE domains across the Android ecosystem. Practically, software design contributions like the *infinite feed IEC* approach described in Section 7.3.1 assist researchers by providing a fully-constructed program infrastructure for enabling IEC on the limited resources of a mobile device. That is, #filters is a first step in personalized evolutionary content for mobile users, and, most importantly, it lays a strong foundation for an entirely new class of experiments for an underutilized research platform.

It is important to note that such concerns about the influence and availability of evolutionary experiments on mobile devices is more than an engineering challenge for the field. It is becoming increasingly clear from both industry analysis [34, 49] and widespread consumer behavior [18, 29] that mobile devices will continue to play an integral part of everyday life from leisure activities to workplace productivity. To have a limited presence on such an important platform is a significant risk to the longterm relevance of the fields of ALife and EC. Moreover, the hallmark feature of the most popular mobile applications with billions of active users, e.g. Facebook, Instagram, and Snapchat, is the direct availability of deeply personal content [18]. Yet personalized content is a well-documented and defining benefit of IEC and has been demonstrated across a variety of complex and often subjective domains [19, 45, 48, 75, 86, 87]. Directly exploiting this open opportunity for the field, #filters and the WIN framework represent a significant first step in creating a prominent presence on an important computational platform, while providing the inspiration and concrete research tools for future research in the area.

CHAPTER 9: CONCLUSION

This dissertation introduced Worldwide Infrastructure for Neuroevolution (WIN), a framework that enables researchers to easily build persistent collaborative evolutionary experiments accessible online and open for contribution from multiple users. Validated through five major contributions, the **hypothesis** of this work is that the WIN framework is an effective platform-building tool for assisting researchers in creating large-scale collaborative interactive evolution (CIE) experiments across a wide range of evolutionary domains.

The first contribution, the MaestroGenesis software I co-created with Amy K. Hoover, demonstrated an IEC experiment in the domain of music composition that forms the foundation upon which the larger WIN framwork was constructed. The second, a new virtual creature domain named Indirectly Encoded SodaRace (IESoR), established a visually compelling and objective research domain for artificial life research. Both the new IESoR domain and previously constructed picture evolution domain powering Picbreeder [73] were integrated into the third contribution, a prototype WIN framework. Further supporting the main hypothesis, this prototype framework validated the ability for the WIN library to replicate the functionality of existing CIE applications like Picbreeder and augment more objective-driven evolutionary domains like IESoR with collaborative features. The fourth contribution, a developer survey conducted with WIN, bolstered assertions that WIN could enable researchers to quickly construct and launch new variations to existing CIE experiments. The final contribution is a fully functional mobile application *#filters* built for the Android platform, revealing the generic WIN framework's support for multiple computational platforms from the browser to mobile devices.

At the heart of the work presented in this dissertation is an open question of how the fields of ALife and EC can most effectively take advantage of the boundless opportunities unlocked by the

widespread adoption of both the Internet and mobile computing [34, 49, 88]. Thus the promise of this work is to deliver a meaningful reduction in the resource and time investment required for a single researcher to extend almost any evolutionary domain to reach hundreds or thousands of users.

It is worthwhile to consider that the fields of ALife and EC may be at a delicate inflection point, where the right combination of shared community resources and software infrastructure like WIN could propel both fields to the forefront of the crowd-sourcing and open science movements.

LIST OF REFERENCES

[1] Auerbach, J., and Bongard, J. 2010. Evolving CPPNs to grow three dimensional structures. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2010)*. New York, NY: ACM Press.

[2] Auerbach, J. E., and Bongard, J. C. 2012. On the relationship between environmental and morphological complexity in evolved robots. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2012)*, 521–528. New York, NY: ACM Press.

[3] Bahceci, E., and Miikkulainen, R. 2008. Transfer of evolved pattern-based heuristics in games. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games (CIG-2008)*. Piscataway, NJ: IEEE Press.

[4] Bongard, J. C., and Hornby, G. S. 2013. Combining fitness-based search and user modeling in evolutionary robotics. In *Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference*, GECCO '13, 159–166. New York, NY, USA: ACM.

[5] Bongard, J. C., and Paul, C. 2000. Investigating morphological symmetry and locomotive efficiency using virtual embodied evolution. In *Proceedings of the Sixth International Conference on Simulation of Adaptive Behavior*, 420–429. MIT Press.

[6] Bongard, J. C. 2013. Ludobots website. The Ludobots software is publicly available at http://www.uvm.edu/ ludobots/.

[7] Buk, Z.; Koutník, J.; and Snorek, M. 2009. NEAT in HyperNEAT substituted with genetic programming. In *Inernational Conference on Adaptive and Natural Computing Algorithms (ICANNGA-2009)*, 243–252. Berlin: Springer.

[8] Channon, A. 2001a. *Evolutionary Emergence: The Struggle for Existence in Artificial Biota*. Ph.D. Dissertation, University of Southampton.

[9] Channon, A. 2001b. Passing the alife test: Activity statistics classify evolution in geb as unbounded. In *Proceedings of the European Conference on Artificial Life(ECAL-2001)*. Springer.

[10] Chen, C. 2014. How does instagram develop their filters? `http://www.quora.com/How-does-Instagram-develop-their-filters`.

[11] Clune, J., and Lipson, H. 2011. Evolving three-dimensional objects with a generative encoding inspired by developmental biology. In *Proceedings of the European Conference on Artificial Life (ECAL-2011)*, 141–148.

[12] Clune, J.; Beckmann, B.; Pennock, R.; and Ofria, C. 2009. HybrID: A hybridization of indirect and direct encodings for evolutionary computation. In *Proceedings of the European Conference on Artificial Life (ECAL-2009),*.

[13] Clune, J.; Beckmann, B.; McKinley, P.; and Ofria, C. 2010. Investigating whether HyperNEAT produces modular neural networks. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2010)*. New York, NY: ACM Press.

[14] Clune, J.; Stanley, K. O.; Pennock, R. T.; and Ofria, C. 2011. On the performance of indirect encoding across the continuum of regularity. *IEEE Transactions on Evolutionary Computation*.

[15] Clune, J.; Ofria, C.; and Pennock, R. 2008. How a generative encoding fares as problem-regularity decreases. In *Proceedings of the 10th International Conference on Parallel Problem Solving From Nature (PPSN 2008)*, 258–367. Berlin: Springer.

[16] Clune, J.; Pennock, R. T.; and Ofria, C. 2009. The sensitivity of HyperNEAT to different geometric representations of a problem. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2009)*. New York, NY, USA: ACM Press.

[17] Colton, S., and Torres, P. 2009. Evolving approximate image filters. In Giacobini, M.; Brabazon, A.; Cagnoni, S.; Di Caro, G.; Ekárt, A.; Esparcia-Alcázar, A.; Farooq, M.; Fink, A.; and Machado, P., eds., *Applications of Evolutionary Computing*, volume 5484 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. 467–477.

[18] comScore/the Kelsey group. 2015. comscore reports december 2014 u.s. smartphone subscriber market share. Press Release. `http://www.comscore.com/Insights/Market-Rankings/comScore-Reports-December-2014-US-Smartphone-Subscriber-Market-Share`.

[19] Cook, M.; Colton, S.; Raad, A.; and Gow, J. 2013. Mechanic miner: Reflection-driven game mechanic discovery and level design. In Esparcia-Alcázar, A., ed., *Applications of Evolutionary Computation*, volume 7835 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. 284–293.

[20] Cooper, S.; Khatib, F.; Treuille, A.; Barbero, J.; Lee, J.; Beenen, M.; Leaver-Fay, A.; Baker, D.; Popovic, Z.; and players, F. 2010. Predicting protein structures with a multiplayer online game. *Nature* 466(7307):756–760.

[21] Crockford, D. 2006. RFC 4627 - The application/json Media Type for JavaScript Object Notation (JSON). Technical report.

[22] Crowston, K., and Prestopnik, N. R. 2013. Motivation and data quality in a citizen science game: A design science evaluation. In *Proceedings of the 2013 46th Hawaii International Conference on System Sciences*, HICSS '13, 450–459. Washington, DC, USA: IEEE Computer Society.

[23] Dahl, R. L. 2009. Node.js software package. The Node.js software package is publicly available at http://nodejs.org/.

[24] Dawkins, R. 1986. *The Blind Watchmaker.* Essex, U.K.: Longman.

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

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

[27] Drchal, J.; Kapra, O.; Koutnik, J.; and Snorek, M. 2009. Combining multiple inputs in HyperNEAT mobile agent controller. In *19th International Conference on Artificial Neural Networks (ICANN 2009)*, 775–783. Berlin: Springer.

[28] Drchal, J.; Koutník, J.; and Snorek, M. 2009. HyperNEAT controlled robots learn to drive on roads in simulated environment. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC-2009)*. Piscataway, NJ, USA: IEEE Press.

[29] Frier, S. 2014. Instagram user count at 300 million passes twitter. `http://www.bloomberg.com/news/articles/2014-12-10/instagram-says-itll-delete-spam-accounts-forever`.

[30] Gabriel, I., and Viorel, N. 2012. *awutm*, volume 50. chapter Neuro-evolution in Zero-Sum Perfect Information Games on the Android OS, 27. 2.

[31] Galiegue, F., and Zyp, K. 2013. Json schema: core definitions and terminology draft-zyp-json-schema-04. Working Draft.

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

[33] Gonzalez, R. C., and Woods, R. E. 2006. *Digital Image Processing (3rd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc.

[34] Google. 2013. Our mobile planet: United states of america. understanding the mobile consumer. `http://services.google.com/fh/files/misc/omp-2013-us-en.pdf`.

[35] Green, D. 2007. Vlab website. The VLAB website is is publicly available at http://vlab.infotech.monash.edu.au/.

[36] Haasdijk, E.; Rusu, A. A.; and Eiben, A. 2010. HyperNEAT for locomotion control in modular robots. In *Proceedings of the 9th International Conference on Evolvable Systems (ICES 2010),*.

[37] Harding, S. 2008. Evolution of image filters on graphics processor units using cartesian genetic programming. In *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*, 1921–1928.

[38] Hein, D.; Hild, M.; and Berger, R. 2007. Evolution of biped walking using neural oscillators and physical simulation. In *RoboCup 2007: Proceedings of the International Symposium*, LNAI. Springer.

[39] Hickinbotham, S.; Weeks, M.; and Austin, J. 2013. The alife zoo: cross-browser, platform-agnostic hosting of artificial life simulations. In *Proceedings of the European Conference on Artificial Life (ECAL-2013)*.

[40] Honey, M. A., and Hilton, M., eds. 2010. *Learning science through computer games and simulations*. National Academies Press.

[41] Hoover, A. K., and Stanley, K. O. 2009a. Exploiting functional relationships in musical composition. *Connection Science Special Issue on Music, Brain, & Cognition* 21(2):227–251.

[42] Hoover, A. K., and Stanley, K. O. 2009b. Exploiting functional relationships in musical composition. *Connection Science Special Issue on Music, Brain, and Cognition* 21(2 and 3):227–251.

[43] Hoover, A. K.; Szerlip, P. A.; ; and Stanley, K. O. 2011. Generating musical accompaniment through functional scaffolding. In *Proceedings of the 8th Sound and Music Computing Conference (SMC-2011)*.

[44] Hoover, A. K.; Szerlip, P. A.; and Stanley, K. O. 2011. Interactively evolving harmonies through functional scaffolding. In *GECCO '11: Proceedings of the 13th annual conference on Genetic and evolutionary computation*, 387–394. Dublin, Ireland: ACM.

[45] Hoover, A. K.; Szerlip, P. A.; and Stanley, K. O. 2012. Generating a complete multipart musical composition from a single monophonic melody with functional scaffolding. In Maher, M. L.; Hammond, K.; Pease, A.; Perez, R. P. Y.; Ventura, D.; and Wiggins, G., eds., *Proceedings of the 3rd International Conference on Computational Creativity (ICCC-2012)*.

[46] Hoover, A. K.; Szerlip, P. A.; and Stanley, K. O. 2013. Implications from music generation for music appreciation. In *Proceedings of the 4th International Conference on Computational Creativity (ICCC-2013)*.

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

[48] Hui, O. J.; Teo, J.; and On, C. K. 2011. Interactive evolutionary programming for mobile games rules generation. In *Sustainable Utilization and Development in Engineering and Technology (STUDENT), 2011 IEEE Conference on*, 95–100.

[49] IDC. 2014. Smartphone os market share, q2 2014. `http://www.idc.com/prodserv/smartphone-os-market-share.jsp`.

[50] Jaskowski, W.; Krawiec, K.; and Wieloch, B. 2008. Neurohunter - an entry for the balanced diet contest. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2008) Contest Program")*. New York, NY: ACM Press.

[51] Joachimczak, M., and Wrobel, B. 2012. Open ended evolution of 3D multicellular development controlled by gene regulatory networks. In *Proceedings of the Thirteenth International Conference on Artificial Life (ALIFE XIII)*, 67–74. Cambridge, MA: MIT Press.

[52] Kawrykow, A.; Roumanis, G.; Kam, A.; Kwak, D.; Leung, C.; Wu, C.; Zarour, E.; Sarmenta, L.; Blanchette, M.; Waldispühl, J.; and players, P. 2012. Phylo: A citizen science approach for improving multiple sequence alignment. *PLoS ONE* 7(3):e31362.

[53] Koza, J. R. 1998. *Genetic Programming II: Automatic Discovery of Reusable Programs*. Cambridge, MA: MIT Press.

[54] Krcah, P. 2007. Evolving virtual creatures revisited. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2007)*. New York, NY, USA: ACM Press.

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

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

[57] Lehman, J., and Stanley, K. O. 2011b. 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*, 211–218. Dublin, Ireland: ACM.

[58] Lintott, C. J.; Schawinski, K.; Slosar, A.; Land, K.; Bamford, S.; Thomas, D.; Raddick, M. J.; Nichol, R. C.; Szalay, A.; Andreescu, D.; Murray, P.; and Vandenberg, J. 2008. Galaxy zoo: morphologies derived from visual inspection of galaxies from the sloan digital sky survey. *Monthly Notices of the Royal Astronomical Society* 389(3):1179–1189.

[59] Luke, S. 2010. *The ECJ Owner's Manual – A User Manual for the ECJ Evolutionary Computation Library*, zeroth edition, online version 0.2 edition.

[60] Maley, C. C. 1999. Four steps toward open-ended evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-1999)*, 1336–1343.

[61] Matsui, T.; Arakawa, K.; and Nomoto, K. 2006. A nonlinear filter system for beautifying face images with enhancement using interactive evolutionary computing. In *Intelligent Signal Processing and Communications, 2006. ISPACS '06. International Symposium on*, 534–537.

[62] McOwan, P., and Burton, E. 2000. Sodarace website. `http://sodarace.net`.

[63] McOwan, P. W., and Burton, E. J. 2005. Sodarace: Adventures in artificial life. In *Artificial Life Models in Software*. Springer. 97–111.

[64] Morse, G.; Risi, S.; Snyder, C. R.; and Stanley, K. O. 2013. Single-unit pattern generators for quadruped locomotion. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2013)*. New York, NY, USA: ACM.

[65] Mouret, J.-B., and Doncieux, S. 2010. SFERESv2: Evolvin' in the multi-core world. In *Proc. of Congress on Evolutionary Computation (CEC)*, 4079–4086.

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

[67] Nielson. 2014. The digital consumer. `http://www.nielsen.com/content/dam/corporate/us/en/reports-downloads/2014%20Reports/the-digital-consumer-report-feb-2014.pdf`.

[68] Pang, C. 2013. What are hashtags and how to use them on social media. `http://www.wix.com/blog/2013/10/what-are-hashtags-and-how-to-use-them-on-social-media/`.

[69] Piskur, J.; Greve, P.; Togelius, J.; and Risi, S. 2015. Braincrafter: An investigation into human-based neural network engineering. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC-2015)*. Piscataway, NJ, USA: IEEE Press.

[70] Plugge, E.; Hawkins, T.; and Membrey, P. 2010. *The Definitive Guide to MongoDB: The NoSQL Database for Cloud and Desktop Computing*. Berkely, CA, USA: Apress, 1st edition.

[71] Reil, T., and Husbands, P. 2002. Evolution of central pattern generators for bipedal walking in a real-time physics environment. *IEEE Transactions on Evolutionary Computation* 6(2):159–168.

[72] Risi, S.; Lehman, J.; D'Ambrosio, D. B.; Hall, R.; and Stanley, K. O. 2012. Combining search-based procedural content generation and social gaming in the petalz video game. In *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE 2012)*. Menlo Park, CA: AAAI Press.

[73] Secretan, J.; Beato, N.; D.Ambrosio, D. B.; Rodriguez, A.; Campbell, A.; Folsom-Kovarik, J. T.; and Stanley, K. O. 2011. Picbreeder: A case study in collaborative evolutionary exploration of design space. *Evolutionary Computation* 19(3):345–371.

[74] Silvertown, J. 2009. A new dawn for citizen science. *Trends in Ecology and Evolution* 24(9):467–471.

[75] Sims, K. 1991. Artificial evolution for computer graphics. *Proceedings of the ACM Special Interest Group on Graphics and Interactive Techniques* 319–328.

[76] Sims, K. 1994. Evolving 3D morphology and behavior by competition. Cambridge, MA: MIT Press. 28–39.

[77] Soros, L. B., and Stanley, K. O. 2014. Identifying necessary conditions for open-ended evolution through the artificial life world of chromaria. In *Proceedings of the 14th International Conference on the Synthesis and Simulation of Living Systems (ALife XIV),*.

[78] Standish, R. 2003. Open-ended artificial evolution. *International Journal of Computational Intelligence and Applications* 3(167).

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

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

[81] Stanley, K. O. 2003. *Efficient Evolution of Neural Networks Through Complexification*. Ph.D. Dissertation.

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

[83] Szerlip, P., and Stanley, K. O. 2013. Indirectly encoded sodarace for artificial life. In *Proceedings of the European Conference on Artificial Life (ECAL-2013)*.

[84] Szerlip, P., and Stanley, K. O. 2014. Steps toward a modular library for turning any evolutionary domain into an online interactive platform. In *Proceedings of the Fourteenth International Conference on Artificial Life (ALIFE XIV)*. Cambridge, MA: MIT Press.

[85] Szerlip, P. 2014. 10 minute win-picbreeder installation tutorial. `http://cs.ucf.edu/~pszerlip/win/winpbvideo.webm`.

[86] Szumlanski, S. R.; Wu, A. S.; and Hughes, C. E. 2005. Collaborative interactive evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) Poster Session*. New York, NY: ACM Press.

[87] Takagi, H. 2001. Interactive evolutionary computation: fusion of the capabilities of ec optimization and human evaluation. *Proceedings of the IEEE* 89(9):1275–1296.

[88] Taylor, T. 2014. Artificial life and the web: Webal comes of age. *CoRR* abs/1407.5719.

[89] Verbancsics, P., and Stanley, K. O. 2010. Evolving static representations for task transfer. *Journal of Machine Learning Research (JMLR)* 11:1737–1769.

[90] Verbancsics, P., and Stanley, K. O. 2011. Constraining connectivity to encourage modularity in HyperNEAT. In *GECCO '11: Proceedings of the 13th annual conference on Genetic and evolutionary computation*, 1483–1490. Dublin, Ireland: ACM.

[91] Woolley, B. G., and Stanley, K. O. 2011. On the deleterious effects of a priori objectives on evolution and representation. In *GECCO '11: Proceedings of the 13th annual conference on Genetic and evolutionary computation*, 957–964. Dublin, Ireland: ACM.

[92] Woolley, B. G., and Stanley, K. O. 2014. Novel human-computer collaboration: Combining novelty search with interactive evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2014)*. New York, NY, USA: ACM. To appear.

[93] Yaeger, L. 1994. Computational genetics, physiology, metabolism, neural systems, learning, vision and behavior or polyworld: Life in a new context. In Langton, C. G., ed., *Artificial Life III, Proceedings Volume XVII*, 263–298. Addison-Wesley.

[94] Yosinski, J.; Clune, J.; Hidalgo, D.; Nguyen, S.; Zagal, J. C.; and Lipson, H. 2011. Evolving robot gaits in hardware: the hyperneat generative encoding vs. parameter optimization. In *Proceedings of the 20th European Conference on Artificial Life*.