DOI: http://dx.doi.org/10.7551/978-0-262-32621-6-ch147

Steps Toward a Modular Library for Turning Any Evolutionary Domain into an Online Interactive Platform

Paul Szerlip and Kenneth O. Stanley

Dept. of EECS (Computer Science Division), University of Central Florida, Orlando, FL 32816
pszerlip@eecs.ucf.edu, kstanley@eecs.ucf.edu

Abstract

Natural evolution inspires the fields of evolutionary computation (EC) and artificial life (ALife). A prominent feature of natural evolution is that it effectively never ends. However, most EC and ALife experiments are only run for several days or weeks at a time. Once an experiment concludes, reproducing, observing, or extending the results often requires considerable effort. In contrast, some Collaborative Interactive Evolution (CIE) systems, e.g. Picbreeder, were designed to preserve results as potential stepping stones to build upon later while taking advantage of human insight to solve challenging problems. Traditionally, building long-running and open experiments similar to Picbreeder presents a complex and time-consuming software challenge. To reduce this challenge and thereby remove the barrier to situating almost any experiment within an interactive online framework, this paper presents the initial prototype for Worldwide Infrastructure for Neuroevolution (WIN). Built in the model of Picbreeder, WIN is a modular library for significantly reducing the complexity of creating fully persistent, online, and interactive evolutionary platforms for any new or existing domain. WIN Online, the public interface for WIN, provides an online collection of domains built with WIN that lets novice and expert users browse and meaningfully contribute to ongoing experiments. Two example experiments in this paper demonstrate WIN's potential to quickly bootstrap any evolutionary domain with online and interactive capabilities.

Introduction

The fields of evolutionary computation (EC) and artificial life (ALife) are inspired by the products of natural evolution. As researchers in these areas, we hope to reproduce or one day exceed natural evolution's most inspiring qualities, for example through *open-ended evolution* (Channon, 2001; Maley, 1999) or evolving a diverse collection of virtual morphologies (Auerbach and Bongard, 2012; Hornby and Pollack, 2002; Lehman and Stanley, 2011; Sims, 1994; Szerlip and Stanley, 2013). However, almost all EC and ALife experiments fall short on one very salient feature of natural evolution: that it effectively never ends. Natural evolution perpetually branches from previous discoveries in a complex and never-ending process.

In stark contrast, presently, a research group that evolves e.g. a controller for a biped walking robot (Hein et al., 2007; Reil and Husbands, 2002) typically will report the result, publish a version of the code, retire the experiment, and move on to new domains. Although the experimental results are indeed published, the ability to reproduce, observe, or extend those results generally requires considerable effort outside of the originating group. Yet in natural evolution the stepping stones that lead to the greatest discoveries almost never foreshadow the circuitous and dramatic discoveries that follow them. For instance, who could predict that a flatworm would one day evolve to become a human being (Raff, 1996)? Every experimental artifact left behind is a potential stepping stone lost forever.

Interestingly, there are unique experimental systems that attempt to collect such potential stepping stones. Collaborate Interactive Evolution (CIE) systems are designed to allow contributions from multiple users over the course of the experiment (Szumlanski et al., 2005). One such CIE system, Picbreeder, a genetic art program for breeding pictures, was explicitly designed to allow users to branch collaboratively from previously discovered results in the space of pictures (Secretan et al., 2011). That is, every picture evolved in Picbreeder is either a descendant of another picture inside the system, or started from a simple random starting point. Additionally, the system is still active after seven years, and every picture discovered is available online to continue extending at `http://picbreeder.org/`.

In this way, while Picbreeder can only evolve pictures, it provides precedent for evolutionary systems capable not only of preserving stepping stones accessible to the community, but also taking advantage of of human insight to search the space of all artifacts for the most meaningful. Yet the benefit of human intuition does not only apply to search spaces with hard to define or subjective optimization metrics. In fact, recent experiments suggest that humans are capable of interleaving their own intuition with automated algorithms, yielding better results than the automated algorithms can alone (Bongard and Hornby, 2013; Woolley and Stanley, 2014). In an ideal world, any experiment would have the ability to leverage human intelligence to aid the search while also allowing discovered results to remain accessible to the community in perpetuity for future extension by humans, or by some clever algorithm.

Unfortunately, building experiments capable of interaction across the Internet is complex and time consuming. In fact, Picbreeder took over a year to build the online infrastructure (Secretan et al., 2011). Though web technologies have progressed in the years since Picbreeder was built, it would still take considerable effort to build such an infrastructure around any arbitrary experiment.

To begin to address this gap, this paper presents the initial prototype for Worldwide Infrastructure for Neuroevolution (WIN), a library that significantly reduces the complexity of creating fully persistent, online, and interactive (or automated) evolutionary platforms around any domain. The main outcome is to demonstrate that WIN can quickly ramp up CIE domains as well as effectively extend experimental domains that were not originally designed to be persistent or part of a CIE application. Yet the ambitions of WIN go beyond these example domains. At present, most researchers operate within isolated groups. The long term aim of WIN is to make it trivial to connect any individual or lab platform to the world, providing both a stream of online users, and archives of data and discoveries for later continuation. In short, the goal is to be able to continue where someone, or perhaps some algorithm, left off.

Background

In the fields of ALife and EC, many tools exist to assist researchers in constructing new experimental domains. Software packages like the Java-based ECJ suite (Luke, 2010) 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). Other efforts in the community aim to enhance a communal pool of resources for collecting and sharing results. For example, the ALife Zoo aims to take advantage of cloud computing to host a shared platform for running ALife simulations (Hickinbotham et al., 2013). The Virtual Complexity Lab (VLab) is an online resource that aggregates a variety of ALife simulations to stimulate interest in the field (Green, 2007).

Together, these platforms represent a significant contribution to the community, helping to proliferate new domains and 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.

The idea that casual human users can aid serious scientific endeavors has gained credibility in recent years with a number of online citizen science (Crowston and Prestopnik, 2013; Newman et al., 2012) 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. Within evolutionary robotics, Bongard (2013) created a simulator that sets precedent for harnessing amateur users for crowd-sourcing intricate robot controllers. Yet the benefits of crowd-sourcing can apply to more than any one domain.

There is thus a need for software that goes beyond organizing domains, simulations, or resources in the community. Such software could reconfigure scientific experiments to explicitly accumulate the results of the past in a *publicly* accessible format where they can be reproduced, observed, and extended with minimal effort.

WIN Architecture

To fully address the needs of the community, WIN serves as an active collaborative tool that not only stores past results, but presents them immediately for further exploration and extension. 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 (Dahl, 2009), 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 by the Picbreeder web service.

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.

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 is available in 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. 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.

Saving Data in WIN

However, simply making WIN modular and event-driven is not a panacea for integrating WIN with any domain. Each evolutionary experiment can have distinct genetic encodings with custom algorithms to mutate and cross over genotypes, or special methods for exploring the search space.

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 (Crockford, 2006), as well as the JSON Schema specification for data validation (Galiegue and Zyp, 2013). 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 (Plugge et al., 2010).

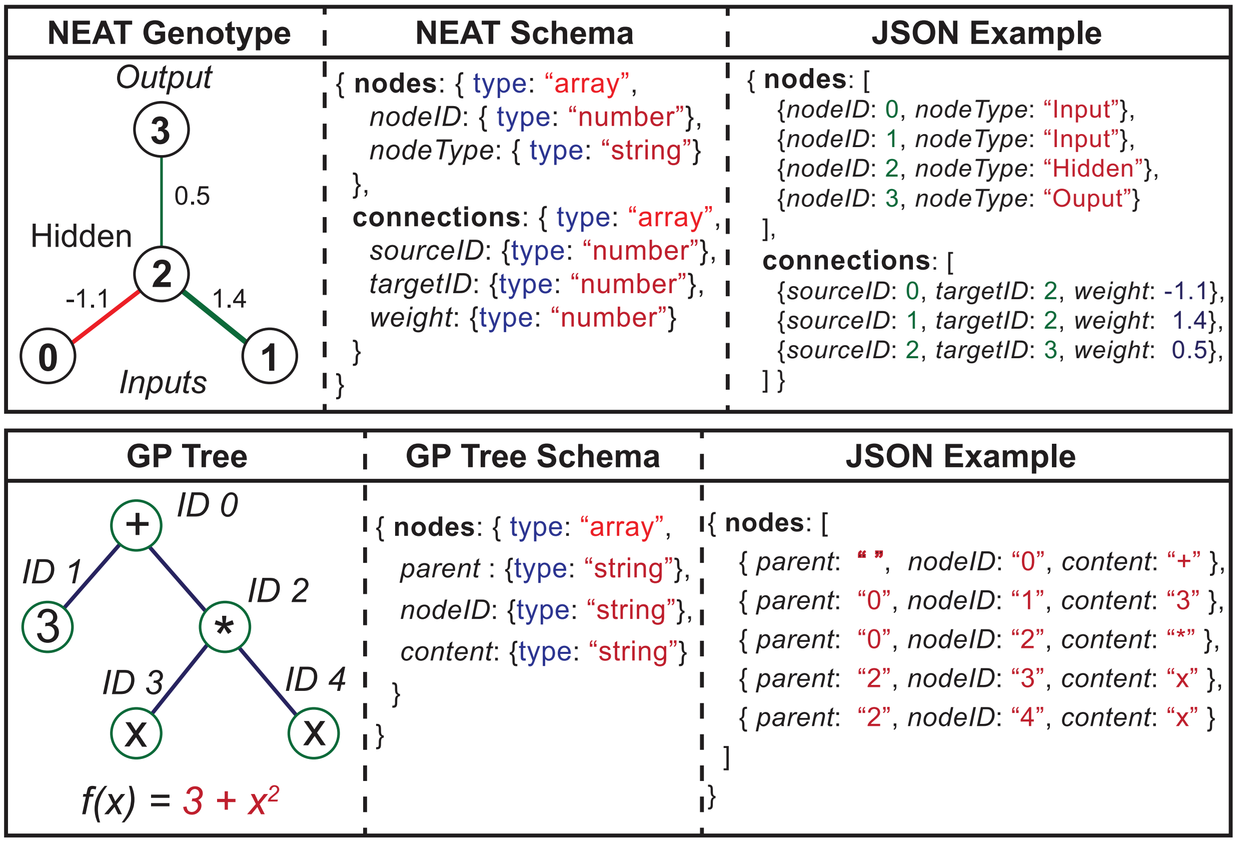
Note that because WIN utilizes Node.js and JavaScript for its underlying functionality, it is more convenient to select JSON for data formatting over other storage types, e.g. XML. Regardless of the specific storage format, the key required property is the ability to represent a wide variety of data 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 1 shows how different genetic encodings can be represented as simple JSON schema, all of which can be saved by WIN. 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.

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 (2011) 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



```
{ nodes: { type: "array",
    nodeID: { type: "number"},
    nodeType: { type: "string"}
  },
  connections: { type: "array",
    sourceID: {type: "number"},
    targetID: {type: "number"},
    weight: {type: "number"}
  }
}
```

```
{ nodes: [
    {nodeID: 0, nodeType: "Input"},
    {nodeID: 1, nodeType: "Input"},
    {nodeID: 2, nodeType: "Hidden"},
    {nodeID: 3, nodeType: "Ouput"}
  ],
  connections: [
    {sourceID: 0, targetID: 2, weight: -1.1},
    {sourceID: 1, targetID: 2, weight:  1.4},
    {sourceID: 2, targetID: 3, weight:  0.5},
  ] }
```

```
{ nodes: { type: "array",
    parent : {type: "string"},
    nodeID: {type: "string"},
    content: {type: "string"}
  }
}
```

```
{ nodes: [
    { parent: " ", nodeID: "0", content: "+" },
    { parent: "0", nodeID: "1", content: "3" },
    { parent: "0", nodeID: "2", content: "*" },
    { parent: "2", nodeID: "3", content: "x" },
    { parent: "2", nodeID: "4", content: "x" }
  ]
}
```

Figure 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 (Stanley, 2007). Below, a GP-Tree (Koza, 1998) 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.

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.

As an experiment accumulates data, each object stored by WIN represents a potential branching point for future research with a traceable history of where the data originated. In this way, WIN aims to be a stepping stone accumulator for any evolutionary domain, allowing previously-finite experiments to exploit the possibilities of never-ending search.

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 artificial life 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.

Incoming users are first presented with an active list of domains hosted by WIN Online. A small text description is included for each domain allowing the user to choose the most relevant to their interests. For domains that run entirely in the browser users of WIN Online may seamlessly transition from browsing all domains to browsing the current collection of artifacts contained within the selected domain. Depending on the programming language of the domain, the user may need to download a desktop client to access the experiment. 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 *Saving Data in WIN*, 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/`.

Example Domains

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 posssibilities 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 (Secretan et al., 2011). The second is IESoR, a Sodarace-inspired (McOwan and Burton, 2005) domain for evolving two-dimensional morphologies capable of ambulation (Szerlip and Stanley, 2013). Some resulting phenotypes from both Picbreeder and IESoR are shown in figure 2. 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) (Takagi, 2001) 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.

Notably, IESoR was originally coded and simulated in JavaScript. For this demonstration, the IESoR simulation was rewritten in C++, and wrapped by a JavaScript WIN module. Writing the IESoR simulation in a non-scripting language demonstrates that even though the WIN system is written in JavaScript, WIN modules can be written in a native programming language like C++. Supporting native and scripting languages allows WIN modules to enjoy the performance benefits of native code when required while taking advantage of the ease of scripting when optimal performance

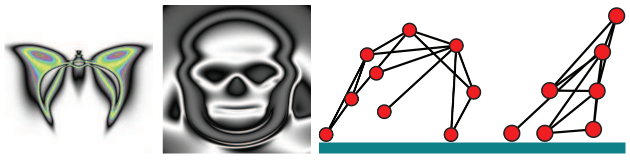

(a) Picbreeder (b) IESoR

Figure 2: **Products of Evolution.** Picbreeder artifacts (a) are n x n pixel images where the RGB values are constructed from the outputs of a CPPN (Stanley, 2007). (b) A pair of two-dimensional ambulating creatures are shown from the IESoR domain (Szerlip and Stanley, 2013).

is not necessary.

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 (Stanley, 2007). 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.

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 in 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 *Saving Data in WIN*, 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 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. As the user explores the domain through the IEC interface, whenever an artifact must be displayed, an event is passed to WIN 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.

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.

IESoR

In the original IESoR, an automated NSGA-II multiobjective search (Deb et al., 2002) evolved functional two-dimensional ambulating morphologies similar to the creatures depicted in figure 2b. 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 (2014) 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 whenever it is appropriate.

To enable user interaction with an automated search, a new WIN module named win-NSGA was created to handle the complexity of this interleaving search. Following precedent in Woolley and Stanley (2014), 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 (2014).

Altogether, the final program is a collection of HTML5 user interfaces connected through Node.js to the IESoR simulations being run in C++. 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.

WIN Phylogenies

An important element of 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. To save discovered artifacts, the user clicks a publish button on the user interface that dispatches an event to WIN to store the chosen object in the WIN database. On the path to the published artifact, the user has likely made multiple selection choices or run several evolutionary searches before choosing to publish. WIN allows the researcher to configure how much from these intermediate choices and associated meta-information is stored in WIN.

Recall that regardless of configuration, WIN always tracks parent-child relationships among the published artifacts. As users interact with a domain, each published object adds a single branch to the tree of artifacts inside the database. At any point during the ongoing experiment, researchers may compile part or all of the published objects into an artificial phylogeny. WIN assists in constructing phylogenies by providing methods for retrieving the full parent and children objects for any artifact within the database. Researchers can repeatedly query these methods to unravel a chain of research artifacts and construct a phylogeny.

To highlight this WIN feature, phylogenies for win-Picbreeder and win-IESoR are shown in figure 3a and 3b, 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 complete win-IESoR interface for new public contributions is under construction.

In contrast to traditional experimental results, the results in this section do not embody the conclusion of an experiment. Instead, these two domains help to exemplify the promise of WIN to aid researchers in creating experiments that may potentially never end. The resulting phylogenies are an important step towards validating that the WIN platform can deliver on this promise.

Discussion and Future Work

WIN is currently a prototype that will be brought to eventual maturity through more documentation, complete illustrated examples, and a sophisticated WIN module manager. As the WIN library matures, the documentation will be updated to reflect the nature of the library. Similarly, an increasing collection of coding examples will provide a consistently improving reference to developers working with the platform.

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 could be possible. The lightweight design of WIN aspires to create a new type of research collection full of open domains assembled together for easy and perpetual access through WIN Online. Moreover, WIN opens the door to new and expansive research questions. WIN empowers us to start imagining what it would be like one day to have a living archive of research where every object from every evolutionary experiment in progress is actively preserved with the full context of its ancestry known and its future open to exploration. What phenomena might emerge from the data when evolution is no longer run by labs in isolation for days or weeks, but by many labs in parallel across the world for years?

Conclusion

This paper represents 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. Similarly, WIN Online aggregates domains built with the WIN platform into a single source for new and exciting research that allows users to browse and participate. The WIN library demonstrates that it is indeed possible to construct an online and interactive platform around almost any evolutionary domain, which may help to proliferate the availability of experiments that continually collect potential stepping stones for extension and thereby effectively never end.

Acknowledgments

This work was supported by the National Science Foundation under grant no. IIS-1002507. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors(s) and do not necessarily reflect the views of the National Science Foundation.

References

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)*, pages 521–528, New York, NY. ACM Press.

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

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, pages 159–166, New York, NY, USA. ACM.

Channon, A. (2001). *Evolutionary Emergence: The Struggle for Existence in Artificial Biota*. PhD thesis, University of Southampton.

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

Crowston, K. and Prestopnik, N. R. (2013). Motivation and data quality in a citizen science game: A design science evaluation. In *2013 46th Hawaii International Conference on System Sciences (HICSS)*, pages 450–459. IEEE Press.

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

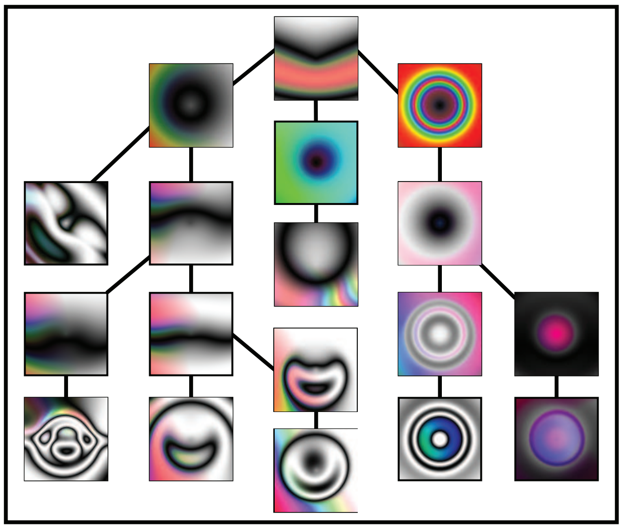
Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

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

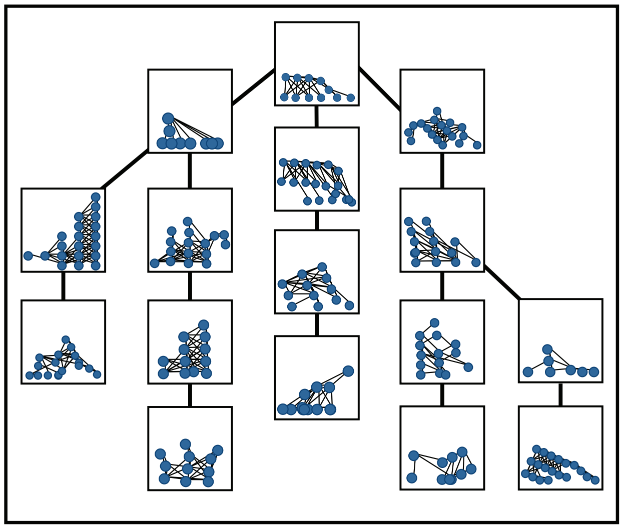
Green, D. (2007). Vlab website. The VLAB website is is publicly available at http://vlab.infotech.monash.edu.au/.

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.

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)*.



(a) win-Picbreeder



(b) win-IESoR

Figure 3: **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/`.

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

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

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

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

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

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

Newman, G., Wiggins, A., Crall, A., Graham, E., Newman, S., and Crowston, K. (2012). The future of citizen science: emerging technologies and shifting paradigms. *Frontiers in Ecology and the Environment*, 10(6):298–304.

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

Raff, R. A. (1996). *The Shape of Life: Genes, Development, and the Evolution of Animal Form*. The University of Chicago Press, Chicago.

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.

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.

Sims, K. (1994). Evolving 3D morphology and behavior by competition. pages 28–39. MIT Press, Cambridge, MA.

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.

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

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.

Takagi, H. (2001). Interactive evolutionary computation: Fusion of the capacities of EC optimization and human evaluation. *Proceedings of the IEEE*, 89(9):1275–1296.

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*, pages 957–964, Dublin, Ireland. ACM.

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.