DOI: http://dx.doi.org/10.7551/978-0-262-31050-5-ch024

An ecology-based evolutionary algorithm to evolve solutions to complex problems

Sherri Goings[1], Heather Goldsby[2], Betty H.C. Cheng[3], and Charles Ofria[3]

[1]Computer Science Dept., Carleton College, MN, 55057
[2]University of Washington, Seattle, WA, 98195
[3]Deptartment of Computer Science & Engineering, Michigan State University, East Lansing, MI, 48824
sgoings@carleton.edu

Abstract

Evolutionary algorithms have shown great promise in evolving novel solutions to real-world problems, but the complexity of those solutions is limited, unlike the apparently open-ended evolution that occurs in the natural world. In part, nature surmounts these complexity barriers with ecological dynamics that generate a diverse array of raw materials for evolution to build upon. The authors previously introduced *Eco-EA*, an evolutionary algorithm that integrates these natural ecological dynamics to promote and maintain diversity in the evolving population. Here, we apply the Eco-EA to the real-world software engineering problem of evolving behavioral models for deployed nodes in a remote sensor network for flood monitoring. We show that the Eco-EA evolves good behavioral models faster than a traditional EA, generates a more diverse suite of models than a traditional EA, and creates models that are themselves more evolvable than those created by a traditional EA.

Introduction

Evolutionary algorithms (EAs) have shown great promise in evolving novel solutions to real-world problems, but the complexity of those solutions is limited, unlike the apparently open-ended evolution that occurs in the natural world. In part, nature surmounts these complexity barriers with natural ecological dynamics that generate an incredibly diverse array of raw materials for the evolutionary process to build upon, the efficacy of which has been demonstrated in the artificial life system Avida (Cooper and Ofria, 2002).

For EAs to solve more complex problems, we must study how highly complex traits arise in the natural world, and where EAs fall short in duplicating these dynamics. The complexity of solutions produced by traditional EAs is typically limited by rapid convergence to a single solution on a sub-optimal local peak, resulting in stagnation. EA researchers recognize the importance of maintaining variation in evolving populations to prevent stagnation and make use of a variety of diversity preserving techniques. However, it has proven difficult to reach the levels of species density and variety found in nature (such as in bio-films (Tyson et al. 2004) or biodiversity hotspots like rain forests (Gaston 2000)) or even the high intra-species variance of individual values for a given trait. In nature, simple ecological forces promote this diversity, due to both spatial and temporal environmental heterogeneity, combined with negative frequency-dependent selection (Tilman, 1982).

Diversity in a population can provide other significant advantages beyond forestalling stagnation. Potential benefits to evolutionary algorithms include: (1) maintenance of a selection of good solutions for the researcher to choose from, often with slightly different properties; (2) representation across a Pareto front for multi-objective optimization problems; (3) the use of different partial-solutions as starting points to build the full solution from, without the researcher needing to know the ideal path; (4) resilient solutions that can withstand environmental changes; and (5) significantly more rapid evolution of targeted complex functions. Robust ecological communities exhibit all of these traits.

The authors previously introduced a method to integrate ecological factors promoting diversity into an EA using limited resources, and showed that populations evolved with this method were able to find and cover multiple niches in a simple string matching problem (Goings and Ofria, 2009). Here, we apply this new ecology-based evolutionary algorithm (Eco-EA) to a real-world problem in software engineering, and show that this approach yields several advantages over a traditional EA, including:

1. faster evolution of satisfactory solutions
2. evolution of a more diverse array of solutions
3. creation of solutions with greater evolvability that are easily adapted to succeed in different environments.

These results indicate that the ecology-based EA facilitates the evolution of solutions to complex problems.

Background

Eco-EA

As demonstrated by (Cooper and Ofria, 2002), forcing individuals to compete for multiple limited resources will force a population to maintain higher levels of diversity. A traditional EA can be thought of as having only one resource, where each individual's fitness is determined by the amount of that resource it can obtain. In most cases, the population size in an EA is fixed, thus making space its only limited resource (which organisms claim as they replicate). In the Eco-EA proposed by the authors in (Goings and Ofria, 2009), each function performed by an individual is associated with a distinct resource. When an individual performs a function it

receives a predetermined fraction of the currently available amount of the associated resource and its fitness is increased proportionately. These resources are set up as the computational equivalent of a well-stirred chemostat; that is, each resource flows into the environment at a constant rate, and a small percentage of the available resource flows out, limiting the total accumulation. Exploration of new areas of the fitness landscape is highly rewarded as an unused resource will accrue in quantity; as such, the individual first to discover the resource will receive a large fitness boost. However, when many organisms perform functions that consume the same resource, the availability of that resource will decrease until further organisms who attempt to draw from it do not receive enough reward to offset the opportunity cost of targeting a different resource.

Eco-EA in Avida

The experiments performed in this study used the Avida digital evolution research platform (Ofria and Wilke, 2004). Avida maintains a population of asexual self-replicating computer programs ("digital organisms") that exist in a computational environment and are subject to mutations and natural selection. Each digital organism has a genome that is a sequence of instructions in a special-purpose programming language. As in natural organisms, this genome specifies the behavior of the individual. Typically, in Avida, this behavior includes the replication of the organism, but for this study we used an explicit fitness function and organisms were replicated in time inversely proportional to their fitness (i.e. higher fitness yields faster replication), similar to the process of a steady-state evolutionary algorithm. This change removed the extra selection pressure for organisms to improve their replication mechanism and simplified the analysis of individual organisms. Random mutations occur during replication and include substitutions, insertions, and deletions. The Avida instruction set is designed so that mutations always yield a syntactically correct program, albeit one that may not perform any meaningful computation. When an organism replicates, its offspring replaces a randomly chosen individual currently in the population. Thus Avida maintains a constant population size.

The environment used in this study contains a set of resources, each of which corresponds to a user-defined task. An organism must perform a task to receive a portion of the available corresponding resource. The fitness of an organism is determined by how much of each resource it consumes. In most Avida studies, as with most evolutionary algorithms, resources are unlimited, creating a single-niche environment where an organism receives a fixed amount of resource for each task completed. Thus, the fitness gained for completing a task is constant and does not reflect how many other organisms are also performing that task. In this study, however, we incorporate the ecological factor of limited resources to create a multi-niche environment that encourages the evolving population to diversify.

Avida-MDE

For this study, we use a software engineering extension to Avida called Avida-MDE (Avida for Model-Driven Engineering), previously developed by Goldsby and Cheng (Goldsby and Cheng, 2008a). We briefly describe the motivation for the creation of Avida-MDE, establish its links to real-world problems, and provide a high-level overview of how it uses Avida to automate software engineering research.

Model-driven engineering is a leading software engineering approach to developing complex software-based systems, including on-board control software for automotive and flight systems, ecosystem monitoring, and robotic systems. Many of these systems are considered high-assurance, meaning that they must satisfy safety requirements under a variety of environmental conditions. Model-driven engineering works by systematically refining graphical models that can be analyzed for adherence to requirements using a variety of analysis tools, and then automatically used to generate code (Schmidt, 2006). Konrad *et al.* have proposed a modeling and analysis process for such high-assurance systems (Konrad et al. 2007) where a system is represented by a class diagram that captures the structural elements and several behavioral models. A given behavioral model comprises a set of state diagrams, one for each class in the class diagram, and represents the behavior of the system under specific environmental conditions.

Manually developing the behavioral models for a system can be tedious and error prone, since each model must be created independently and it requires the developer to have foreknowledge of the possible environmental conditions. Avida-MDE is a digital evolution tool that automates this process by generating a suite of behavioral models given information from the class diagram (Goldsby and Cheng, 2008b). At a high level, Avida-MDE accepts a list of triggers, guards, and actions (created using class diagram elements) as input. These inputs are provided to each digital organism, which uses them as raw material for constructing a set of state diagrams. A new genetic language was implemented in Avida-MDE to enable organisms to manipulate the state diagrams and thus change the behavior of the model it generates. The details of this language and how the digital organisms generate models can be found in (Goldsby and Cheng, 2008b). The key concept is that a mutation to an organism's genome changes the behavioral model that it creates.

To evaluate the generated behavioral models (and thus the organisms themselves), Avida-MDE uses a suite of software engineering tools. Several tasks were added to the Avida environment, which have previously been linked only to unlimited resources. *Software engineering metric tasks*, such as minimizing the number of transitions and maximizing the number of deterministic states, guide the evolutionary process to generate models that adhere to commonly advocated software engineering practices. *Scenario tasks* reward organisms for creating models that support one desired execution path, or scenario. Scenarios encapsulate small excerpts of model behavior that can be combined and expanded to achieve the desired overall system behavior. To account for the uncertainty in the execution environment, a developer can specify two types of scenarios; (1) *required functional scenarios* must be supported by the generated models; (2) *non-functional (NF) scenarios* each of which specify a different way to achieve the same functional objective with different non-functional characteristics (e.g., quality, reliability). A model must support at least one of each

type of NF scenarios. The specific NF scenario supported by a model impacts its non-functional behavior. Next, *witness property tasks* reward models for having at least one execution path that supports a desired system property. Lastly, *property tasks* are included to reward models for having all possible execution paths support a desired system property. For example, "no data is ever lost," "battery levels never drop below a threshold value," or "water level never exceeds a maximum value."

Grid-Stix

Avida-MDE was previously used to generate behavioral models for Grid-Stix, a light-weight flood warning system that comprises a set of sensor nodes. Grid-Stix is used to monitor the water levels for potential flood conditions with the River Ribble in England (Hughes et al. 2006). Flooding is an increasing and costly problem for the United Kingdom, and early flooding predictions enable fast responses to avert flood damage. However, prediction accuracy must be balanced by two other non-functional considerations: energy efficiency (because sensor nodes have a limited power supply) and fault-tolerance (because sensor nodes are deployed remotely). The objective of the case study was to generate a suite of behavioral models for a single sensor node, where the models make different non-functional tradeoffs (i.e., different combinations of energy efficiency, prediction accuracy, and fault-tolerance) and yet all satisfy the overall functional objective of monitoring the river to collect data and pass it along to nearby nodes.

Different scenario tasks captured different non-functional tradeoffs. Specifically, three tasks rewarded models that supported scenarios for setting different processor speeds while completing various functions on the sensor, and six tasks rewarded models that supported scenarios where the sensor used different data transmission methods. A model needs to only have one path that performs a scenario behavior in order to receive the associated reward, and can receive a partial reward for partial completion of a scenario. For example, one scenario required a node to set its processor speed to 100, then query the pressure sensor at this speed for the water depth, and finally to set its depth data to the query result. A model received 50% of this scenario task reward if it set its processor speed to 100, 75% if it also queried the pressure sensor, and 100% if it completed the entire scenario.

Witness and property tasks built upon the scenario tasks to reward for desired overall system behavior; for example sending flood predictions based on current water depth. This prediction-sending witness task rewarded organisms that developed models that contained an execution path that checked the water depth, calculated a prediction, and transmitted that prediction. The associated property task only rewarded a model if every possible execution path performed that same behavior. Checking if a model supported a scenario was simple and quick, however checking if a model satisfied a witness or property task was difficult and time-intensive; in the worst case all possible execution paths of the model had to be checked.

To avoid unnecessary witness and property task checking, models were required to support a minimum set of scenarios before they were even considered as candidates for satisfying overall system properties. For example, a model could not perform the previous witness/property example of sending a prediction based on current water depth if it did not use some method to check the water depth and successfully send its prediction. Thus, there was no reason to check for this system property unless a model supported one scenario associated with each of those behaviors. In fact, to satisfy any of the Grid-Stix behavioral requirements, a model needed to support one of each of the scenario alternatives (i.e., one processor speed and one transmission method), as well as 3 other required scenarios. These combinations of the 3 processor speed scenarios and 6 transmission method scenarios yielded 18 possible behavioral models or phenotypes, each of which represented a different combination of the non-functional properties (energy efficiency, prediction accuracy, and fault-tolerance). Although the previous Avida-MDE study successfully generated satisfactory behavioral models that represented some of the phenotypes, diverse models were found only by evolving many separate populations (the original study evolved 40 separate populations each with 3,600 individuals), and still the experiments were unable to discover all 18.

Experiments and Results

Generating a diverse suite of models

Our first objective is to assess how well the Eco-EA version of Avida-MDE performs compared to the original, single-niche version of Avida-MDE. The Grid-Stix problem provides an excellent case study for comparison, since one of the desired outcomes is to generate a suite of models, each of which minimally satisfies the required properties specified by the developer, but may also contain additional behavior that makes it suitable for domains that were not explicitly provided. A simple way to determine what additional behavior a model may possess is to consider which scenario it uses from each of the non-functional scenario sets. As described previously, there are 18 possible combinations of NF scenarios and therefore 18 unique phenotypes a model may represent, each of which yields a slightly different behavior in terms of energy efficiency, prediction accuracy, and fault-tolerance. The original version of Avida-MDE was unable to evolve all 18 possible phenotypes, even across 40 runs.

We compare the efficacy of the Eco-EA version of Avida-MDE in evolving a diverse suite of models to Goldsby and Cheng's previous results (Goldsby and Cheng 2008b). The key difference between the two approaches is how the NF scenarios are rewarded. In both versions of Avida-MDE, organisms can only receive a fitness gain for one scenario from each of the sets of NF scenarios (in the Grid-Stix study, one processor speed and one transmission method). If an organism supports multiple scenarios from a given set, then it is rewarded only for the first one it supports. In the original Avida-MDE, all tasks in the environment, including these scenario tasks, add a fixed amount to an organism's fitness when they are performed. In the Eco-EA version, each NF scenario task corresponds to a limited resource in the environment. When an organism performs one of these scenario tasks it consumes a fraction of the available resource, reducing the amount of that resource available to other

organisms. The fitness gain the organism receives is proportional to the amount of resource it consumes. This resource-dependent fitness encourages organisms to evolve to support little-used scenarios, and creates an overall diverse population of models in terms of non-functional properties. The rest of the Avida-MDE tasks (including the required scenarios) are still rewarded in the Eco-EA using the standard fixed-reward method; these tasks represent properties and behavior required in all models and therefore we want them to confer a constant fitness gain regardless of the number of other individuals performing the same tasks.

We perform 2 sets of 20 experiments, one set in each version of Avida-MDE. Slight improvements made to the original Avida-MDE after the previous results were published necessitated re-running the initial experiments in order to fairly compare the results of the Eco-EA version of Avida-MDE. We ran each experiment for 25,000 updates (updates are units of time in Avida that are roughly proportional to generations) or 24 hours, whichever came first.

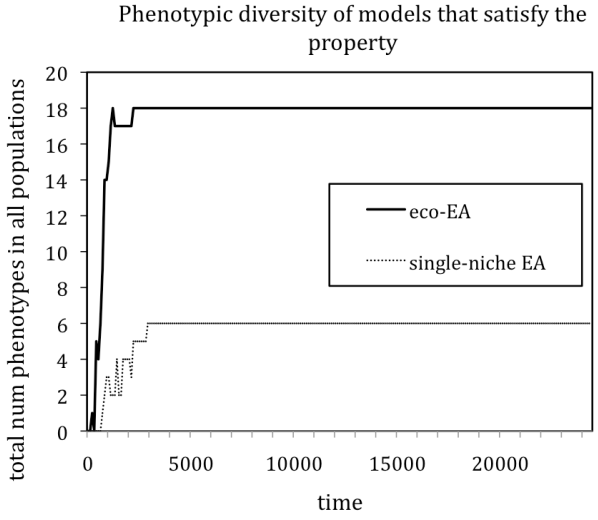

Figure 1. The number of unique phenotypes of models that satisfy the property in terms of non-functional property trade-offs found by all 20 runs in each environment over time. In this Grid-Stix problem there are 18 possible combinations of transition each of which results in different non-functional behavior in the models. In the Eco-EA (limited resource environment), invariant-satisfying models representing each of the 18 non-functional phenotypic possibilities quickly evolve. In the tradition EA (single niche environment), models satisfying the invariant evolve more slowly and fewer of the non-functional based phenotypes are found even after a long period of evolution. Each experiment evolves a population of 1,000 individuals for 24 hours or 25,000 updates, whichever comes first.

As discussed above, checking property and witness tasks is time-consuming, leading populations to become very slow in Avida time once many individuals satisfy the requirements to be checked for these tasks, so the absolute 24 hour time limit is imposed as well. In this pair of experiments all of the 20 Eco-EA replicates evolve to satisfy the property task and reach the 24 hour limit, ending between 1,000 and 5,000 updates. Ten of the single-niche EA replicates reach the 24 hour limit (the 9 that evolve the property task and one other that has models being checked for the property though it never evolves), ending between 2,000 and 23,000 updates, and the other 10 end at the 25,000 update cutoff.

We find that the Eco-EA version of Avida-MDE not only generates a more diverse suite of final model phenotypes, but that it also evolves models satisfying the required functional property significantly faster than the traditional, single-resource approach. Figure 1 shows the number of total unique phenotypes of models satisfying the required property found across 20 Avida experiments over time. The Eco-EA finds models satisfying the property before reaching 1,000 updates of evolution (~400 generations), and all 20 replicates find models by 5,000 updates. Across all 20 replicates the Eco-EA finds property-satisfying models of each of the 18 non-functional phenotypes within 2000 updates of evolution (~800 generations). In contrast, the traditional approach using a single niche only finds any model satisfying the required property in half of the replicates, and even in those that do find a satisfactory model the average time one is found is three times as long as in the Eco-EA (5000 updates vs. 1500 updates). Even after 25,000 updates of evolution the single niche approach finds property-satisfying models representing only 6 of the 18 possible phenotypes.

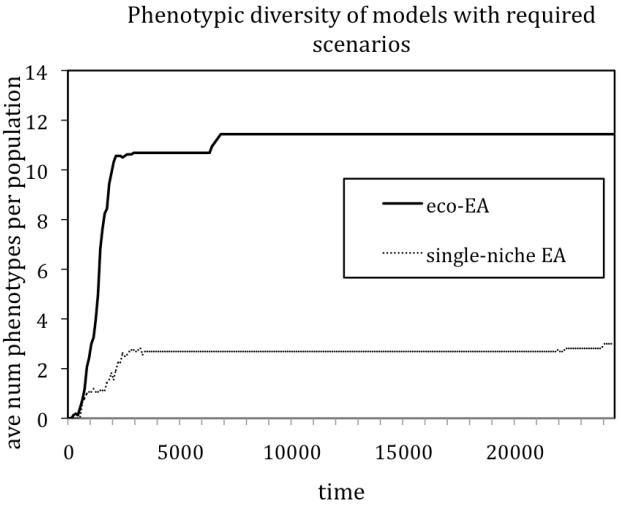

Figure 2. The average number of unique phenotypes of all models in each population in terms of non-functional properties. Eco-EA populations quickly diversify to cover most of the possible phenotypes well before evolving models that satisfy the property, while the single-niche EA is stuck on just one or two phenotypes per population. This means there are less evolutionary paths to find a model satisfying the property in the single-niche EA, and hence it takes longer. Each experiment evolves a population of 1,000 individuals for 24 hours or 25,000 updates, whichever comes first.

The Eco-EA version of Avida-MDE also yields a significantly more diverse set of models in each individual experiment than the single-niche EA. Every one of the 20 experiments using the Eco-EA yielded property-satisfying models. The final populations contained coexisting models representing between 8 and all 18 different phenotypes, with a mean of 14.8 phenotypes per population. In contrast, only 9 of the 20 single-niche Avida-MDE experiments evolved any property-satisfying models, with a maximum of 4 phenotypes in a single population. The average number of phenotypes found in the final populations of single-niche EA experiments was 2.85 ($p<.001$ comparing 2.85, $s=3.7$ to 14.8, $s=2.9$, with 38df, using the independent group t-test for means).

One could argue that since we know all 18 target phenotypes, we could simply evolve each of them in independent populations. However, there are several reasons we would expect this seemingly simpler method would not perform as well as Eco-EA. First, the Eco-EA is more generalizable to other problems; in many cases, developers will not know *a priori* what novel behavior a model may evolve and thus it is not always possible to enumerate the desired phenotypes. Second, the complex behavior required for a model to satisfy the required functional properties must be built on simpler behavior such as supporting scenarios. We posit that rewarding for many scenarios yields more potential pathways for evolution to follow in finding a model that satisfies the property. Once a single property-satisfying model is found, it may be possible for that model to change its non-functional behavior while still maintaining the required behavior.

The theory that the inclusion of more scenarios yields more evolutionary pathways and thus leads to faster evolution also may explain why the Eco-EA finds models satisfying the developer's requirements faster than the single-niche EA. Figure 2 shows the average number of unique phenotypes (based on NF scenarios) of all models in each population, including those that do not satisfy the required property. To test this theory we performed experiments where instead of including tasks for all of the NF scenarios in the environment, we included only one scenario from each of the 2 sets, a single processor speed and a single transmission method. We performed 5 replicates of each of the 18 environments thus created, for a total of 90 experiments (as compared to the 20 performed including all of the scenarios). We found that when only rewarding for a single phenotype, no model satisfying the required behavioral property ever appeared. The Eco-EA populations diversify quickly to contain individuals of almost all of the phenotypes in each population, while the single-niche populations are stuck on just one or two of the possible phenotypes, giving evolution fewer possible paths to a model satisfying the property.

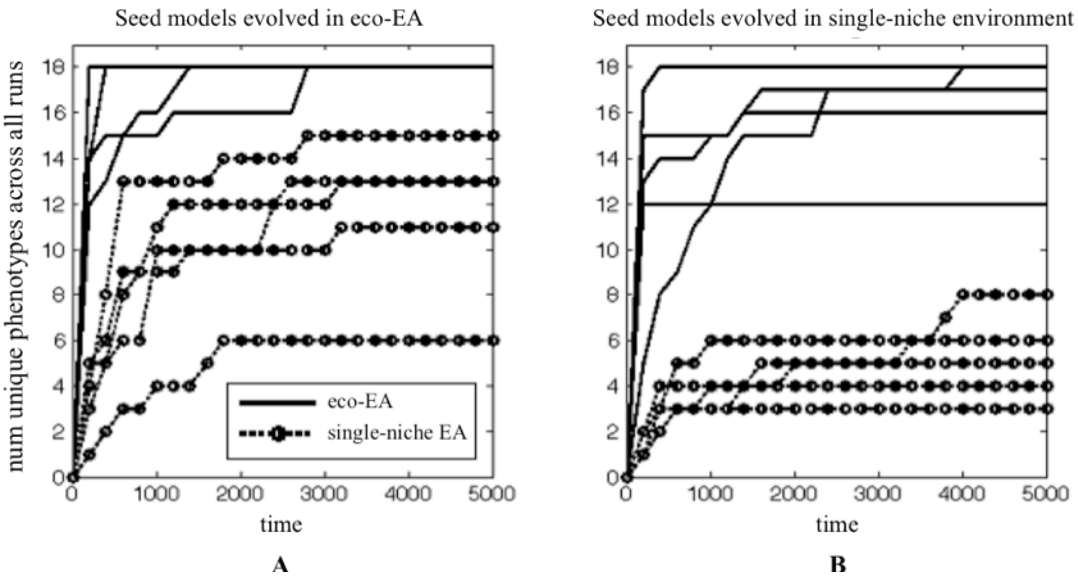

Figure 3. The number of unique phenotypes of models that satisfy the property found by all 20 runs for each treatment over time. (A) Performance of each version of Avida-MDE when seeded with each of the 5 models originally evolved in the Eco-EA environment. While the individual models yield highly varying results, the Eco-EA quickly evolves all 18 phenotypes no matter which of the 5 it is seeded with. The single-niche environment is never able to find all 18 phenotypes. (B) Similar results occur when populations are seeded with models originally evolved in the single-niche environment. The Eco-EA now only generates all 18 phenotypes for 2 of the initial models, but still generates more phenotypes in the worst case (12) than the single-niche EA generates in the best case (8). Each experiment evolves a population of 1,000 individuals for 24 hours.

Evolvability of Models

A common situation is for a developer to have already developed one model suited to a given set of conditions, and needs a suite of models appropriate for a variety of condition domains. We therefore compared the evolving population of the Eco-EA version of Avida-MDE to that of the single-niche EA when the population is initially filled with copies of one individual that builds a model already satisfying the required behavior.

We randomly selected 5 individuals that generated models satisfying the required property from those evolved using the Eco-EA version of Avida-MDE, with the specification that they each come from a different replicate population and each represent a different non-functional phenotype. We then did the same with the models evolved using the original Avida-MDE, ensuring that we chose the same 5 phenotypes as the former set. For each of the 10 chosen models, we used the model to seed the initial populations of 20 replicate experiments where we continued evolution in the Eco-EA environment, and 20 where we continued evolution in the original single-niche environment.

We find two key results; 1) the Eco-EA environment generates a more diverse suite of models more quickly than the original single-niche environment; 2) the individuals evolved in the Eco-EA environment appear to be more evolvable in terms of generating diverse phenotypes than those evolved in the single-niche environment. Figure 3 shows that the Eco-EA version of Avida-MDE quickly generates diverse populations representing models of many (and often all) phenotypes no matter which model the population is seeded with, while the single-resource EA tends to only evolve phenotypes close in genetic space to that of the initial model.

It also appears that models originally evolved in the Eco-EA environment yield more diverse phenotypes in either environment when they are used to seed the initial population; the Eco-EA generates all 18 possible phenotypes when seeded with any of the 5 models initially evolved using the Eco-EA, and the single-niche EA generates over 11 phenotypes when seeded with 4 of these models, while the most it ever finds when seeded with models initially evolved in the single-niche environment is 8 phenotypes. The increased evolvability of models initially evolved in the Eco-EA version of Avida-MDE can be seen more clearly in figure 4, where the average results across all 5 seed models are shown for each of the 4 treatments.

Once again we find that the Eco-EA version of Avida-MDE not only evolves a more diverse set of phenotypes more quickly than the single-resource approach across sets of all 20 runs, but it also yields higher diversity in individual runs. When averaging all runs across all 10 seed models, the Eco-EA evolves an average of 17.1 phenotypes per run, while the single-resource EA evolves an average of only 8.4 phenotypes ($p<.001$ comparing 17.1, s=1.25 to 8.4, s=2.7 using the independent group t-test for means).

The individual run diversity also differs based on which environment the seed models were evolved in. Averaging all runs from both environments when seeded with the 10 models evolved using the Eco-EA, 14.8 unique phenotypes are generated per run, vs. 10.7 phenotypes per run when populations are seeded with the models evolved in the single-niche environment ($p<.001$ comparing 14.8, s=1.7 to 10.7, s=2.2 using the independent group t-test for means).

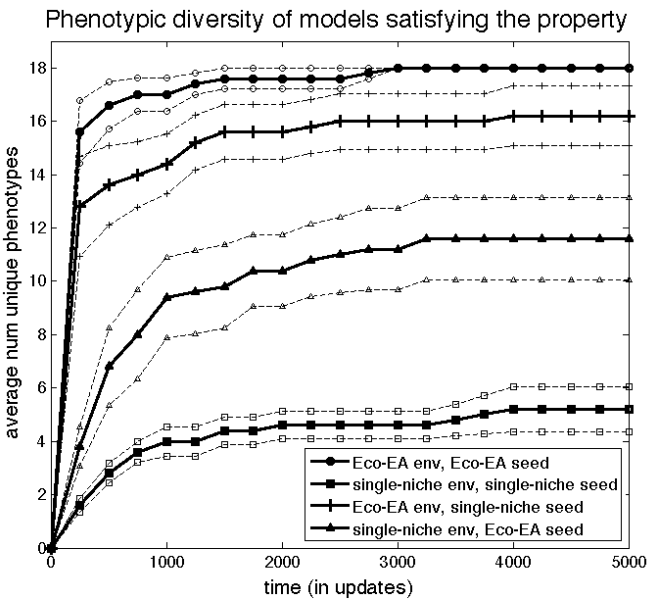

Figure 4. Average of data with error bars (+/- 1 standard error) for each of 4 experimental treatments (All combinations of 2 types of seed models; those evolved in the Eco-EA environment or those evolved in the single-niche environment, and 2 environments for continued evolution; the Eco-EA and the single-niche). The line for each treatment represents the average of the 5 sets of experiments, one for each model used in that treatment. The data for each of the 5 sets is the number of unique phenotypes found by all 20 populations in that set over time. The Eco-EA finds on average a more diverse set of models than the single-niche EA no matter which type of models it is seeded with. Both environments find a significantly more diverse set of models when seeded with models initially evolved using the Eco-EA than those evolved using the single-niche EA.

This result is something we would like to explore more thoroughly, as we can not yet identify what exactly makes the models evolved in the Eco-EA environment more able to diversify in any environment during further evolution. We found that one of the models evolved by the Eco-EA actually represented multiple phenotypes itself, as it stochastically performed one of 2 different options for the transmission scenario. However the other 4 models did not show this behavior so that cannot explain the overall result. Hypotheses we would like to test include that the Eco-EA evolved models could do well when they switch frequently between performing different scenarios, and so there may be selective pressure for them to be only one or two mutations away from performing a different set of scenarios at any given time.

Conclusion

In this paper, we compared the performance of Eco-EA to a more traditional EA (Avida-MDE) on a complex software engineering problem. Specifically, we used both Eco-EA and Avida-MDE to generate software models for a flood warning system. For this problem, there were 18 possible models (phenotypes) that all met the functional system objectives (i.e., detect flooding), but did so using a variety of different non-functional tradeoffs. Eco-EA provided three significant advantages over Avida-MDE. First, Eco-EA more rapidly evolved organisms that generated models that satisfied the developer's requirements. Second, Eco-EA evolved a more diverse set of solutions that represented models with different properties. Lastly, when the models created by Avida-MDE and Eco-EA were used as seeds for subsequent experiments, the solutions created by Eco-EA exhibited greater evolvability. These results indicate that the Eco-EA facilitates the evolution of solutions to complex problems.

In the future, we plan to apply Eco-EA to complex problems in different domains. One potentially interesting area of investigation is problems whose solutions may require explicit cooperation among the various species present within the population. Additionally, we are working on extending Eco-EA to other areas of evolutionary computation, such as natural problem decomposition and multi-objective optimization.

Acknowledgements

This work has been supported in part by NSF grants CCF-0541131, CCF-0820220, CCF-643952 and CCF-0523449, DBI-0939454, Army Research Office grant W911NF-08-1-0495, Ford Motor Company, the DARPA "Fun Bio" program (HR0011-05-1-0057), and a Quality Fund Program grant from Michigan State University. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, U.S. Army, Ford, or other research sponsors. The authors also thank C. Adami and everyone in the Digital Evolution Lab at Michigan State for their support, ideas, and critiques contributed to this research, especially Dave Knoester.

References

Cooper, T.F. and Ofria, C. (2002). Evolution of stable ecosystems in populations of Digital Organisms. In *Artificial Life VIII,* pages 227-232. MIT Press, Cambridge, MA.

Tyson, G.W. et al. (2004). Community structure and metabolism through reconstruction of microbial genomes from the environment. *Nature,* 428:37–43.

Gaston, K. J. (2000). Global Patterns in Biodiversity. *Nature* 405:220-227.

Tilman, D. (1982). *Resource Competition and Community Structure*. Princeton University Press, Princeton, NJ.

Goings, S. and Ofria, C. (2009). Ecological Approaches to Diversity Maintenance in Evolutionary Algorithms. In *IEEE-Alife*, pages 124-130. Published by IEEE.

Ofria, C. and Wilke, C.O. (2004). Avida: A Software Platform for Research in Computational Evolutionary Biology. *Artificial Life,* 10:191-229.

Goldsby, H. J. and Cheng, B. H. C. (2008a). Automatically Generating Behavioral Models of Adaptive Systems to Address Uncertainty. In *ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MoDELS)*, pages 568-583. Springer-Verlag Berlin, Heidelberg.

Schmidt, D.C. (2006). Model-Driven Engineering. *IEEE Computer*, 39:2.

Goldsby, H. J. and Cheng, B. H. C. (2008b). Avida-MDE: A Digital Evolution Approach to Generating Models of Adaptive System Behavior. In *Genetic and Evolutionary Computation Conference,* pages 1751-1758. ACM, New York City, NY.

Hughes, D., Greenwood, P., Coulson, G., Blair, G., Pappenberger, F., Smith, P. and Beven, K. (2006). An intelligent and adaptable flood monitoring and warning system. In *5th UK E-Science All Hands Meeting*.

Konrad, S., Goldsby, H., and Cheng, B. H. C. (2007). i2MAP: An Incremental and Iterative Modeling and Analysis Process. In *ACM/IEEE Int. Conference Model-Driven Engineering Languages and Systems*, pages 451-466. Springer, New York City, NY.