Chapter 5
Applying Ecological Principles to Genetic Programming

Emily Dolson and Wolfgang Banzhaf and Charles Ofria

Abstract In natural ecologies, niches are created, altered, or destroyed, driving populations to continually change and produce novel features. Here, we explore an approach to guiding evolution via the power of niches: ecologically-mediated hints. The original exploration of ecologically-mediated hints occurred in Eco-EA, an algorithm in which an experimenter provides a primary fitness function for a tough problem that they are trying to solve, as well as ”hints” that are associated with limited resources. We hypothesize that other evolutionary algorithms that create niches, such as lexicase selection, can be provided hints in a similar way. Here, we use a toy problem to investigate the expected benefits of using this approach to solve more challenging problems. Of course, since humans are notoriously bad at choosing fitness functions, user-provided advice may be misleading. Thus, we also explore the impact of misleading hints. As expected, we find that informative hints facilitate solving the problem. However, the mechanism of niche-creation (Eco-EA vs. lexicase selection) dramatically impacts the algorithm's robustness to misleading hints.

Emily Dolson
BEACON Center for the Study of Evolution in Action and Department of Computer Science and Ecology, Evolutionary Biology, and Behavior Program, Michigan State University, East Lansing, MI, USA e-mail: dolsonem@msu.edu

Wolfgang Banzhaf
BEACON Center for the Study of Evolution in Action and Department of Computer Science, Michigan State University, East Lansing, MI, USA e-mail: banzhafw@msu.edu

Charles Ofria
BEACON Center for the Study of Evolution in Action and Department of Computer Science and Ecology, Evolutionary Biology, and Behavior Program, Michigan State University, East Lansing, MI, USA e-mail: ofria@msu.edu

5.1 Introduction

Natural evolution produces effective solutions to complex problems, often well beyond the ability of human engineers to duplicate. If we are to harness these natural evolutionary dynamics, we must understand the full depth of how they function and why they are so effective. In this paper, we explore how ecological factors promote more open-ended evolutionary systems that have a greater potential to produce complex, dynamic, and practical solutions to targeted problems. First we discuss why we believe that this approach can help solve difficult AI problems, then we review the current scientific understanding of ecological dynamics of interest. Next, we present experiments that we have performed using Eco-EA and lexicase selection to test the potential of ecologically-mediated hints, before discussing the implications of these results, and finally laying out the next steps we plan to take in this line of research.

5.1.1 Motivation

Many problems in machine learning center around creating artificial intelligence systems that can resolve challenging problems that are traditionally solved by humans. Often, these programs are written with the goal of mimicking the strategies employed by skilled humans. When human strategies can be clearly articulated as a set of well-defined rules, the process of writing the AI can be straight-forward. However, humans tend to rely on intuition when solving many types of problems. While human intuition is often effective, it is challenging to abstract into an algorithm that an AI can follow without missing important nuances. Attempts to do so often produce rigid AIs that fail to appropriately adapt to situations that are subtly different than those that were originally expected. Such issues are exacerbated for problems where early decisions determine what scenarios the AI will later encounter. For example, in many board games, minor mistakes early on can drastically reduce success and alter what options are available for the middle and end of the game.

This property—whereby early decisions shape what strategies are possible later—can pose even more substantial obstacles to evolving AIs. If a good strategy is critical early on, an AI without one would consistently lose. However if a good early strategy is not sufficient for overall success, the selection pressure in favor of it will be weak (at best) relative to its importance. Specifically, in any problem where many tasks must all be performed reasonably well for fitness to be non-zero, it is nearly impossible for evolution to get enough initial traction to be successful. Taken, from another perspective, the fitness landscape for such a problem will be almost entirely flat, with a small rugged region featuring steep peaks that are challenging to even find, let alone navigate. Previous attempts to make this landscape more easily navigable by giving the evolving AIs ”hints” based on strategies successfully employed by humans have generally been ineffective due to the aforementioned difficulties with abstracting human intuition.

We hypothesize that we can apply ecological dynamics to evolve models of human decision making and reliably create human-competitive AIs. Here, we present our concepts in terms of playing games, as this is an intuitive set of problems to think about, but the techniques we propose should generalize to other categories of problems. The key is to have evolution be responsible for the process of abstracting general strategy from human intuition. We believe that we can achieve this goal by supplying data on human decisions from a wide variety of scenarios and selecting for AIs that are capable of *predicting* the move that a given human made in any given situation. This approach will allow selection to operate evenly across early-, mid-, and late-game strategy, removing the issue of temporally compounded mistakes. Furthermore, it eliminates the need for humans to be able to codify their strategies into a concrete rule set.

Selecting for AIs that can predict human moves has a second, more fundamental benefit. Often, problems that are too complex for evolutionary computation to solve in isolation can be solved if there is also a fitness benefit for solving simpler, related problems. These simpler problems, often referred to as ”building blocks”, [**?**] cause genomes within the population to accumulate information that is relatively easy to repurpose into solving the actual problem [17], a technique that has proven to be effective in evolutionary computation [1]. We hypothesize that AI's evolving to predict a human's choices will often do so by recreating the underlying building blocks that comprise that human's strategy—even if the human does not recognize it themself.

An obvious problem with this approach is that some humans may be poor at a given problem. For example, predicting their moves in a game may be impossible because they are effectively random, or worse employing actively poor strategy that will lead the evolving population down a maladaptive path. Fortunately, Goings and Ofria previously developed an evolutionary algorithm that is resistant to bad advice: Eco-EA [7, 5, 6]. Eco-EA creates limited resources associated with tasks that are expected to be valuable on the way to a high-quality solution. This approach proved effective on example problems, in part because human intuition could be used to select the rewarded tasks and the algorithm would associate limited resources with them. As long as no organisms are performing the rewarded traits, the resources build up in the population (to a limit) and make the associated tasks more valuable. Once the resources are in use, however, their abundance dwindles until they supply fitness to only a small fraction of the population. Resources associated with useful sub-tasks should produce evolutionary building-blocks that get incorporated as parts of a more complex strategy and used in the overall solution. Unhelpful or misleading resources, however, should be used by only a small fraction of the population, resulting in a trivial slowdown as compared to running the algorithm without the resources present.

For example, if we wanted to evolve a controller for a robot that would perform search-and-rescue, we would never get there if we started with random controllers and only rewarded robots that successfully rescued people. We could, however, use ecologically-mediated hints by adding a set of limited resources that each rewarded some component, such as (1) moving to a target location, (2) exploring, (3) fully scanning an area, (4) identifying dangers, (5) navigating around obstacles, (6) identifying trapped people, (7) moving toward trapped people, (8) freeing trapped people, (9) moving with people, and (10) finding your way back to safety. With all of these helpers as building blocks, it is easy to imagine that a controller would eventually evolve that could occasionally find and return people safely, which would allow it to start getting rewarded by the primary (unlimited) fitness function. This approach uses ecologically-mediated hints to push the population to solve the problem from multiple directions at once; before the whole problem is solved, some individuals might be able to find people, but then ignore them. Some will explore and then find their own way back. Others might be able to free people they find, but not know what to do with them next. Some evolutionary trajectories might be more likely than others to continue to evolve toward the final goal. Of course, some of the "hints" might be associated with counter-productive tasks. If there were a limited resource associated with avoiding all dangers (because a misguided programmer thought this would be helpful), the result might be to make it less likely for the robot to find trapped people. With ecologically-mediated hints, however, only a small portion of the population (a single niche) would be rewarded for this ability, so others would still plow ahead and complete the mission.

The inspiration for this approach comes from the study of eco-evolutionary dynamics [12]. Ecology and Evolution are considered sister fields of study within Biology, often separated by the misconception that their dynamics occur on two different timescales. However, the lines between these disciplines have blurred as it has been acknowledged that ecological and evolutionary dynamics strongly influence one another [22]. Practitioners of evolutionary computation have recognized this relationship, but typically use ecological effects only for preserving diversity to prevent premature convergence on a sub-optimal solution and produce a wider range of solutions to choose among. In Artificial Life systems, however, ecology has also been linked to promoting open-ended evolution in the form of increased complexity, novelty, cooperation, distributed problem solving, and intelligence [26]. As such, we believe that richer ecological dynamics are a source of substantial untapped potential for evolutionary computation.

A key component of this approach is that in an ecology, niches drive the types of diverse solutions that appear. If an organism is the first to occupy a new niche, it must have some traits associated with that niche, but is otherwise free from direct competition, allowing it to sustain substantial loss of unrelated traits. This dynamic allows for many different pathways to coexist in a population, any of which can be followed to solve the high-level problem, with the most successful strategies dominating multiple niches. In essence, these co-existing niches facilitate the creation of a variety of building blocks leading to successively more complex strategies.

Another benefit to an eco-evolutionary approach is that ecological communities are, by definition, at least somewhat diverse. While promoting diversity in evolutionary computation has long been recognized as critical to avoiding the problem of premature convergence, most existing mechanisms to promote diversity in evolutionary computation select for solutions that are distinct from each other, regardless of other qualities [8]. In natural systems, however, diversity arises due to organisms filling niches, each requiring specific phenotypic traits for success. Thus there is pressure for a diversity of functional traits. Furthermore, new niches are continuously created in nature as organisms interact with each other and modify both their physical and social environment. In problem domains such as playing complex games, this diversity of solutions becomes even more important, as there is no single, deterministic, best strategy. Instead, there are various strategies that are effective in different situations and against different opponents, just as there would be in an ecological community.

5.1.2 *Ecological approaches in evolutionary algorithms*

A number of highly effective evolutionary-computation techniques owe their success to ecological dynamics. First, there is lexicase selection [23], a different approach to creating niches associated with various sub-problems, which has proven extraordinarily successful in genetic programming [11, 10]. In lexicase selection, a large number of test cases are used as criteria for evaluation. Each time an organism has to be selected to propagate into the next generation, the test cases are applied in a random order, filtering out all but the most fit candidates each time. Once a single candidate remains (or all of the test cases have been applied and a random individual is chosen from the final set) it is replicated into the next generation and the process is repeated. The fact the the ordering of these test cases continually changes means that solutions successful at different subsets of tasks are all able to co-exist. Although lexicase selection has traditionally been used with test cases, there should be no reason it cannot be used with multiple fitness functions instead to provide ecologically-mediated hints. In this way, any hint that could be given to a system like Eco-EA could just as easily be provided to lexicase selection as well.

From an ecological perspective, lexicase selection creates pressure for the population to diversify into different niches that are based on building blocks to a more complex problem. In order for an organism to be evolutionarily successful, it must be among the very best at at least one of the test cases/fitness functions. In most cases, it must be among the best at many of them (probably a related group). Thus, organisms compete with others within the niches created by each test case/fitness function. Eco-EA also promotes intra-niche competition, but in a subtly different way. Limited resources are used up as organisms receive fitness bonuses, meaning that they compete with other organisms benefiting from the same bonus. Thus, those with higher fitness will competitively exclude less fit organisms that depend too much on the same resources [2].

This distinction between competition in lexicase selection and in systems with limited resources has a few implications. First, limited resources allow for generalists that do not excel at any tasks but are decent at many. It is unclear whether this flexibility opens up additional pathways through the fitness landscape. Second, lexicase selection always exerts selective pressure to improve on all test cases/fitness functions, whereas limited resources create little incentive to expand into a niche that is already fully occupied. This dynamic may make lexicase selection less robust to receiving bad advice about how to solve a problem than other mechanisms for providing ecologically-mediated hints. Third, lexicase selection automatically incentivizes organisms that excel at uncommon combinations of test cases/fitness functions. Eco-EA could, however, be adjusted to mimic this property by adding additional resources. Ideally, we will be able to find a hybrid approach to providing ecologically-mediated hints that combines Eco-EA with lexicase selection to achieve the best of both worlds.

MAP-Elites is another successful algorithm that leverages ecological dynamics and user input [20]. In MAP-Elites, the user chooses axes along which they believe variation will be important. MAP-Elites then breaks the hyperspace defined by these axes into discrete regions. Each region can be occupied by at most one solution. When a new candidate solution is created, it is assessed on each axis and the corresponding bin is located. If the new candidate solution has a higher fitness than the previous occupant of that bin, it replaces that occupant. Otherwise, it is tossed out. Evolution proceeds by choosing occupants of various bins in the search space and mutating them to create new candidate solutions.

The sub-division of axes for MAP-Elites explicitly partitions the search space into niches in a way that has clear parallels to choosing tests to provide to lexicase selection or hints to associate with limited resources. The lack of directionality in MAP-Elites, however, makes it distinct from lexicase selection. At face value, it might seem to make it different from Eco-EA as well. Certainly this intuition is true to some extent - hints provided as limited resources do have directionality to them. However, the foundation of limited resources is negative frequency-dependence which, in addition to reducing the risk of the entire population being led astray, has the important effect of promoting diversity along the axis of the hint. This diversity goes beyond merely allowing a portion of the population to drift; niche partitioning should force the population to spread out along the axis in question to produce meaningful diversity that may be helpful in solving the overall problem.

There are a wide variety of other ecologically-inspired strategies that have proven to be effective for maintaining diversity in evolutionary algorithms. These largely fall into four categories: niching/speciation (e.g. [8, 25]), parent selection (e.g. [4, 18]), dividing the population into subpopulations (e.g. [14, 15]), and adjusting the objective function to favor diversity and/or novelty (e.g. [19]). Of these approaches, only niching/speciation consistently promotes stable coexistence of different types of strategy. Even among niching/speciation strategies, most emphasize a diversity of phenotypes rather than the diversity of evolutionary building blocks that Eco-EA and Lexicase Selection promote. While solutions built on different building blocks may often exhibit different phenotypes, they may also arrive at similar phenotypes despite taking very different paths through the fitness landscape. These different paths will likely result in underlying differences in genetic architecture that may influence which behaviors are easy for a lineage to evolve next. As such, we suspect that diversity of building blocks will promote greater evolutionary potential than other kinds of population diversity.

5.1.3 Limited resources and Eco-EA

The original formulation of limited resources that led to ecologically-mediated hints arose in work by Cooper and Ofria where they demonstrated that limited resources are sufficient to evolve stable branching of different ecotypes [3]. The resulting ecological community was incredibly simple, with competition being the only form of interaction between organisms. However, we hypothesized that these simple ecologies (with meaningful differences between niches) were better positioned to solve complex problems than populations in which diversity was promoted by other means.

Goings and Ofria initially tested a more applied form of limited resources in Eco-EA, using a toy bitstring matching problem, with the goal of maintaining a population of diverse solutions [7]. In this experiment, resources (either limited or unlimited) were associated with different bitstrings that could be matched to varying extents by members of the population. When resources were unlimited, the population converged on matching a single one of these strings. Limiting the resources, however, produced consistent subpopulations specialized on each string. The negative frequency dependent selection imposed by the limited resources caused the different bitstrings to stably coexist within the population. These results held when the task was to match a more general pattern of bits, rather than an exact string.

Following this initial conceptual test, Goings et al. applied Eco-EA to a complex real-world problem: generating models for the behavior of sensor nodes in a flood warning network [6]. Not only did Eco-EA successfully evolve a diversity of models that satisfied the constraints of the problem, but the models evolved by Eco-EA were better than a comparison algorithm at continuing to diversify when transferred to an environment without limited resources. These results demonstrate the strength of Eco-EA at evolving a diversity of solutions to complex problems.

Eco-EA is just one example of the benefits that making more thoughtful and intentional use of ecology can provide. Here, we define ecological dynamics as interactions between members of the population that affect fitness. We propose that such dynamics provide a variety of benefits, some of which have begun to be put in to practice and some of which have not. The most obvious of these effects is the benefit to diversity alluded to above.

5.1.4 Complexifying environments

Thus far, the ecological communities that we have discussed have only contained competitive interactions between organisms. In nature, however, there are many other kinds of interaction, such as mutualism, parasitism, and predation. The network of interactions within a community grows increasingly complex over evolutionary time, as the evolution of new species creates new niches for other species to evolve into. This gradual complexification has two implications that may be important for evolutionary computation: 1) The gradual increase in complexity of the biotic environment facilitates the continual production of more complex niches and thus the evolution of more complex traits, and 2) Ecological communities as a whole can often perform functions that no single species could perform alone.

The niches inhabited by various members of an ecological community usually bear commonalities. While they are not identical, the ability to occupy one niche is often a building block for the ability to occupy a different one (a clear example of this effect in nature is metabolic pathways for metabolizing various resources). When new niches appear, they are closely enough related to existing niches that it's plausible that existing populations will evolve to inhabit them. Without the gradual feedback loop of increasing complexity that eco-evolutionary feedbacks enable, evolving the ability to exploit newly established niches would be incredibly improbable.

Community-level functions represent a different mechanism for solving complex problems. For example, a forest, collectively, is able to store solar energy in sugar molecules, fix nitrogen, take in various nutrients from the soil, and use these nutrients and energy to power mobile, decision-making agents. Of course, these functions are each performed by different species in the community. While it may be possible to evolve a single species that carries out all of these tasks, it is worth considering the possibility that it is easier to evolve a community that carries them out collectively. The fact that such communities seem to be so common in nature is certainly suggestive of this possibility. Such an approach would have a variety of potential benefits. For example, it would promote more functional modularity, a trait that is believed to be critical for evolutionary potential. Moreover, the closely interdependent species found in a complex ecological community are a likely precursor to egalitarian major transitions, which are believed by many to be a critical step towards evolving complex species. Lastly, as ecological communities can effectively be thought of as a set of subroutines running in parallel, they lend themselves easily to evolving parallel programs, somewhat akin to Holland's bucket brigade algorithm [13] and more recent advances in learning classifier systems.

Incorporating such complex interactions into evolutionary algorithms will be challenging, and is far beyond the scope of the current paper. However, these factors are yet another reason that we believe that ecological dynamics have the potential to be a powerful force in evolutionary computation in the long run.

5.2 Methods

We expect that ecologically-mediated hints provided via Eco-EA and lexicase selection will display similar set of strengths. Both maintain a diverse population with respect to the hints that are provided. However, as discussed above, we expect the use of limited resources to be robust to hints that turn out to be counter-productive, while lexicase selection will have difficulty escaping uncontested bad advice. To assess the accuracy of these hypotheses, we evaluate them in the context of a proof-of-concept problem where we can easily manipulate the quantity and quality of the hints.

5.2.1 10-dimensional box problem

The search space for our proof-of-concept problem is a 10-dimensional box. All sides of the box have a length of 1. Candidate solutions in the population are sequences of 10 floating point numbers between 0 and 1, representing a point within the box. The goal is to find the origin (i.e. the sequence [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]). This problem could be trivially solved by using the inverse Euclidean distance between a point within the box and the origin as the fitness function. To simulate a more challenging problem, we use the following fitness function:

$$fitness = \begin{cases} .01 & \sqrt{\sum_{i=1}^{10} x_i^2} > .1 \\ \frac{1}{\sqrt{\sum_{i=1}^{10} x_i^2}} & \sqrt{\sum_{i=1}^{10} x_i^2} \leq .1 \end{cases} \tag{5.1}$$

In this function, inverse Euclidean distance from the origin is the fitness only when Euclidean distance from the origin is less than 0.1 (which, given the high dimensionality of the space, represents 2.5×10^{-13} of the possible positions). For all points in the box that are farther away from the origin, fitness is 0.01. We used 0.01 rather than 0 as the base fitness to ensure the Eco-EA's fitness multipliers would have an effect. The cut-off of 0.1 was chosen such that it is unlikely for evolution to solve the problem without hints. Using this fitness function creates a quintessential ”needle in a haystack” problem, where the fitness landscape is flat except for one incredibly tall and thin peak. Such problems are generally considered to be among the most challenging for evolutionary computation to solve, as they do not allow for incremental improvement.

To make this problem more possible for evolution to solve, we can provide hints about the optimal value for each dimension. Since the goal is to minimize all dimensions, a good (i.e. informative) hint would be to minimize an individual dimension. Conversely, maximizing an individual dimension would be a bad (i.e. misleading) hint, leading a population away from the goal. We provided various combinations of good and bad hints to Eco-EA, lexicase selection, and to standard tournament selection. As an additional control, we ran an equivalent number of trials using unaltered tournament selection.

5.2.2 Eco-EA implementation

The crux of the idea behind Eco-EA is that it must provide hints about how to solve a problem and incentivize following them in a manner that leads to negative frequency-dependence. There are a variety of ways to implement this concept, and there are likely various trade-offs amongst them that are worthy of a systematic study. For the purposes of this chapter, we have used an implementation as similar as possible to the original Eco-EA implementation [7].

Each hint is associated with a resource. That resource flows into the environment at some rate, I, and flows out at some rate, O. For these experiments, the inflow rate, I, was 100, and the outflow rate, O, was .01. Thus, 100 units of each resource entered the environment over the course of each generation, and 1% of the total quantity of each resource exited the environment at the end of each generation. Resources entered the environment at a constant rate as fitnesses were evaluated to minimize stochastic effects from the order in which solutions were evaluated.

There are a few parameters that are necessary for determining how resources are consumed and how they impact fitness. First, there is C_f, the consumption fraction. Just as no organism in an ecosystem in nature is capable of consuming all of a resource in that environment, no solution in Eco-EA is. C_f specifies what fraction of the total quantity of resource any individual solution consumes. Next, there is m, the maximum amount of resource an individual is capable of consuming at any one time. For all experiments, we use $C_f = .0025$ and $m = 5$ to maintain consistency with previous Eco-EA research [5]. Additionally, there is the c, the cost of attempting to use a hint. In order to create negative frequency dependence, there must be a cost to attempting to use a hint when too many other members of the population are also attempting to use it. In most scenarios, there is an implicit cost to attempting to use a hint, stemming from the trade-offs inherent in choosing to take one action over another. In a problem as simple as the 10-dimensional box problem, however, there is no such implicit cost. Thus, without an explicit cost, there will not be negative frequency-dependence. Since negative-frequency dependence is a core element of Eco-EA, we impose a cost of $c = 1$. Lastly, if there is a cost to attempting to use a hint, then we also must define the range within which a solution is considered to be attempting to use that hint. We call this range the niche width, n, which is set to 0.2 for all of these experiments. Since the maximum score for any hint function is 1, this means that solutions must score a minimum of 0.8 on a hint function in order to potentially get a reward or pay a cost from it.

In order to calculate the fitness impact of a hint, we need two more pieces of information. The first is R, the current amount of the relevant resource present. The second is s, the score on the hint function, which is squared to incentivize even small increases in performance on the hint function. The amount of resource successfully used, A, is calculated with the equation:

$$A = \begin{cases} 0 & s < 1 - n \\ min((s^2 * C_f * R) - c, m) & s \geq 1 - n \end{cases} \tag{5.2}$$

A is subtracted from the current amount of resource in the environment, and is used to update the base fitness of the current organism with the equation:

$$fitness = fitness * 2^A \tag{5.3}$$

Thus, successfully using more than c resource will multiply the solution's fitness by a number greater than one, whereas using less than c resource will multiply the solution's fitness by a number less than one.

These calculations are performed for all hints for all members of the population to determine resource-adjusted fitness for all candidate solutions. Tournament selection with a tournament size of two is then performed on the population based on these fitness values. To ensure that the impact of hints doesn't wash out small fitness gains on the main fitness function, every generation created with Eco-EA also contains a copy of the individual from the previous generation with the highest base fitness.

5.2.3 Lexicase selection implementation

Traditionally, lexicase selection is given a large number of test cases. For each iteration of selection, these test cases are placed in a random order. Each member in the population is evaluated on each one in sequence. For each test case, only those solutions that performed best are kept in contention to be selected. When only a single solution is left, it is placed into the next generation. Ties are broken randomly.

We argue that test cases can be thought of as a subset of the broader category of hints about how to solve a problem, and that lexicase selection should generalize to any kind of hint. So, in place of test cases, we use the hints on solving the 10-dimensional box problem described above. For every selection event, we randomly order these hint fitness functions (along with the overall fitness function) and filter the population based on this ordering.

5.2.4 Tournament selection implementation

In an iteration of tournament selection, a pre-determined number (two, in these experiments) of individuals are chosen at random from the population. The fittest of them is selected to reproduce. In this paper, we use tournament selection as our control, as it is effectively a less-informed version of the implementation of Eco-EA that we use here.

5.2.5 Configuration details

For all experiments, we evolved a population of 5000 vectors containing 10 floating-point numbers between 0.0 and 1.0 (inclusive) for 50,000 generations. The next population for each generation was chosen using one of the three selection schemes being compared: lexicase selection, Eco-EA, or tournament selection. To avoid giving Eco-EA an unfair advantage due to its use of elitism (in a problem domain where that could only be beneficial), we always preserved a copy of the individual in the population with the highest fitness. For selection schemes requiring a tournament size (tournament selection and Eco-EA), we used a tournament size of two.

We placed the individuals selected by each iteration of the selection scheme into the next generation. Each site in each genome was mutated by adding a value randomly selected from a Gaussian distribution centered at 0 with a standard deviation of .05. Subsequently, we recombined each vector with a random other vector, using one-point crossover.

5.2.6 Statistical methods

To determine the effects of good advice, bad advice, and selection scheme on the probability of solving the problem, we performed a logistic regression. The predictor variables were the number of good hints, the number of bad hints, a boolean indicating whether lexicase selection was used, and a boolean indicating whether Eco-EA was used. Tournament selection, being the control, was the base case to which all other conditions were compared. The regression coefficients for the interactions between good or bad hints and the selection type were used as the test statistic for all statements about the effect of a hint type on a selection scheme.

All statistics were computed using the R statistical computing language, version 3.4.3 [21]. Since some combinations of variables were able to perfectly separate successes from failures, which can bias the results of logistic regression, we used Firth's bias reduction technique, as implemented in the R package brglm [16]. All plots were made using the ggplot2 package [27].

5.2.7 Code availability

The full source code for the experiments and analysis in this paper is freely available at `https://github.com/emilydolson/eco-ea-box`. This code makes heavy use of the evolutionary computation modules in the Empirical library, which is available at `https://github.com/devosoft/Empirical`. All code for this paper is open source and freely available.

5.3 Results and Discussion

Both Eco-EA and lexicase selection make effective use of good advice (see Figure 5.1). Eco-EA solved the problem consistently when given at least four good hints. Lexicase selection did even better, solving the problem most of the time when given as few as two good hints. Good hints enable both lexicase selection and Eco-EA to significantly outperform tournament selection (logistic regression, $\beta = 1.3334$ for lexicase, $\beta = 1.8363$ for Eco-EA, $p < .0001$ for both).

These data strongly support our hypothesis that Eco-EA is essentially unaffected by receiving bad hints; the regression coefficient for the interaction between Eco-EA and bad hints is not significantly different from 0 (logistic regression, $\beta = 0.0633$, $p = .66$). Lexicase selection, on the other hand, is harmed dramatically by bad hints (logistic regression, $\beta = -10.6841$, $p < .0001$).

Overall, our results illustrate the power of ecological approaches as a vehicle for providing hints to evolutionary algorithms. We have provided a proof-of-concept that this technique can make it possible for evolution to solve problems that would otherwise have been out of reach. Moreover, we have clarified the instances in which two specific ecological approaches, Eco-EA and lexicase selection, are most appropriate; lexicase is best when all of the hints are accurate, whereas Eco-EA is robust in scenarios where there may be some misleading hints.

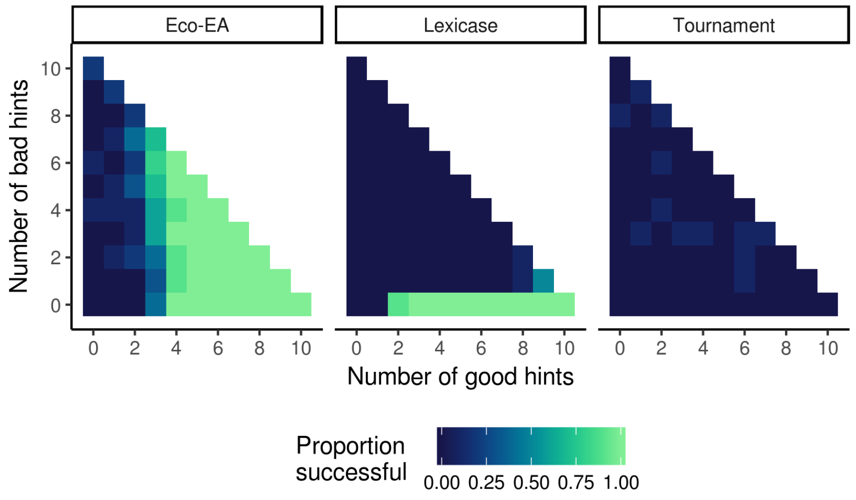

Fig. 5.1: Impact of good and bad advice on Eco-EA and Lexicase. Heat maps for each algorithm show the success rate of that algorithm in the presence of varying quantities of good and bad advice. Note that tournament selection is not actually capable of receiving hints; it is presented here as a control. Each cell in the heat maps represents the proportion of runs (out of 10) that successfully found the optimal solution to the 10-dimensional box problem.

5.4 Conclusions and future work

Thus far, we have tested ecologically-mediated hints on only simple model problems as a proof-of-concept, but we expect this approach to excel on complex problems where fitness functions do not provide a clear path from random starting conditions to meaningful solutions. Artificial Intelligence is a perfect example of this type of problem. Problem-solving strategies may require many different components to coordinate to formulate plans. Board-game-playing agents are a particularly accessible problem that has these properties: they are complex, while still being experimentally tractable, and are often well-studied. They involve clear measures of success, while allowing for multiple co-existing strategies. Most importantly, they are intuitive to humans who can provide suggestions for limited resources to produce building blocks for complex strategies.

The next step will be to more thoroughly explore the parameter space in which providing hints via Eco-EA and lexicase selection is effective. In the process, we hope to gain insight that will allow us to develop an approach that includes the best properties of both. Thus far, we have explored the impact of good and bad hints that are orthogonal to each other. However, most hints in real world problems will not be independent, and we expect that this connection may influence the way Eco-EA and lexicase selection respond to them. Similarly, we have not explored the impact of completely neutral hints, which may be harmful if they are present in excessive quantities. That said, a resource is helpful only when it produces a building block that turns out to be useful for a more complex goal, after which it is no longer needed. As such, many resources would be helpful if they were around just long enough for the associated niche to be filled as a stepping-stone to more complex niches.

If good advice is helpful and other advice is harmless, it should be possible to bootstrap the solving of a problem by generating random hints. Each hint, once used, will remain for a limited amount of time (providing an opportunity to be used as a building-block) before it is removed and replaced by a new hint associated with a new randomly-determined behavior. These random behaviors will usually be harmful and thus ignored, but any that turn out to be helpful building blocks should be incorporated into the successful players.

Finally, we will use human problem-solving patterns to create niches for evolving crowd-sourced AIs. While we believe transient resources will have some utility, most of those building blocks are likely to be useless, merely using up computational time. A benefit of working with board games, however, is that we can collect a wealth of information about how human players make decisions across a variety of situations. Rather than reward building blocks that we identify from those logs, we can instead create limited resources that reward AI players for consistently predicting the next move made by a human player. In other words, we do not need to understand why a human player made a decision in order to reward an AI for following the same type of strategy. If that player plays well, the building blocks produced to mimic them should be generally useful, even for some other strategy types. Ultimately, the biggest rewards will still come from winning games, so mimicking poor players should have a minimal negative impact (as is usually the case with Eco-EA).

References

1. Bongard, J. C. and Hornby, G. S. (2010). Guarding Against Premature Convergence While Accelerating Evolutionary Search. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, GECCO '10, pages 111–118, New York, NY, USA. ACM.
2. Chesson, P. (2000). Mechanisms of Maintenance of Species Diversity. *Annual Review of Ecology and Systematics* 31:343–366.
3. Cooper, T. F. and Ofria, C. (2002). Evolution of stable ecosystems in populations of digital organisms. In *Artificial Life VIII: Proceedings of the Eighth International Conference on Artificial life*, pages 227–232.
4. De Jong, K. A. (1975). Analysis of the behavior of a class of genetic adaptive systems.
5. Goings, S. (2010). Natural niching: Applying ecological principles to evolutionary computation. *Dissertation*, Michigan State University.
6. Goings, S., Goldsby, H. J., Cheng, B. H., and Ofria, C. (2012). An ecology-based evolutionary algorithm to evolve solutions to complex problems. *Artificial Life*, 13:171–177.
7. Goings, S. and Ofria, C. (2009). Ecological approaches to diversity maintenance in evolutionary algorithms. In *IEEE Symposium on Artificial Life, 2009. ALife '09*, pages 124–130.
8. Goldberg, D. E. and Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimization. In *Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 41–49. Hillsdale, NJ: Lawrence Erlbaum.
9. Harper, R. (2012). Spatial Co-evolution: Quicker, Fitter and Less Bloated. In *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*, GECCO '12, pages 759–766, New York, NY, USA. ACM.
10. Helmuth, T. and Spector, L. (2015). General Program Synthesis Benchmark Suite. *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO '15, pages 1039–1046, New York, NY, USA, ACM.
11. Helmuth, T., Spector, L., and Matheson, J. (2015). Solving Uncompromising Problems With Lexicase Selection. *IEEE Transactions on Evolutionary Computation*, 19(5):630–643
12. Hendry, A. P. (2016). *Eco-evolutionary Dynamics*. Princeton University Press.
13. Holland, J. H. (1985). Properties of the bucket brigade. In *Proceedings of an International Conference on Genetic Algorithms*, pages 1–7, Hillsdale, NJ, USA. Lawrence Erlbaum Assoc.
14. Hornby, G. S. (2006). ALPS: The Age-layered Population Structure for Reducing the Problem of Premature Convergence. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, GECCO '06, pages 815–822, New York, NY, USA. ACM.
15. Hu, J., Goodman, E., Seo, K., Fan, Z., and Rosenberg, R. (2005). The Hierarchical Fair Competition (HFC) Framework for Sustainable Evolutionary Algorithms. *Evolutionary Computation*, 13(2):241–277.
16. Kosmidis, I. (2017). brglm: Bias Reduction in Binary-Response Generalized Linear Models, version 0.6.1
`http://www.ucl.ac.uk/~ucakiko/software.html`
17. Lenski, R. E., Ofria, C., Pennock, R. T., and Adami, C. (2003). The evolutionary origin of complex features. *Nature*, 423(6936):139–144.
18. Mahfoud, S. W. (1992). Crowding and preselection revisited. *Urbana*, 51:61801.
19. Mouret, J.-B. and Doncieux, S. (2009). Using Behavioral Exploration Objectives to Solve Deceptive Problems in Neuro-evolution. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, GECCO '09, pages 627–634, New York, NY, USA. ACM.

20. Mouret, J.-B. and Clune, J. (2015). Illuminating search spaces by mapping elites. In *arxiv:1504.04909*
21. R Core Team. (2017). R: A Language and Environment for Statistical Computing. `https://www.R-project.org`
22. Schoener, T. W. (2011). The Newest Synthesis: Understanding the Interplay of Evolutionary and Ecological Dynamics. *Science*, 331(6016):426–429.
23. Spector, L. (2012). Assessment of problem modality by differential performance of lexicase selection in genetic programming: a preliminary report. In *Proceedings of the 14th annual conference companion on Genetic and evolutionary computation*, pages 401–408. ACM.
24. Spector, L. and Robinson, A. (2002). Genetic programming and autoconstructive evolution with the push programming language. *Genetic Programming and Evolvable Machines*, 3(1):7–40.
25. Stanley, K. and Miikkulainen, R. (2004). Competitive coevolution through evolutionary complexification. *J. Artif. Intell. Res. (JAIR)*, 21:63–100.
26. Taylor, T., Bedau, M., Channon, A., Ackley, D., Banzhaf, W., Beslon, G., Dolson, E., Froese, T., Hickinbotham, S., Ikegami, T., McMullin, B., Packard, N., Rasmussen, S., Virgo, N., Agmon, E., Clark, E., McGregor, S., Ofria, C., Ropella, G., Spector, L., Stanley, K. O., Stanton, A., Timperley, C., Vostinar, A., and Wiser, M. (2016). Open-Ended Evolution: Perspectives from the OEE Workshop in York. *Artificial Life*, 22(3):408–423.
27. Wickham, H. (2009). *ggplot2: Elegant Graphics for Data Analysis.* Springer-Verlag, New York.