This is to certify that the
dissertation entitled

NATURAL NICHING: APPLYING ECOLOGICAL PRINCIPLES TO EVOLUTIONARY COMPUTATION

presented by

SHERRI GOINGS

has been accepted towards fulfillment
of the requirements for the

Ph.D. degree in Computer Science
Ecology, Evolutionary Biology, and Behavior

Major Professor's Signature

5-12-2010

Date

MSU is an Affirmative Action/Equal Opportunity Employer

NATURAL NICHING: APPLYING ECOLOGICAL PRINCIPLES TO EVOLUTIONARY COMPUTATION

By

Sherri Goings

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Computer Science
Ecology, Evolutionary Biology, and Behavior

2010

ABSTRACT

NATURAL NICHING: APPLYING ECOLOGICAL PRINCIPLES TO EVOLUTIONARY COMPUTATION

By

Sherri Goings

Evolutionary algorithms have shown great promise in evolving novel solutions to real-world problems, but the complexity of those solutions is still limited, unlike the apparently open-ended evolution that occurs in the natural world. The power of traditional evolutionary algorithms is constrained by rapid convergence to a single solution on a sub-optimal local peak, leading to stagnation. In part, nature surmounts these complexity barriers with ecological dynamics that generate a diverse array of raw materials for evolution to build upon.

In this dissertation I focus on one ecological force that increases diversity: frequency-dependent selection that arises from competition among individuals for finite resources. I explore the benefits of incorporating this force into an algorithmic framework, focusing on how this mechanism can increase diversity to provide many evolutionary paths to a problem solution. The use of niching to simultaneously approach a single solution from many directions has not been extensively studied in previous literature. I study competition for limited resources in a digital evolution system and examine the importance of specific resource parameters on population dynamics. I show that my techniques are robust at increasing diversity over a broad range of settings for each parameter, and that the parameter settings are governed by a relatively simple set of equations.

Finally I introduce Eco-EA, a general form of an evolutionary algorithm that associates a limited resource with each trait to be evolved. I apply Eco-EA to several problems, including a real-world software engineering problem, and I show that the Eco-EA yields several advantages over traditional evolutionary algorithm approaches, including: (1) significantly

more rapid evolution of targeted complex functions; (2) discovery and maintenance of a diverse set of partial solutions that together solve a problem; (3) maintenance of a selection of high-quality final solutions for the researcher to choose from, often with slightly different properties; and (4) discovery of solutions that are more evolvable when placed in new environments.

This dissertation is dedicated to my mother, who has provided endless support and encouragement throughout my dissertation research and in memory of my father, who gave me the confidence to be where I am today.

ACKNOWLEDGEMENTS

I would like to thank the people who made this dissertation possible. I am deeply grateful to my family, who have always supported and encouraged me in whatever endeavors I undertake, and have stood by me through all of the ups and downs of grad school. I am also forever indebted to my advisor, Charles Ofria, not only for his time, effort, and insights on my work itself, but for his friendship, understanding, and unbreakable optimism that has sustained my own confidence and drive to complete this journey. I am equally grateful to Rich Enbody, who has been a mentor and friend through my entire college career, and who was never too busy to answer a question, give a word of encouragement, or just listen when I needed it.

I would like to thank the entire Devolab; I am incredibly lucky to have worked with such a great group of people at Michigan State, and I cant begin to enumerate all of the ways these friends and colleagues have made my achievements possible. A special thanks to Dule Misevic, Elizabeth Ostrowski, and Jason Stredwick for their guidance and mentoring as the older and wiser members of the lab when I began, and to Jeff Clune for his support and friendship as we experienced the first few years of the PhD process together, as well as his stellar ideas in our early collaborations. Also thank you to all of the other people who have shared projects and collaborated on publications; Rob Pennock, Rich Lenski, Bill Punch, Erik Goodman, Betty Cheng, Heather Goldsby, and Bess Walker. I could not have begun to complete this thesis without the insights and suggestions of my entire dissertation committee, Charles Ofria, Erik Goodman, Rich Lenski, Phil McKinley, and Rich Enbody. Thank you for your precious time and effort.

Finally the deepest thanks to all of my friends, who may not have directly helped me with my dissertation, but whose support has made everything else in my life possible. Though I cannot list you all here, I would like to give a special thanks to Janine Konkel, Tamara Convertino, and all of the beautiful women of my Strut and Michigan State Ultimate Frisbee Teams over the years. The experience of the support and bonding that can

only come from the kind of hard work and competition we experienced together has forever changed my life. Also thanks to Rob Post, Molly Park, and Eric Witham for your constant friendship through some of the most difficult times of my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Images in this dissertation are presented in color.

Chapter 1

Introduction

An *Evolutionary Algorithm (EA)* is a search algorithm that abstracts mechanisms of organic evolution (such as reproduction, mutation, and selection) and uses them to seek optimal solutions to problems (Back, 1996). 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. To design EAs that can solve more complex problems, we must examine how complex traits arise in nature and how EAs fall short in duplicating these dynamics. The complexity of solutions produced by traditional EAs is partly limited by rapid convergence to a single solution on a local optimum, resulting in stagnation. One of the ways nature surmounts this complexity barrier is with natural ecological dynamics that generate a diverse array of raw materials for evolution to build upon.

Diversity in an evolutionary algorithm can provide other advantages beyond forestalling stagnation, including: (1) maintenance of a selection of good solutions for the researcher to choose from, often with slightly different properties; (2) the availability of different partial solutions to be used as building blocks toward the full solution, without the researcher needing to know the ideal path; (3) resilient solutions that can withstand environmental changes; and (4) significantly faster evolution of targeted complex functions. Robust eco-

logical communities exhibit all of these traits.

Designers of evolutionary algorithms recognize the importance of maintaining population diversity and make use of a variety of diversity preserving techniques based loosely on natural mechanisms (Mahfoud, 1995). However, the selective pressures that have led to the high levels of diversity found in the natural world are not yet clearly understood themselves (Tilman, 2000), nor are the nuances of how diversity facilitates the evolution of complex behaviors. In order to incorporate the more subtle dynamics of natural evolution into evolutionary algorithms we must first understand the workings of these mechanisms in nature.

I focus on one ecological force that can promote diversity: frequency-dependent selection that arises from competition among individuals for finite resources (Tilman,1982). A limitation on available resource can lead to the formation of an ecology of diverse species since negative frequency-dependent selection will drive competition to favor rare types (Chow *et al.*, 2004). My goal for this thesis is to explore the dynamics of competition for limited resources in natural ecologies and to use our improved understanding to create an ecology-based evolutionary algorithm capable of solving complex problems.

The remainder of this dissertation is divided into six chapters organized as follows:

- In Chapter 2, I provide further motivation for the importance of diversity and I survey current methods for maintaining diversity in evolutionary algorithms. I also provide background on how competition for limited resources can maintain ecosystems of diverse species in the natural world.

- In Chapter 3, I focus on exploring the dynamics of competition for limited resources using a digital evolution system. I analyze the effects of the parameters that govern the limited resources and describe how to set these appropriately.

- In Chapter 4, I explore the benefits of using limited resources in evolving complex traits. I posit three theories for how environments incorporating this mechanism lead to the evolution of more complex behaviors than those with non-limiting resources.

- In Chapter 5, I further explore the dynamics of competition for limited resources when applied to simple applications where the goal is to evolve a diverse set of solutions that "cover" a solution space. I examine the factors that lead to many unique specialists vs. a few generalists, and compare this natural mechanism of maintaining diversity to other methods currently used in evolutionary algorithms.

- In Chapter 6, I introduce a general concept of an ecology based evolutionary algorithm (called Eco-EA), and apply it to a real-world application of evolving UML models for nodes in a remote sensor network. I show three clear advantages of the Eco-EA over a traditional EA as applied to this problem, including:

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

- Finally, in Chapter 7, I provide a summary, detail future directions that I plan to take this research, and provide my initial speculations and results in these directions.

Chapter 2

Background and Related Work

Evolutionary Algorithms are inspired by the ability of natural evolution to produce the complex life found in the biological world. One of the most striking features of biological life is its diversity, however a standard EA does not yield a diverse population of individuals but rather tends to quickly converge to a single phenotype that may not represent the global optimal solution to a problem. The diversity found in nature keeps biological populations from stagnating and allows the open-ended evolution we see in the real world. In order to improve the ability of evolutionary algorithms to find solutions to complex problems, we must incorporate more of the natural dynamics that maintain diversity in biological populations. In this chapter I will discuss the potential benefits of diverse ecosystems in nature, the theory on how diversity is achieved and maintained in the natural world, and previous mechanisms of increasing diversity in evolutionary algorithms. It is important to note that in this thesis the type of diversity we refer to is species richness, where species are defined in an EA as functionally distinct phenotypes.

2.1 Diversity in Nature

The impacts of diversity on natural ecosystem processes are not fully understood (Tilman, 2000), however two proposed benefits of a diverse ecosystem have been supported by the-

ory and experimentation. The first benefit is greater productivity of an overall ecosystem when it is composed of many species rather than only a few (Tilman *et al.*, 1996; Tilman *et al.*, 1997; Hector *et al.*, 1999). The second benefit is that a diverse ecosystem is more reliable or stable than a homogenous population (McCann, 2000). We propose a potential third benefit in the long-term time frame of evolution, which is that a diverse population yields more potential paths for evolution to follow and thus increases the probability that a new complex trait will evolve.

Several mechanisms leading to the high level of diversity seen in natural ecosystems have been proposed. In general, coexistence of many different species will occur when there are "evolutionarily persistent interspecific trade-offs" (Tilman, 2000) between the abilities of different species to respond to different environmental factors. Several such interspecific trade-offs may occur, including differences in species abilities to compete locally and ability to disperse widely, species success in average conditions and exploitation of times of resource abundance, or species ability to compete for alternative resources (Hastings, 1980; Armstrong and McGehee, 1980; Huisman and Weissing, 1999; Tilman, 1982). In this thesis I will focus on the last of these potential trade-off, that is the abilities of different species to utilize alternative resources in an environment containing multiple distinct, finite resources.

A large body of literature, both theoretical and experimental, explores how competition for limited resources can maintain diversity in the natural world, but I will cover here only a few key points that relate to incorporating this method into evolutionary algorithms (A more in-depth treatment can be found in Tilman, 1982). Competition for multiple resources will yield diverse species only under certain conditions. First different species must have varying abilities to use different resources. The simplest case to consider is when each species specializes on a different resource, creating a "niche" associated with each resource, although multiple species may still be maintained in certain cases of overlapping niches as well. Second the amount of available resource must be sufficiently limited to create neg-

ative frequency-dependent selection; that is, the more individuals that are using a single resource, the less benefit they should each receive from it. As long as there is enough resource available that the benefit to individuals using it is higher than the cost, a selective pressure exists for more individuals to use that resource. Both theoretical and experimental evidence has shown that species richness is highest at intermediate levels of resource availability (Chow *et al.*, 2004). Finally given no other population interactions, a well-mixed environment can indefinitely maintain only as many species as there are distinct resources available. These conditions must all be met if we hope to use competition for multiple limited resources to increase diversity in an evolutionary algorithm.

2.2 Diversity in Evolutionary Algorithms

Designers of evolutionary algorithms recognize the importance of maintaining population diversity and make use of a variety of diversity-preserving techniques. Increasing the diversity of an EA population is generally recognized as having two potential purposes: to allow the exploration of more of the search space in order to generate a better single solution; and to form and maintain multiple distinct solutions. Many diversity-preserving methods focus only on the role diversity plays in preventing a population from stagnating by prematurely converging to a single solution before the best solution had been found. These methods slow convergence but are not capable of explicitly maintaining multiple species in a population for long periods of time. Other techniques exploit both potential advantages of diversity; these methods (called "*niching*") are capable of not only avoiding convergence but also of finding and maintaining multiple diverse solutions (Mahfoud, 1995). Research into niching methods in evolutionary algorithms has been largely focused on the ability to find and maintain multiple solutions, such as in multi-objective problems or multi-modal functions, but in this thesis I will focus on how maintainining multiple sub-solutions on the way to a single final solution can improve the ability of an algorithm to find solutions to

complex problems.

2.2.1 Preventing Premature Convergence

A technique called *crowding* was one of the first methods suggested to prevent premature convergence of an evolutionary algorithm while searching for a single solution (DeJong, 1975). This method slows the loss of population diversity by modifying only the replacement step of an evolutionary algorithm to force new individuals to replace similar individuals in the population. Each time a new individual is ready to be placed in the population, a small subset of the population is sampled, and the individual determined to be most similar to the new one is replaced. Similarity between individuals is determined by either a genotypic or phenotypic distance measure. The crowding technique has been shown to significantly increase the speed of search and quality of solution found by an EA before convergence. Simple crowding is, however, subject to a high rate of replacement errors that occur when an individual on one fitness peak is replaced by an individual from another peak. Populations eventually converge to one fitness peak and the benefits of using crowding in replacement are lost.

Deterministic crowding was designed by Mahfoud (1992) to address the issue of replacement error. In this method the entire population is split into pairs that undergo crossover to produce two offspring. Each offspring competes with its most genetically similar parent and the winners of these competitions are put back into the population. Deterministic crowding minimizes replacement errors, but cannot eliminate them entirely and populations still slowly converge. Both crowding methods require the use of a distance measure between individuals, but for many problems it can be difficult to design an effective phenotypic distance measure for many problems and genotypic distance may not be well correlated to fitness peaks in the landscape.

A different approach that does not require a distance measure is to lower gene flow by subdividing a population into several small populations such as in the *island* model.

This method allows competition only between individuals in the same subpopulation. A small amount of migration occurs between subpopulations to spread good solutions among the entire population. Even this low level of migration between subpopulations allows the population to eventually converge to a single solution. The *diffusion* model implements a complex population structure to achieve the same goal of lowering gene flow without explicitly creating distinct subpopulations (Back *et al.*, 1997).

An extension of the island model is the *Hierarchical Fair Competition (HFC)* model (Hu and Goodman, 2002). In this method individuals are grouped into subpopulations based on their fitness. Migration between subpopulations moves in only one direction; when an individual exceeds the fitness-based admission threshold of the next higher subpopulation it is moved up, but individuals are never moved into a lower fitness subpopulation. The HFC model maintains a constant flow of individuals representing potentially new areas of the search space moving from the lower into the higher subpopulations, however each subpopulation may still converge, leading to stagnation of the overall algorithm.

2.2.2 Niching

The previous methods work by slowing the loss of existing diversity, but do not inherently direct exploration to new areas of a search space. Several techniques use direct measures of diversity to increase exploration in an evolutionary algorithm and prevent populations from ever converging. One of the first technique of this type was the *Diversity-Control-Oriented* genetic algorithm (Shimodaira, 1999). In this method the survival probability of each individual is partly determined by the genetic distance between it and the current best individual in the population. A higher distance means a higher survival probability, encouraging exploration of areas genetically distant from the current best solution. Two other approaches divide a population into subpopulations focused on either exploration or exploitation. The *Shifting-Balance* genetic algorithm (Oppacher & Wineberg, 1999) divides the population into a large core population and several small colony subpopulations. The

core population exploits the most promising areas of the search space by receiving through migration the best individuals from the colony populations. The colony populations are used for exploration; they are forced to search in different areas of the fitness landscape than the core population. The *forking* genetic algorithm (Tsutsui *et al.*, 1997) utilizes a similar population structure, but assigns the populations opposite roles; the parent population is responsible for exploration, continuously searching for new peaks, while a number of child populations try to exploit previously detected promising areas. The forking genetic algorithm is unique among those discussed so far in that it finds and indefinitely maintains solutions representing multiple peaks in the fitness landscape and so is considered a true niching algorithm. A final method in this class is the *Diversity-guided* EA (Ursem, 2002). This method uses just one population but alternates between exploration and exploitation phases based on the current level of diversity in the population; a high level of diversity triggers the exploitation phase and a low level triggers the exploration phase.

One of the most popular niching methods used in evolutionary algorithms is *fitness sharing* (Goldberg & Richardson, 1987). This method maintains separate species covering multiple peaks in the fitness landscape while simultaneously encouraging the discovery of new peaks. A GA using fitness sharing should never converge, and the number of individuals present in any given species will be roughly proportional to the height of the peak those individuals are representing. Fitness sharing uses a distance measure to force an individual to "share" its fitness with others that are within its "niche radius". The individual does not actually share its fitness in the sense of giving it away; its own fitness is simply reduced by an amount based on the distance between it and each other organism within the niche radius. The closer another individual is, the larger the fitness reduction to the one being evaluated. This sharing function encourages exploration in the fitness landscape as individuals that are different from the rest of the population are awarded their full fitness, while those that are similar, clustered around peaks, end up with only a small fraction of their initial fitness. Goldberg was able to achieve impressive results on multimodal functions

(Goldberg, Deb, & Horn, 1992), but fitness sharing has drawbacks as well. Like crowding, fitness sharing requires the use of a distance measure. Also a high level of problem domain knowledge may be required to effectively set the niche radius and determine the sharing function in the context of the fitness landscape for the problem being solved. Finally, this method is largely dependent on a relatively uniform distribution of peaks in the search space (Darwen and Yao, 1996).

Niched genetic algorithms are often improved by the use of some method of limited sexual recombination between population individuals in order to enforce stricter speciation. Individuals representing different fitness peaks generally do not produce a fit offspring when mated. Inter-species recombination therefore is often a waste of search time. Two different methods have been suggested by Goldberg (1989) to limit mating to within-species without losing the ability of the algorithm to perform effective search. The first is to add a tag to each individual in the population that undergoes mutation and crossover but is not part of the fitness function, and then allow individuals to mate only with others that have the same tag. The second is to allow an individual to mate only with another within a certain distance (determined by the same measure used for niching) of itself. Both of these limited mating schemes have shown improvements on niching alone.

Resource sharing is a form of implicit fitness sharing, often found in the context of learning classifier systems, where one example to be classified has only a limited amount of credit to give that must be divided among each rule that classifies it correctly. In this case each example can be viewed as a finite resource. A less obvious instance of implicit fitness sharing is the immune system model analyzed by Forrest *et al.* (1993). Forrest's implicit sharing method preserves diversity in a single population and leads to emergent problem decomposition in a simple immune system environment. Forrest *et al.* abstract and simplify the immune system by representing antibodies and antigens with simple binary strings. They evolve a population of antibodies to "cover" a given set of antigens. To assign fitness they choose a single antigen from the set of antigens, and n antibodies from the evolving

population. The antibody most closely matching the antigen receives fitness proportional to the number of bits it matches, the other antibodies receive no credit. This process is repeated many times each generation. Forrest et al. show that this mechanism preserves diversity through a form of implicit fitness sharing.

2.3 Conclusion

Fitness sharing and implicit resource sharing are true niching techniques that directly aim to maintain multiple diverse species of individuals in a population. These methods work at a high level by directly manipulating credit assignment to create a negative frequency-dependent selection in which competition favors rare types. Both methods have been shown to be effective in certain problem domains, yet still have not yielded the open-ended evolution necessary to solve complex problems that we see occur in the natural world. In this thesis I will use an artificial life system to explore a more explicit resource sharing method inspired directly by natural ecological communities (Cooper & Ofria, 2002), and use my new understanding of this method to create an improved ecological evolutionary algorithm.

Chapter 3

Exploring the Dynamics of Competition for Limited Resources in Artificial Life

Natural systems can evolve intricate ecologies that create and maintain diverse organisms and rapidly evolve complex traits and survival strategies. If we wish to apply similar dynamics in computational systems to evolve solutions to complex problems, we must first understand these dynamics in the natural world. Studying the evolutionary formation of natural ecosystems, however, is difficult due to the time involved, the fragility of the ecosystems, and the amount of data that would need to be observed to get a clear picture of what was happening. Even in relatively simple laboratory microcosms of model bacteria, detailed studies are practically intractable. To overcome these difficulties in part, I use a digital evolution system, Avida, that allows a researcher to track the genetic, phenotypic, and ecological state of the system at all times.

In this chapter I explore competition for limited resources as a natural mechanism for creating and maintaining diverse organisms. I describe the specific implementation of a limited resources environment in Avida, and study the effects on population dynamics of the parameters that are used to control the limited resources

3.1 Avida

For most of the experiments in this thesis I use the digital evolution research platform Avida, described fully by 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 phenotype of the individual, including its own replication. 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 digital organisms in Avida live in an environment that contains a set of resources, each of which can be metabolized by the digital organisms for them to gain more CPU cycles. When an organism performs a user-defined task, it receives a portion of the available corresponding resource, up to a fixed maximum. The amount of resource the organism obtains, along with the value of that resource, determines the change in an organism's *metabolic rate* which is how quickly it gains CPU cycles. The fitness of an organism is determined by its metabolic rate (the rate at which is produces CPU cycles) divided by its gestation requirments (the number of CPU cycles it needs to use to produce an offspring). Avida uses a unit of time called an "update", during which a total of 30 x population size CPU cycles are processed. During a CPU cycle, one genomic instruction is executed within a single organism in the population. These instructions are allocated to each individual in proportion to metabolic rate, and thus an organism that consumes a resource to increase its metabolic rate is expected to make more copies of itself relative to organisms lower metabolic rates, all else equal.

In most Avida studies, as with most evolutionary algorithms, resources are unlimited,

creating a single-niche environment where an organism receives a constant amount of resource for each task completed, regardless of the behaviors of any other organisms in the population. We will call this traditional Avida environment the "unlimited resource" environment.

3.2 Competition for Limited Resources in Avida

A "limited resources" environment in Avida was first studied by Cooper and Ofria (2002). This type of environment limits the availability of resources such that the use of a resource by one organism reduces the amount left to be used by all others. In all of the studies I present in this thesis, these resources are set up as the computational equivalent of a well-stirred chemostat; each resource flows into the environment at a constant rate, and a small percentage of the available resource flows out, limiting the total possible accumulation. Exploration of new areas of the fitness landscape is highly rewarded as an unused resource will accumulate; as such, the first mutant able to make use of the resource will receive a huge fitness boost. However, when many organisms perform functions that consume the same resource, the availability of that resource will decline until further organisms who attempt to draw from it do not receive enough reward to overcome competitors who are targeting a different resource.

3.2.1 Limited Resources Parameters

Environmental resources are governed in Avida by a set of user-defined parameters. Two parameters regulate the globally available amount of a resource in an analog of a bacterial chemostat (*): *inflow* determines the amount of resource that flows into the environment each update, and *outflow* sets the fraction of the current resource that flows out of the environment each update. In the absence of consumption by individuals in the population, a resource will accrue in the environment until

$$R_i * O = I \tag{3.1}$$

where R_i is the current amount of resource i in the environment, O is the outflow fraction and I is the inflow amount, at which point R will remain constant. Two further parameters determine the amount of resource an organism receives when it executes the associated task. A *consumption fraction* (C_f) sets the fraction of R_i (current resource amount) an organism can consume with each performance of the associated task, however this amount is sometimes capped by an overall *maximum* (m). If

$$C_f * R_i > m \tag{3.2}$$

the organism consumes only m units of the resource. This limitation is akin to a biological organism that can acquire more of a resource when it is plentiful in the environment, but only up to its ability to metabolize the resource; it cannot convert an infinite amount of resource into energy. The imposed maximum consumption slows the initial growth phase of an organism that discovers an unused resource by limiting the amount of energy the organism can get from each use of that resource, no matter how much of the resource has accrued in the environment.

3.2.2 Limited Resources Parameters – Stable State

All four parameters (I, O, C_f, and m) interact to determine the dynamics of a limited-resource environment in Avida, however the interactions are not as complicated as they may first appear and can be largely understood in terms of ecological resource competition theory. According to Tilman (1982), a well-mixed ecosystem can maintain only as many distinct species as there are limiting resources, barring any other types of interactions among individuals or between individuals and the environment. For a resource to be limiting in an Avida environment its availability must be restricted to a low enough level

that not all individuals in a population can receive the *maximum* consumption amount. Resources at this low level create the negative frequency-dependent selection that maintains multiple distinct species by favoring rare types. How much of a resource do we expect an individual to receive? Let us first consider a population that has reached a stable state with Nu_i organisms using resource i each update (resource inflow is set in terms of units/update; organism generations, however, depend on several factors including the metabolic rate of the organism). In the stable state the amount of resource that each organism receives (A) is given simply by Equation (3.3) if we ignore the outflow.

$$A_i = \frac{I}{Nu_i} \tag{3.3}$$

The outflow for all experiments in this thesis was set to 1% of current resource per update. This reduction does have a small effect on the exact numbers in our equations but the effect is small enough that we can ignore it and still produce a close approximation.

In all experiments described in this thesis an organism was allowed to perform each task associated with a resource only once in its lifetime, so the value we need to determine if it is possible for all individuals to receive m units of resource is the inflow of a resource over an organism's lifetime. Assuming the population is in a steady state where all individuals have the same fitness, we can estimate the number of updates it takes any organism in the population to reproduce based on the average gestation time across all individuals (an individual with a longer gestation time must have a higher metabolic rate and thus be receiving a larger proportion of CPU cycles per update than an individual with a shorter gestation time or they would not have equal fitness). Given a population-wide replication time in updates (T_r) of average gestation time divided by 30 (as stated earlier 30 * population size instructions are executed per update), the previous equation becomes

$$A_i = \frac{I * T_r}{N_i} \tag{3.4}$$

where N_i represents the number of organisms in the population that use a resource i during their lifetime. Now we can define a resource as limiting if A_i is less than the maximum resource usable, m, given N_i equal to the entire population size N. Or in terms of setting the resource inflow parameter, it should be set such that the following equation is true.

$$I * T_r < m * N \tag{3.5}$$

Equation (3.5) states that for a resource to be limiting, the amount of resource that flows into the population each generation must be less than the maximum amount a population could consume each generation. If Equation (3.5) is not true for a given resource, every individual in the population that uses the resource will receive the maximum energy no matter how many individuals are using it; this scenario is equivalent to the unlimited-resource environment with no frequency-dependent selection.

Note that the consumption fraction is not involved in any equation thus far. The only effect the consumption fraction parameter has once a population reaches a stable state is to determine the constant level of resource in the environment (given a limiting resource) at that state. Once again ignoring outflow the stable level of a limiting resource is given by the following equation, taking A_i from the stable state equation above.

$$R_i = \frac{A_i}{C_f} \tag{3.6}$$

So in the end, of the four resource parameters (I, O, C_f, and m), only the resource inflow and the maximum consumable have a major effect on the dynamics of a population in a steady state. The consumption fraction is only a factor in the amount of resource found in a population; it does not directly affect the amount of resource an individual obtains, and the outflow is low enough to have only a small effect.

Equations (3.4) and (3.5) make clear that the gestation time of individuals is also im-

portant in determining an optimal resource inflow setting. In traditional Avida experiments gestation time is determined by the individuals themselves and may vary both between experiments and even during the course of a single experiment. However several factors allow a reasonable estimate for gestation time to be used in setting the inflow parameter. First, populations are seeded with an individual ancestral organism that can replicate but cannot perform any other functions, and the average population gestation time generally stays within an order of magnitude of this initial individual's gestation time. We will show later that the inflow parameter is relatively robust and can vary by an order of magnitude while still yielding good results. Second, it is possible to restrict the gestation time of Avida organisms to stay within a smaller range, or even to be constant; I use this method in Chapter 5 for simplicity of analysis. Finally, gestation time is an issue specific to a few systems such as Avida, and is implementation dependent; in systems with synchronous generations, such mathematical complications will not be an issue.

3.2.3 Limited Resources Parameters – Non-stable State

The interactions between the parameters are more complex when considering a population not in a stable state, and I will discuss them only briefly here. The initial growth phase that occurs after the discovery of a new resource is short enough as to not have a major effect on the population dynamics, given a reasonably large population size ($N > 500$). As shown in Equation (3.1), the inflow and the outflow parameters determine the maximum amount of an unused resource that can accumulate in the environment. A larger initial pool of resource leads to a longer initial growth phase where individuals receive a large reward for targeting the resource. The consumption fraction (C_f) and maximum cap (m) interact to determine the shape of the initial growth curve (the rise and potential fall of the number of individuals targeting the resource). At the extreme of $Cf = 1.0$, only m will limit how much of the initial resource pool an individual targeting it receives. If m is infinite (no cap on resource usage), the first individual to use a resource will receive the entire resource pool

and replicate very quickly, but the second time it or its offspring attempt to draw from the resource only the amount that has newly flowed into the population will be available. Thus the amount of energy individuals targeting that resource receive is immediately governed by Equation (3.3), and the subpopulation of individuals using that resource will quickly stabilize to hold as many individuals as that resource inflow can support. If m is small and $C_f = 1.0$, each individual will draw that maximum amount of resource, until eventually the initial pool is depleted, and once again Equation (3.3) will govern the amount of resource each individual receives. In this case the number of individuals initially targeting a resource could easily overshoot the number that the resource can support in a stable state, and this subpopulation may quickly decline after its initial growth phase to reach its final stable size. Figure 3.1 shows the initial growth phase and resource depletion when a new resource is targeted in an Avida population, given $C_f = 1.0$ and both high and low m (2 and 5000, respectively).

The population holds 1000 organisms and was initially filled with organisms that performed a simple task yielding a constant reward that doubled their execution speed. We allowed resources to accumulate for 200 updates (with $I = 100$ and $O = 0.01$) before we placed five individuals into the population that performed a second simple task corresponding to the accumulated resource. These new individuals received a bonus that multiplied their execution speed by 2^A where A is the amount of resource they consumed. There were no mutations and we fixed the gestation time of all individuals to be 30, such that there was approximately one population generation per update. At a stable state we expect the resource to be able to support 100 individuals (10% of the population) performing the second task because Equation 3.3 determines that each individual will receive 1.0 units of resource at this subpopulation level, and thus have an equal fitness to the individuals performing the first task.

As expected the large m caused the subpopulation performing the second task to quickly grow to 100 individuals and then maintain that level, while the low m caused the second

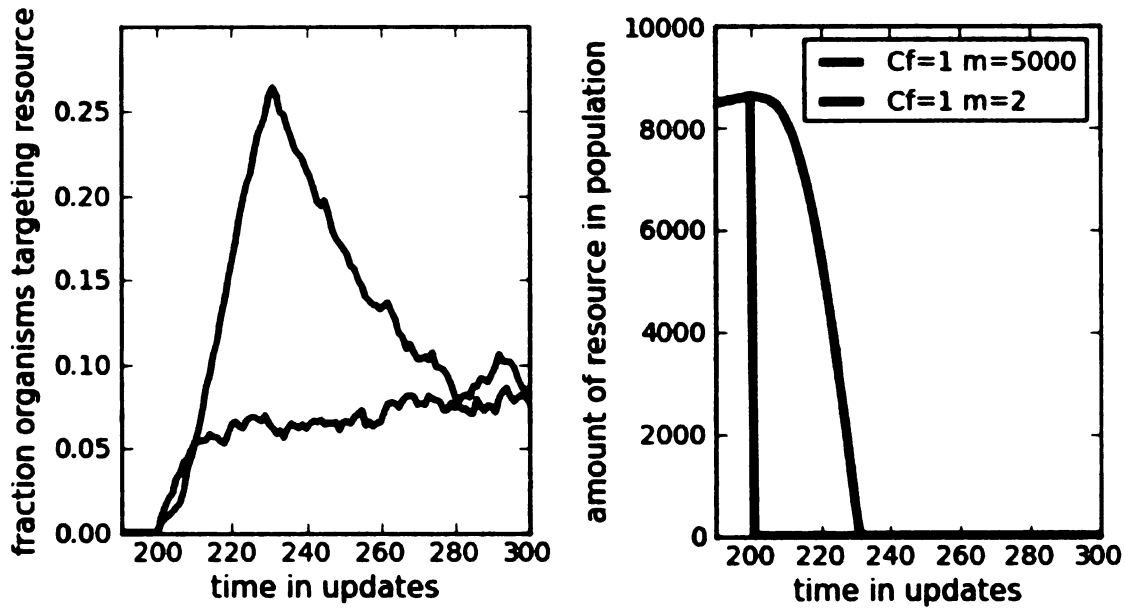


Figure 3.1: Effects of maximum consumption parameter on limited resources population dynamics. (a) The fraction of organisms in a population targeting a limited resource after introduction and (b) the level of the targeted resource over time (averaged over 5 experiments). The resource inflow (I) is 100 units per update (for this population 1 update = 1 generation), the outflow fraction (O) is 0.01, and the consumption fraction (C_f) is 1.0. Two different maximum consumption values (m) are shown, one effectively unlimited ($m = 5000$) and one small ($m = 2$).

task subpopulation to overshoot the stable state level, but decline as soon as the initial resource pool was depleted to reach the stable state level. Note that a value for m of less than 1.0 would cause the subpopulation to never grow as individuals would have a lower fitness than the base population.

An intuitive sense of the effect of the consumption fraction is more difficult to infer, but we expect that a higher C_f will minimize the overshooting of the expected stable level of the subpopulation and cause a faster return to the stable level if it is overshot. This dynamic should occur because the initial pool of resource will be consumed as quickly as possible and Equation (3.3) will take over. We performed the same experiments with a range of consumption fraction values and the two extreme maximum consumption values discussed above as well as a middle level. Figure 3.2 shows the initial growth phase and resource depletion for the effectively unlimited maximum consumption ($m = 5000$). The lower consumption fractions do overshoot the expected stable state subpopulation level, while the highest more slowly and steadily grow to reach it.

The lowest C_f of 0.00025 allows the first individuals using the resource to obtain just over 2 units ($> 8000 * 0.00025$), causing the slowest initial growth, however the slow consumption also allows for the longest growth curve as it takes a long time for the resource to be depleted to its stable state level (4000 from Equation (3.6)), as well as the largest overshot of the expected subpopulation level. Note that this curve crosses the curve for a C_f of 1.0 when the subpopulation reaches approximately 50 individuals; the $C_f = 1.0$ curve is governed by Equation (3.3) at this point, which means each individual in this subpopulation is also receiving approximately 2 units of resource (100/50).

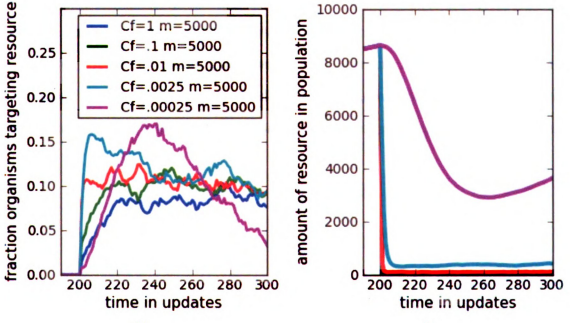


Figure 3.2: Effects of consumption fraction on limited resources population dynamics (infinite max). (a) The fraction of organisms in a population targeting a limited resource after introduction and (b) the level of that resource over time (averaged over 5 experiments). $I = 100$, $O = .01$, and $m = 5000$. Several consumption fractions are shown, ranging from $C_f = 0.00025$ to $C_f = 1.0$.

Figure 3.3 shows the effects of different consumption fractions when the maximum consumption is limited. At an intermediate setting for m all of the consumption fractions caused the subpopulation to overshoot the stable level of 100 individuals, but as expected the higher the C_f, the less this level was overshot, and the more quickly the subpopulation returned to the stable state. At a low value for m all consumption fractions caused a long, slow growth phase as they gradually depleted the resource. Note that even at the lowest maximum limit of 2 the $C_f = 0.00025$ subpopulation is still not limited by m but only by the C_f; its growth curve is qualitatively the same at all three values of m.

Another effect of the consumption fraction is to determine how stable a population that has reached a steady state really is. Avida creates a relatively noisy environment, akin to natural environments, where drift can be as strong a factor as a weak level of selection.

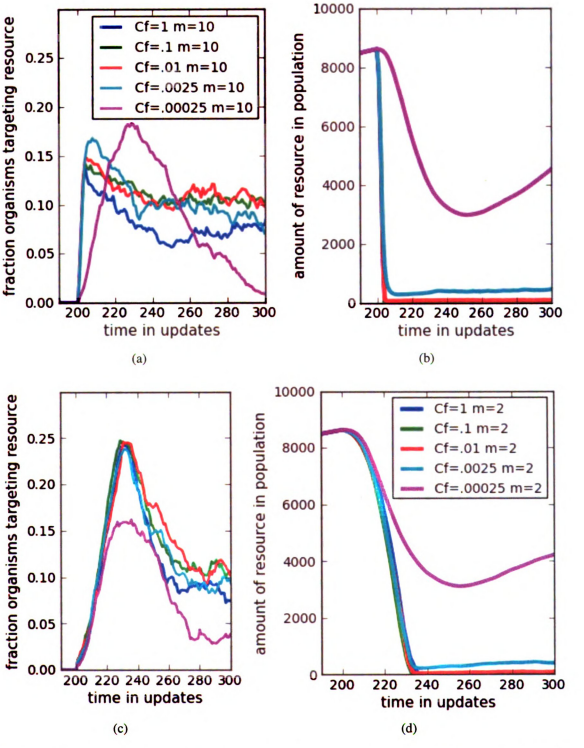


Figure 3.3: Effects of consumption fraction on limited resources population dynamics (limited max). (a) and (c) The fraction of organisms in a population targeting a limited resource after introduction and (b) and (d) the level of that resource over time (averaged over five experiments). Same data as in Figure 3.2 for two more values of m ((a) and (b) $m = 2$ and (c) and (d) $m = 10$). $I = 100$ and $O = .01$. Several consumption fractions are shown, ranging from $C_f = 0.00025$ to $C_f = 1.0$.

A low consumption fraction allows large swings in the number of individuals targeting a resource to cause only small fitness changes. This dynamic can make the negative frequency-dependent selection required to maintain diverse subpopulations too weak to overcome the inherent noise in the environment. Figure 3.4 shows the subpopulation levels given the five consumption frequencies over a longer period of time. The results are similar for all of the tested maximum limits, at $C_f = 0.00025$, the subpopulation levels oscillate widely and often are lost completely.

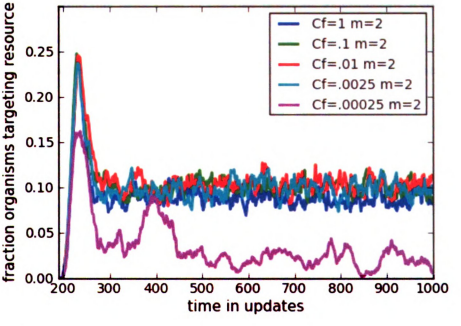


Figure 3.4: Effects of consumption fraction on stable state population dynamics. The fraction of organisms in a population targeting a limited resource after introduction. $I = 100$, $O = .01$, and $m = 2$. Several consumption fractions are shown, ranging from $C_f = 0.00025$ to $C_f = 1.0$. At the lowest consumption fraction negative frequency-dependent selection is weak and subpopulations are often lost through drift. Results are similar for all tested maximum limits.

3.3 Conclusion

While we found that while extreme values for the consumption fraction and maximum limit can have significant effects, the differences are minimal during the growth phase and virtually nonexistent at a stable state for a wide range of values; several orders of magnitude for the C_f. For the work in this thesis we vary only the inflow rate and maximum value, using the equations discussed previously to set those appropriately to ensure limiting resources. The outflow, as mentioned earlier, is always 0.01, and the consumption fraction we set to 0.0025, as this value has been used in the few previous examples of Avida experiments using limited resources, and we have shown here that it is high enough to cause selection that can overcome environmental noise and maintain a stable population state.

Chapter 4

The Effects of Limited Resources on the Evolution of Complex Features

In this chapter I discuss experiments using the Aivda digital evolution system presented in Chapter 3, to explore competition for limited resources as a mechanism to promote diversity and evolve complex traits more rapidly and reliably. Incorporating limited resources into an Avida environment has been shown to lead to the evolution of stable ecosystems of diverse species (Cooper & Ofria, 2002; Chow et al., 2004). I examine how and why maintaining a diverse ecosystem in Avida through the use of limited resources leads to improved adaptive evolution of complex features.

4.1 The 9 Logic Task Environment

In order to explore competition for limited resources as a mechanism to promote diversity and lead to faster evolution of complex traits in Avida, we start with the default Avida environment, which is the most widely studied. This environment, called "9-logic" contains unlimited recourses associated with nine Boolean logic tasks and was used to study the evolution of complex features (Lenski *et al*, 2003). Lenski *et al.* demonstrated to un that Avida organisms could evolve the ability to perform complex logic functions requiring the

coordinated execution of many instructions as long as (1) functions of simpler complexity were rewarded so that they could be used as building blocks, and (2) there were many possibly ways of evolving the complex functions. Here we repeat this study to examine the origin of complex functions in environments with limited resources both to compare their effectiveness to the 9-logic environment and to determine if there are any differences in the underlying forces that drive this process. Our hypothesis is that limited resources environments will produce organisms with complex traits more rapidly due to their higher levels of diversity.

We created a new environment called 9-resource. Tasks in this environment are associated with limited resources, but it is otherwise identical to 9-logic. Previous research on limited resources environments in Avida has shown that populations in these environments can stably maintain more diverse ecosystems (Cooper & Ofria, 2002) and that species richness is highest at intermediate (but still limiting) levels of resource inflow (Chow et al., 2004).

Each logic task in the environment requires organisms to take as input one or two 32-bit strings, and output the result of a bitwise Boolean logic operation. The nine operations are listed below in table 4.1 in order of increasing complexity; one of the simplest, NAND, is available as an instruction that can be mutated into an individual's genome, while the most complex, EQU, requires at least five nand operations and more than 15 total instructions to complete (nature complex features paper). Each of the nine logic tasks is assigned a complexity value based on the minimum number of nand operations that are required to perform the task. Table 4.1 shows the nine tasks and the complexity value of each, given two inputs A and B.

When an organism performs a logic task (by outputting the correct result), it receives an increase to its metabolic rate (i.e., an execution speed bonus) determined by the following equation, where V is the complexity value of the task from table 4.1 and A is the amount of the corresponding resource it consumes.

Task Name	Complexity Value	Logic Operation
NOT	1	~A; ~B
NAND	1	~(A and B)
AND	2	A and B
OR_N	2	(A or ~B); (B or ~A)
OR	3	A or B
AND_N	3	(A and ~B); (B and ~A)
NOR	4	~A and ~B
XOR	4	(A and ~B) or (B and ~A)
EQU	5	(A and B) or (~A and ~B) i.e. are A and B identical

Table 4.1: The 9-logic environment. The complexity values are the minimum number of nand operations required to complete a task. The ~ symbol represents bitwise negation. In cases where logic operations are separated by a semi-colon, either is accepted as completion of a task, but individuals are only rewarded for performing each task only once.

$$Bonus = 2^{A*V} \tag{4.1}$$

In the 9-logic environment an organism always consumes exactly 1 unit of resource when performing a logic function. This means performing NOT doubles its speed of execution while performing EQU causes it to execute 32 times as fast. In the limited resources environment the amount consumed is determined by the resource parameters discussed earlier, and the current availability of the appropriate resource. An organism may perform each task at most once, and all bonuses are multiplied together to determine the total bonus the organism will receive. An organism's metabolic rate is updated only upon replication; each organism executes at a speed based on the performance of its parent.

The focus of the study by Lenski *et al.* was the evolution of the most complex operation, EQU. The authors found that EQU evolved only when at least some of the other eight operations were rewarded, although no specific operation was required. The simpler operations act as building blocks toward the final complex task of performing EQU and there are many ways to use those simpler tasks to build up to the EQU task. We repeated the 9-logic experiments from the previous Lenski *et al.* study using the newest version of the Avida software (2.9) to ensure a fair comparison. The results were qualitatively similar to the original data.

4.2 The Effects of Limited Resources

To examine the effects of limited resource inflow, we must first determine how far we want to limit the resources and how they will compare the original, unlimited experiments. To choose the range of inflow rates to explore, we refer back to Equation 3.5, and need to determine values for T_r (the number of updates an organism requires to reproduce), m (the maximum amount of resources useable by an individual), and the population size.

To calculate T_r, we assume that the population will have a gestation time similar to

the ones found in the 9-logic environment. The average gestation time in the 100 9-logic control replicates that we performed is 189 (s=111), with the extreme outliers at 99 and 950; T_r can be approximated by dividing gestation time by 30, so this result puts T_r in the range 3.3 to 31.7, with a mean of 6.3. Next, we plan to keep population size at 3600 and we set m to 1, the amount of resource that can be used in the unlimited resource trials. Plugging these values into Equation 3.5 gives $I < 545$ for the mean gestation time, and the extremes of the range give $I < 116$ and $I < 1090$. As such, we performed experiments with inflow rates of 10, 30, 100, 300, and 1000. We expect any inflow less than 545 to be at least slightly limiting, on average. Figure 4.1 shows for each treatment the number of populations, out of 100, that have at least one EQU-performing individual at a given update.

At the end of 100,000 updates of evolution, the four treatments using inflow rates of less than 545 all yielded significantly more populations performing equals than the treatments using an inflow rate of 1000 or with unlimited resources (Fisher's exact test with Bonferroni correction for multiple comparisons, largest p value comparing 67 (inflow 10) to 24 (inflow 1000) was $1.3e^{-9}$ before correction). Why did limiting resources lead to so many more populations evolving EQU?

In order to answer this question we looked more closely at individual populations from experiments using the unlimited resource environment and those using an inflow rate of 100 (as representative of the limited resource environments). We found that, as expected, the populations that had evolved in a limited resources environment had higher phenotype diversity as measured using the Shannon index ($p < .001$ Mann-Whitney U test). We defined phenotypes in this experiment based on the exact set of logic tasks an individual performs; given nine logic tasks there are 2^9, or 512 possible phenotypes. The higher phenotype diversity found in the limited resource treatment supports the theory that EQU is more likely to evolve when the population is sampling more of the search space giving more potential paths for evolution to follow. However an analysis of the lineages of the final EQU-performing individuals from each population yielded two other potential theories

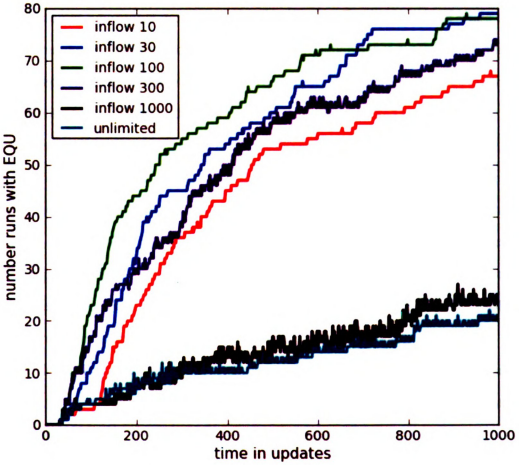


Figure 4.1: Comparing resource inflow rates when evolving EQU. Number of populations that currently contain at least one individual that performs the EQU task out of 100 total populations for each treatment, over time.

for why EQU appears more in the limited resources environments. The next two sections cover these alternate possibilities.

4.2.1 Deleterious Mutations on the Path to EQU

The first alternate theory we explored is based on the value of deleterious mutations in limited resources environments. This class of mutations that were harmful when they first appeared frequently occurred along the lineages of organisms that evolved to perform EQU in the limited resources environment, but were rarer in the unlimited resource environment. In both types of environments these deleterious mutations caused the loss of one or more tasks without any concurrent gain, but were followed soon after by a beneficial mutation that gave the individual a different set of tasks and a higher overall reward. While a mutation that causes only the loss of tasks is deleterious in both environments, it is much less deleterious in the limited resources treatment assuming the tasks lost are being performed by a large portion of the population and thus the corresponding resources are heavily depleted. We know that deleterious mutations can be important for adaptive evolution in unlimited resource populations (Covert, personal communication), and thus we propose that these deleterious mutations may be important to evolving EQU. Thus an advantage of the limited resources environment could be that it allows individuals with these mutations to survive long enough in the population to undergo the potential compensatory beneficial mutation whereas in the unlimited resource environment they are removed by selection too quickly.

We found mutations causing task loss with no concurrent gain in 57 of the 79 lineages that produced EQU in populations evolved in the limited resources environment, and in 16 of these this type of mutation immediately preceded the mutation that introduced EQU into the genome. We found a mutation causing task loss with no gain in only 3 of the 21 lineages that produced EQU in populations evolved in the unlimited resource environment, none of which were followed by the evolution of EQU. To test whether these mutations

are significantly increasing the evolution of equals we performed 100 runs in each environment where we did not allow task loss to occur. Specifically each time a mutation occurred we tested the mutated genome before placing the individual in the population, and if the mutation had caused the individual to lost one or more tasks without gaining any others, we "sterilized" that individual. A sterilized individual is still put in the population and uses CPU cycles, but it is not allowed to replicate. These experiments resulted in 31 populations performing EQU at 100,000 updates in the unlimited resource environment, and 74 populations in the limited resources environment. These are both consistent with the previous number of populations that perform EQU (Fisher's Exact test, 31 vs. 21 $p = .14$, 74 vs. 79 $p = .5$). Thus we conclude that while these mutations may be used on the path to evolving EQU, they are not necessary nor a factor in the success of the limited resources environment in evolving EQU.

4.2.2 Trading Less Complex Tasks to Gain EQU

A second pattern we noticed while analyzing the lineages was that individuals from both treatments generally lose other tasks when they gain EQU, implying that EQU is often evolved by co-opting part of the functionality of one or more of the other tasks. In the limited resources environment these trade-offs almost always yield an individual with a higher fitness no matter how many tasks are lost to gain EQU, because the unused resource associated with EQU accrues such that any individual performing EQU receives a large amount of energy. In the unlimited resource environment the reward for performing EQU is higher than that of any other task, but an individual that trades two or more lesser tasks to gain EQU will often lose energy overall and be selected out of the population. Thus we posit our second theory that EQU appears more in the limited resources environment because it is more conducive to EQU evolving by co-opting the functionality of lesser tasks.

We found that, on average, 3.07 tasks were lost to gain EQU in the unlimited resource environment, and 2.4 in the limited resources environment. For each final population that

evolved EQU, we took the most common EQU-performing individual and found the mutation that first introduced the ability to perform EQU in its lineage. In every one of the 21 lineages from populations evolved in the unlimited resource environment the mutation that introduced EQU increased the fitness of the individual. Of the 79 lineages from populations evolved in the limited resources environment, 65 of the mutations that introduced EQU would have been either neutral or deleterious had they occurred in the unlimited resource environment (33 neutral and 32 deleterious). These data support the theory that EQU often evolves in the limited resources environment by trading off other tasks in a manner that would not be beneficial in the unlimited resource environment.

We tested this theory directly by increasing the reward for equals in the unlimited resource environment by changing the complexity value for equals from 5 to 21. A value of 21 means that gaining EQU will be beneficial to an individual even if it loses every other task in gaining it. We found that the number of populations performing EQU at the end of 100,000 updates increased significantly from 21 to 48 ($p < .0001$ Fisher's Exact test). Clearly the ability to trade off other tasks for EQU is an important factor, but still a significantly higher number of limited resources populations evolved EQU (79) than even the 48 that now performed EQU in the updated unlimited resource environment ($p < .0001$ Fisher's Exact test), thus we conclude that easier tradeoffs for EQU are not the only factor.

Rewarding EQU with a complexity value of 21 only addresses the potential to trade off other tasks for EQU itself, however the ability to build more complex tasks on simpler tasks could affect the evolution of tasks of intermediate complexity as well. We performed a final test where we changed the complexity values for all of the tasks except the two simplest, such that the reward for gaining each new complexity tier was large enough to overcome the loss of all simpler tasks. Specifically, we set the reward values for the nine tasks to 1, 1, 3, 3, 9, 9, 27, 27, and 81. We found that using this reward structure, 56 of the populations evolved in the unlimited resource environment performed EQU at 100,000 updates, still significantly fewer than the 79 limited resources populations ($p < .0001$ Fisher's Exact

test). Trading off simpler tasks in order to gain more complex tasks is clearly important in the eventual evolution of EQU, but it appears the limited resources environment yields an advantage beyond changing the reward structure to encourage tradeoffs.

4.2.3 Diversity

Testing the trade-off theory highlighted the fact that populations sometimes evolved EQU but then lost it. This revelation caused us to question whether the limited resources environment actually increases the likelihood of EQU appearing in a population or only the likelihood of it being maintained in the population once it appears. If the increased diversity in the populations evolved using limited resources is in fact providing evolution with more paths to EQU, then EQU should appear in these populations more than in populations evolved in the unlimited resource environment, regardless of whether it is maintained. Figure 4.2 shows the number of populations that have ever "touched" EQU, i.e. had an individual placed in the population that is capable of performing EQU, as well as the number of populations that currently have an EQU performing individual at a given update.

It appears that EQU does, in fact, appear earlier in the populations evolved in the limited resources environment, but by 100,000 updates the difference in the number of populations that have encountered EQU is negligible. To test if EQU also appears more often, we perform 100 more replicate runs in each environment, but with the change that we remove the reward for EQU, but still keep a count of how many times an individual has a mutation that would cause it to perform EQU. We do not actually allow an individual that would perform EQU to be placed into these populations, but instead replace any mutation that would introduce the ability to perform EQU, and continue to do so until we find a mutation that does not confer the ability to perform EQU. Table 4.2 shows the means across all 100 experiments of each treatment for the first update at which a mutation occurs that would have caused an individual to perform EQU, as well as the average number of times per 1,000 updates an EQU-causing mutation appears after the first. Populations evolved in a

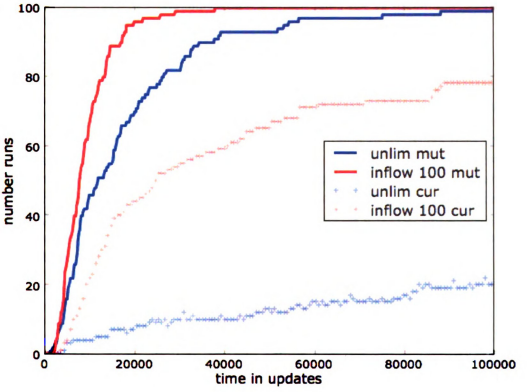


Figure 4.2: Number of populations that ever touch EQU vs. number that currently perform EQU. Solid lines show the number of populations across all 100 replicates that have ever had an individual that is capable of performing EQU, regardless of whether it is maintained. The + lines show the number of populations that currently have an individual capable of performing EQU (the same data from Figure 4.1 for inflow 100 and unlimited resources environments). The limited resources environment not only causes more populations to maintain EQU, but for populations to first discover an individual that can perform EQU earlier.

limited resources environment touch equals both significantly earlier and more often than those evolved in an unlimited resource environment ($p < .001$ Mann-Whitney U test).

Resources	First update touches EQU	Number EQU touches after first per 1k updates
Unlimited	17848	40.4
Limited	9251	150.1

Table 4.2: First update and number times populations touch EQU. Mean data across 100 replicate populations for each of two environments, one using limited resources and one using an unlimited resource. The first update a population touches EQU is determined by the first time the offspring of an individual undergoes a mutation that would cause it to perform EQU. We count number of times a mutation causes this type of phenotypic effect (after the first) over every 1,000 updates of Avida time and we present the mean of those data in the third column of the table.

At first, these results indicate that the most likely explanation for the more frequent appearance of EQU in populations evolving in a limited resources environment is that these populations were more diverse and thus there were more potential paths to EQU for evolution to follow. On further inspection, however, we found that populations evolved in a limited resources environment using an inflow parameter of 10 were even more diverse than using an inflow parameter of 100 (in terms of Shannon diversity) and yet performed worse based on all three metrics for the evolution of EQU. In fact the inflow 10 treatment performs on par with the unlimited resource environment in the two measures of how often a mutation causes EQU to appear in a population. Table 4.3 summarizes the data for the three treatments.

	Num runs perform EQU at 100k updates	First update touches EQU	Number EQU touches after first per 1k updates	Shannon diversity
Unlimited	21	17848	40.4	.31
Inflow 100	79 (159/200)†	9251†	150.1†	.79†
Inflow 10	67 (135/200)†*	16796*	22.4*	1.34†*

Table 4.3: EQU data for different resource environments. Mean data across 100 replicate populations for each of three environments, one with unlimited resources and two with limited resources using different inflow parameters, 10 and 100, respectively. The † symbol designates a value that is significantly different from the unlimited resource environment data. The * marks values from the inflow 10 limited resources environment that are significantly different than the inflow 100 environment. We used Fisher's Exact test to determine significance for the first column data, and Mann-Whitney U tests were used for all other comparisons, as the data were not normal.

To understand why the more diverse populations from the inflow 10 treatment might be less likely to evolve EQU we looked at the number of tasks the average individual in the population is performing. Lenski *et al.* found that EQU is built on the other eight subtasks, and we showed earlier that individuals on average trade off more than two other tasks when they gain EQU. Thus it is likely (and intuitive) that the more subtasks an individual is performing, the more opportunity for EQU to evolve by co-opting the functionality of one or more of those tasks. Figure 4.3 shows the average number of distinct tasks being performed by at least one individual in each population, and the average number of tasks being performed by each individual itself (averaged only over individuals that are doing any tasks at all). At a population level the inflow 10 treatment performs just as many tasks as the other two, but each individual in the population performs significantly fewer tasks. In fact the average individual from the inflow 10 treatment performs fewer tasks than are generally lost by individuals in the other two treatments in order to gain EQU. It seems highly likely that the low number of subtasks performed by any one individual makes it difficult for EQU to evolve in this environment.

To test if the low number of tasks performed by the average individual is the factor degrading the performance of the inflow 10 treatment in evolving EQU, we performed another set of 100 experiments in each of the three environments where we restricted the number of tasks any single individual was allowed to perform to three. If an individual performed more than three tasks it is was not rewarded for any but the first three. Table 4.4 shows the same data as table 4.2 for the experiments with the three task limit per individual. As expected the inflow-100 and unlimited resource treatments perform significantly worse when individuals are limited to performing no more than three tasks, while the inflow 10 treatment performs equally as well as before. The overall result is that the performance of the two limited resource treatments is now qualitatively equivalent in all three measures of EQU evolution while the performance of the unlimited resource treatment is significantly worse than either (see table 4.4 for statistics). This is consistent with the theory that higher

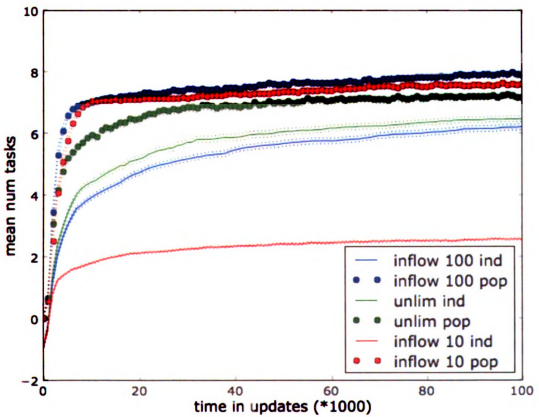


Figure 4.3: Mean number of tasks performed by populations vs. individuals. Means across 100 replicate populations from three environments, one with unlimited resource and two with limited resources using inflow parameters of 10 and 100 respectively. The population lines represent the number of tasks being performed by the entire population at that time, though different individuals are performing different tasks. The individual lines represent the mean number of tasks being performed by only individuals in the population that are performing at least one task, zero task organisms are not counted.

diversity is driving the increased evolution of EQU. Note that the populations evolved in the inflow-10 environment are still more diverse than those evolved using an inflow of 100, but the difference is slight and perhaps not enough to make a difference in the evolution of EQU.

	Num runs perform EQU at 100k updates	First update touches EQU	Number EQU touches after first per 1k updates	Shannon Diversity
Unlimited	10	29567	3.5	.23
Inflow 100	63†	15230†	19.0†	1.01†
Inflow 10	64†	16933†	18.2†	1.22†*

Table 4.4: EQU data when restrict individuals to perform no more than 3 tasks. Mean data across 100 replicate populations for the same three environments as table 4.3 but where individuals are restricted to being rewarded for at most three tasks. The † symbol designates a value that is significantly different from the unlimited resource environment data. The * marks values from the inflow 10 limited resources environment that are significantly different than the inflow 100 environment. Fisher's Exact test was used to determine significance for the first column data, and Mann-Whitney U tests were used for all other comparisons, as the data were not normal.

4.3 Conclusion

In this chapter I have shown that competition for limited resources increases the diversity of populations of digital organisms, and improves the evolution of complex traits. Some of this improvement is caused by the reward structure inherent to the limited resource environment; individuals that trade off several subtasks in order to gain a new task still receive a fitness bonus as the unused resource associated with the new task is abundant. However we showed that the most complex trait, EQU, actually appeared more often and earlier in populations evolved in the limited resources environment, regardless of whether it was maintained. The most likely reason for this increased appearance is that the higher phenotypic diversity caused by competition for limited resources creates more potential paths evolution can follow to produce the complex trait.

Chapter 5

Using Limited Resources to Evolve Diverse Solutions

In Chapter 4 we showed that competition for limited resources can improve the evolution of a single complex trait that is built on subtasks. However, much of the previous work on sharing techniques in evolutionary algorithms is focused not on evolving a single solution to a complex problem, but rather on obtaining a final population of diverse solutions that together represent the overall solution to a problem. For example in learning classifier systems the overall solution is an entire population of rules, or in an artificial immune system a population of antibodies. In this chapter I explore the dynamics of incorporating competition for limited resources into a simplified version of the Avida digital evolution system used in Chapters 3 and 4, both to speed up the evolution of high quality solutions and to increase the diversity of solutions co-existing to "cover" a solution space, rather than relying on a single all-purpose final solution to a complex problem.

5.1 Binary String Cover Problem

I use binary string covering as a diagnostic problem for this study because it is relatively simple and will allow a clear analysis of the behavior of an environment with and without

limited resources. Binary string covering has also been used in several other studies on niching techniques, including Forrest and Smith's implicit fitness sharing in a simple immune system (Forrest *et al.*, 1993; Smith *et al.*, 1993), and Potter's population subdivision algorithm for automatic problem decomposition (Potter and DeJong, 2000). The problem consists of finding a binary vector x that is as close a match as possible to a set V of N binary vectors. The match value for x on each vector v_i in V is defined as

$$S(x, v_i) = \begin{cases} 0 & \text{if } b_i < l/2 \\ (\frac{2b_i}{l} - 1)^2 & \text{otherwise} \end{cases} \tag{5.1}$$

where all strings are of length l and b_i is the number of bit positions perfectly matched between x and v_i. Unless the vectors in set V are all identical no single value of x will be able to perfectly optimize all objectives, and trade-offs must occur. Forrest *et al.* show that this simple problem has applications as an abstract version of an immune system. Each vector in the set V may be viewed as an antigen (a foreign cell), and each vector in the population as an antibody. The immune system is able to recognize a large number of antigens with a much smaller number of antibodies. A population of diverse antibodies must be maintained with a balance between generalists and specialists to best match any antigen. We analyze the behavior of our system in this case and the conditions necessary to maintain multiple match vectors (antibodies) in the population that cover different vectors in V (antigens). The match score of an antibody on a given antigen is based on the number of bits that are the same, which is posed slightly differently than in Forrest et al. where a match was based on complimentary bits, but for all practical purposes these problems are identical.

Each vector in the set V can be viewed as a single objective in a multi-objective function. The overall fitness of a vector x is based on its optimization of match values for each objective vector. However unlike a typical multi-objective problem, our goal is not to find a pareto-front of all possible trade-offs, but to evolve a population that covers all antigens

as well as possible. Therefore our fitness function is designed to encourage the precise matching of bit strings when possible, but if N is large it becomes more important to find generalists matching specific patterns that are common to many vectors in set V than to match any single string.

5.2 Avida as a Genetic Algorithm

These experiments were performed using a simplified version of Avida that functions as a steady-state genetic algorithm. Examining the effects of competition for limited resources in a traditional genetic algorithm allows me to directly compare between the results of these experiments and those of previous niching studies. It is also a first step in showing that this method could be implemented in any EA, and simplifies analysis of the effects of using limited resources versus a traditional genetic algorithm. I chose to implement this genetic algorithm within the Avida framework because it will facilitate transitioning to using the full version of Avida when applying this method to solve more complex problems.

In this simplified version of Avida each digital organism consists of a binary sequence of ones and zeros. We use an explicit fitness function to evaluate their match to each antigen sequence in the environment. The organisms do not self-replicate; instead they are automatically replicated in time inversely proportional to their fitness; a higher fitness means faster replication. The limited resources environment remains the same, however, and the organisms are still placed on a toroidal grid giving a spatial component to the environment. For this problem, each organism encodes a binary string representing one antibody. An additional change in this version of Avida to more closely resemble a genetic algorithm is that organisms reproduced sexually, crossing over only with neighboring organisms on a toroidal lattice and placing offspring adjacent to at least one parent. The mating-restriction and replacement strategy yield an implicit crowding effect, which helps to maintain rare species and reduces the probability of crossover between dissimilar individuals.

5.3 Binary String Cover Problem in Avida

Upon replication, an organism is evaluated by determining its match quality as compared to each string in V, using the function S defined above. Note that if an organism matches fewer than half of the bits for a given string, it receives a match quality of 0. For each match score greater than 0, the organism receives a fitness bonus determined by the match score and the environment. The bonuses for each match are multiplied together to attain the final fitness of an organism. The environments with limited resources (as discussed in Chapter 3) are compared to those with unlimited resources for each of the following experiments. The limited resources environment includes one resource for each string in V; the bonus an organism receives for matching a string is therefore dependent not only on the match quality S but also on the availability of the corresponding resource. The bonus an organism receives and the amount of resource consumed from the environment (A) are determined by the following equations, given a current availability of R units of resource in the environment, and a maximum cap on consumption m.

$$A = min(S * C_f * R, m) \tag{5.2}$$

$$Bonus = 2^A \tag{5.3}$$

I set the resource inflow and maximum to appropriate levels based on the findings of Chapter 3. In this simplified version of Avida we fixed the gestation time of individuals such that approximately one population generation occurs each update, or T_r 1. The experiments in this chapter use smaller population sizes ($N = 200$ or $N = 500$) than those in Chapter 4, to more closely resemble the typical population sizes used in previous niching studies. For the smallest population size, Equation (3.5) thus gives

$$I < 200 * m \tag{5.4}$$

In Chapter 3 we found that a higher maximum consumption value (m) limits the amount subpopulations overshoot expected stable state levels. At these smaller population sizes this may be important, so we set m at 5 instead of 1 as in Chapter 4. The inflow therefore must be less than 1000 in order to be limiting; we set I at 100. In the unlimited resource environment the bonus each organism receives is based only on the match quality S, but with a multiplier of 5 to match the maximum setting in the limited resources environment.

5.4 Effects of Limited Resources

We first tested that limiting the resources available for matching each antigen allows our system to find and maintain diverse types of antibodies in the population. We created an environment with three antigens each associated with a unique resource. The bit strings corresponding to these antigens were:

111100110000111100110000

000011110000111100001111

111100001111000011110000

These sequences represent three distinct peaks in the fitness landscape (later, we explore the behavior of our system given overlap between peaks). Figure 5.1 shows the number of antibodies in the populations that perfectly matched each of the antigens after 100 generations, given the two environments described above. Two sample populations are included along with the average results across ten populations. Each population consisted of 200 individuals.

The unlimited resource environment led populations to find and perfectly match exactly one antigen, although different populations matched different antigens. Each population evolved using the limited resources environment, however, diversified to perfectly match all three antigens, with approximately 1/3 of the antibodies in a population matching each of the antigens. Note that the first string shares half of its bits with each of the other two,

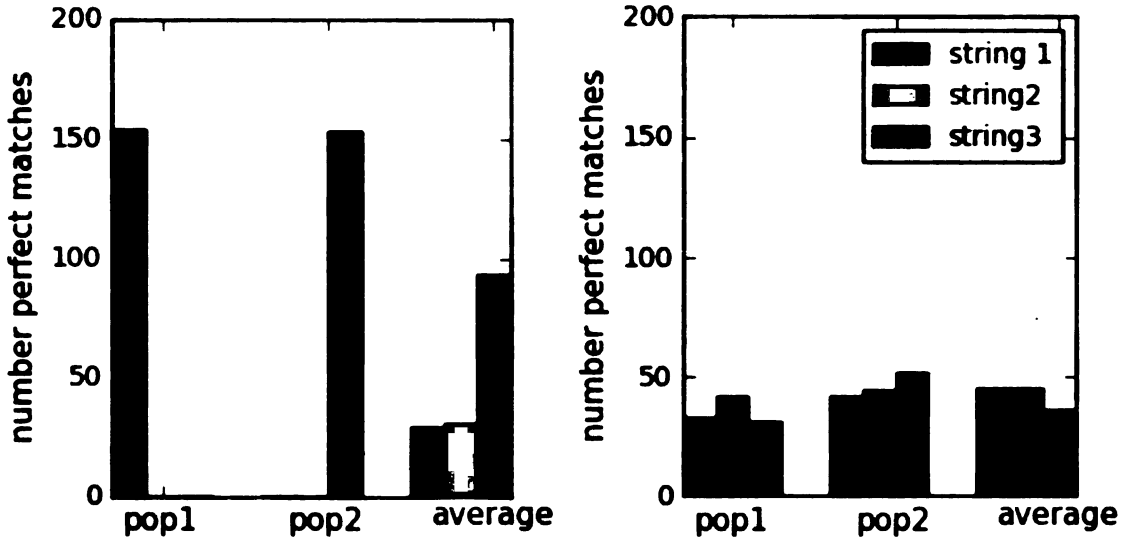


Figure 5.1: Matching antigens. Number of antibodies produced that perfectly match each of three antigens (blue, red, and green represent antigen patterns 1, 2, and 3 respectively) after 100 generations of evolution in populations of 200 individuals. **(a)** Two typical populations and the average of ten populations using an unlimited resource environment. Each population is able to maintain only one type of antibody that perfectly matches a random antigen. **(b)** The same data for populations using limited resources where one resource is associated with each antigen. Populations reach a stable state with approximately an equal number of individuals producing antibodies to match each antigen.

while the second two strings are opposites of each other. It is likely that this is why the unlimited resource environment converges to the first string in the match set more often than either of the other two; in every population a generalist immediately sweeps that matches more than 1/2 the bits of two strings, either the first and second strings or the first and third strings from the match set. The match score S encourages specialization, however, and thus eventually the entire population converges to perfectly match one or the other of the two initial strings. The fact that the first string in the match set is always one of the two strings initially present gives it twice as high a chance of being the final string in the population as either of the last two strings in the match set. We performed 40 more replicate experiments for a total of 50 evolved populations and found that antibodies matching the first string from the match set swept 26 populations, while antibodies matching the second and third strings each swept 12 populations, exactly as predicted by the proposed explanation.

5.4.1 Ability to Generalize Appropriately

Limiting resources in the environment allowed the system to maintain diverse antibodies perfectly matching different antigens in this simple case, but we are interested in how it will generalize to match a large number of antigens given a smaller number of antibodies. To explore this question we created a set of 100 antigens, but we embedded patterns among this set to analyze the ability of our system to automatically find and exploit these patterns in a case where it's not practical to perfectly match every antigen.

We built a match set with 100 antigens and embedded one of the following three patterns into each antigen.

110011001100

001100110011

110100110100

These patterns were placed into every other bit of an antigen's sequence, so an antigen

built on the first pattern looks like

1#1#0#0#1#1#0#0#1#1#0#0#

where the # represents a random bit. We created 33 different antigens following this first pattern, then 33 and 34 of the other two patterns, respectively. Figure 5.2 shows the number of antibodies in a population that perfectly matched each of the three patterns given above (ignoring the random bits). In this experiment, each population contained 500 individuals. Both environments found and exploited patterns, but once again limiting resources in the environment allowed the populations to diversify and maintain antibodies matching each pattern, while the unlimited resource environment was able to find antibodies matching only one pattern in each population. Note that even across all 10 replicate experiments the unlimited resource environment never discovered one of the three patterns.

Potter noted that his population subdivision method found antibody solutions that matched slightly more than the expected number of bits when averaged across all of the antigens, and that up to a point the algorithm did slightly better, on average, the more species it subdivided into. The reason for this improvement lies in the random bits alternating with the pattern in each antigen. These bits were chosen randomly for each of the 100 antigens, but simply through chance the distribution of 1's and 0's for each of these bits was not uniform. Therefore it is possible for a population to find less obvious patterns among these random bits and match them as well, creating more than three actual niches. We tested the limited resources environment to see if it exploited these extra niches created by the non-uniformity in the random bits. We found that the average proportion of the random bits matched across all antigens by each of the common antibodies (those that represent at least 5% of the population) was 0.57 in the limited resource environment. This is significantly higher than the 0.5 we would expect if the random bits were completely uniform ($p < .001$, t-test).

The higher average match value seems to indicate the algorithm did find and exploit the niches created by the random bits, however an alternate explanation could be that the

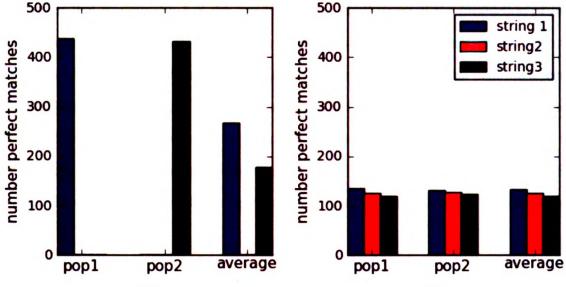


Figure 5.2: Matching embedded patterns. Number of antibodies produced that perfectly match each pattern embedded in the set of antigens after 100 generations of evolution in populations of 500 individuals. **(a)** Two typical populations and the average of 10 populations using 100 unlimited resources, one associated with each antigen. These populations are able to maintain only one type of antibody that perfectly matches one of the patterns. **(b)** The same data for populations using limited resources. Populations reach a stable state with an approximately equal number of individuals producing antibodies to match each embedded pattern.

populations were perfectly matching three individual antigens, one containing each pattern, since the fitness function encourages specialization. Once again we analyzed the common antibodies across all trials and found that only 8% of the antibodies are exact matches to any single antigen. Clearly the individual antibodies were not simply matching single antigens, but were they finding patterns in the random bits? Figure 5.3 matches the proportion of the 100 antigens that contained a 1 at each bit position with the proportion of the dominant antibodies that contained a 1 at the same position. In all cases, if the proportion of ones in the antigens was above 0.5, so was the proportion of ones in the antibodies. Also the further from 0.5 the proportion of ones in the antigens was, the more that position affected the antibodies. Positions 6, 7, 9, and 10 were the only four positions that were skewed more than 0.05 from a proportion of 0.5 ones, and they also clearly had the biggest effect on the antibodies. The populations are achieving the higher match score averages by exploiting less obvious patterns in the 100 antigens arising from the non-uniformity in their random creation.

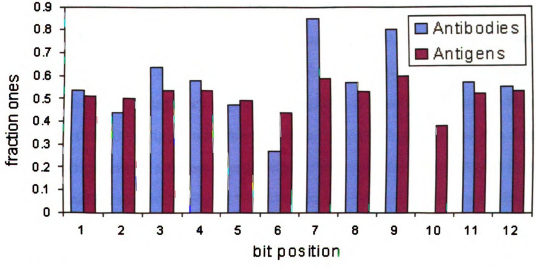


Figure 5.3: Exploiting hidden patterns. Proportion of each random bit position that is a 1 across all antigens and antibodies. The antibodies exploit the non-uniformity in the distribution of 1's and 0's in the antigens to achieve a slightly higher match than would be expected.

We performed a final test of continuing the evolution of each population for another 1000 generations after the first 100, but without mutations and sex (crossover). This ecological phase allowed us to see how many antibodies the populations were actually maintaining in a steady state. We found that 17 of the 20 populations maintained the same 6 antibodies; the other 3 populations each maintained one extra antibody, but in a small proportion that most likely would not last indefinitely. Of these 6 antibodies, only 1 was a perfect match for any one of the antigens, the other 5 represented niches created by the patterns plus the non-uniform random bits. This result shows the limited resources environment is not only able to exploit frequent and clear patterns quickly and efficiently but also to find less obvious patterns to form a better solution cover.

5.4.2 Frequency Dependent Matching

Forrest *et al.* found that in their system antibodies matching a given antigen occurred with a frequency proportional to the sampling rate for that antigen. Frequency dependent matching of antigens is generally a desired behavior; the more an antigen appears the more antibodies will be needed to find and neutralize it, and the more important it is to have an antibody matching that antigen as closely as possible. We found that the limited resources environment also causes frequency dependent matching, as shown in Figure 4. We once again used a set of 100 antigens with the same three embedded patterns, but this time we embedded each pattern in a different proportion of the antigens, 0.55, 0.3, and 0.15 for patterns 1, 2, and 3 respectively. Figure 5.4 shows the population equilibrium reached after 200 generations of evolution in a population of 500 individuals. Every population reached a similar equilibrium with the antibody patterns present in close proportions to the antigen patterns. The same amount of resource inflow is associated with each antigen, so if many antigens with the same pattern appear in the antigen set, there will be more overall resource available for matching that underlying pattern, and more individuals producing matching antibodies will be maintained in the population. The same experiment performed in the

unlimited resource environment yields all populations matching only the most frequent pattern.

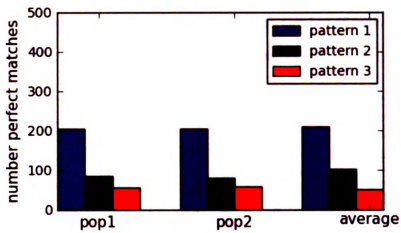


Figure 5.4: Frequency-dependent matching. Number of antibodies produced that perfectly match each pattern embedded in the set of antigens (with uneven frequencies of 0.55, 0.3, and 0.15 respectively) after 200 generations of evolution in populations of 500 individuals using limited resources.

The frequency of patterns in the antigen set also affects the dynamics of antibody evolution. Figure 5.5a shows the highest proportion of each pattern matched by any antibody in the population. The initial population was generated randomly and on average the best match on each pattern was around 80% of the possible bits. The best matches on the two more frequent patterns diverged from the least frequent pattern; the populations quickly evolved to perfectly match the most frequent pattern, and found a perfect match for the second most frequent almost immediately thereafter. The best antibody match for the least frequent pattern actually degraded for the first 30 generations (when a greater selective pressure still existed for the first two patterns) before building up to a perfect match as the population reaches a steady state.

To gain some insight into this pattern of evolution we looked at the number of antibodies matching more than half the bits in a pattern instead of the number of antibodies that match perfectly. One antibody can match more than half the bits in multiple patterns, but

only if it does not match any one pattern perfectly, due to the nature of the patterns. The individuals that produce antibodies matching more than half the bits in a pattern can therefore be thought of as being on a pathway toward evolving that pattern, even if they haven't found the complete match yet. Figure 5.5b shows the number of individuals focusing on each pattern over the first 200 generations of evolution.

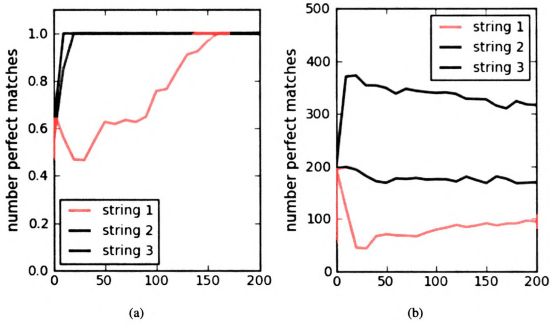


Figure 5.5: Order of pattern evolution. Evolution of antibodies over 200 generations averaged over 10 replicate populations of 100 individuals. **(a)** The proportion of bits in an antigen pattern that the best matching antibody in the population covers. The population evolves to match the two most frequent antigen patterns quickly, but takes over 150 generations on average to find a perfect match for the third infrequent pattern. **(b)** The number of individuals in the population producing antibodies that match more than 1/2 of each pattern. There is an initial decline in the number of organisms focusing on the least frequent pattern, but the count climbs again as the resource levels reach equilibrium. The initial small number of individuals at all matching the infrequent pattern explains why it takes a longer time to evolve a perfect match.

In the initial population the same number of individuals are focusing on each pattern due to random chance, but the distribution of resources causes more to focus on the most frequent pattern, while the number focusing on the least frequent pattern declines. The more abundant resource associated with the first pattern compared to the third pattern gives

those individuals matching it a selective advantage. However, as fewer individuals attempt to acquire the resources associated with the third pattern, these resource levels accumulate; more resource is flowing in each generation than flows out or is being used by the population. If the population focuses on only the first two patterns, the resource associated with matching the third pattern will grow to a sufficient amount where an individual that only partly matches it will receive more resource than another who perfectly matches either of the other two patterns. This dynamic will lead the population of individuals matching the third pattern to expand. As more individuals focus on the third pattern a perfect match will be found, and the population will reach a steady state with perfect matches for each pattern.

5.4.3 Overlapping Niches

All of the experiments discussed thus far have used three antigens or three patterns contained in a set of antigens that represent distinct fitness peaks; that is, the niches overlap minimally. Next we tested the behavior of the limited resource environment given niches with more overlap, by creating a set of four antigens represented by the following bit strings:

111111000000000000000000
000000111111000000000000
000000000000111111000000
000000000000000000111111

Clearly if only one antibody could exist the best generalist would be a string of all 0's, matching 75% of each of the antigens. However the match score function we use to determine fitness is designed to encourage specialists, resulting in a generalist of all 0's and any single specialist perfectly matching one antigen receiving the same total fitness. A specialist antibody that perfectly matches one of the above strings would match all 24 bits of the corresponding antigen, but only 12 bits of each of the other three. Its fitness would

therefore be

$$2^1 * 2^0 * 2^0 * 2^0 = 2 \tag{5.5}$$

across the four antigens (the fitness function rewards individuals only if they match more than 1/2 the bits of a given antigen). A generalist antibody represented by a string of all 0's would match 18 bits of each of the four antigens, giving it a fitness of

$$2^{.25} * 2^{.25} * 2^{.25} * 2^{.25} = 2 \tag{5.6}$$

Therefore we expect each of the four specialists and the generalist of all 0's to be equally competitive in the unlimited resource environment. Our expectation for the limited resource environment is based on Tilman's theory on competition for limited resources in natural ecologies. Assuming there are no other interactions affecting fitness dynamics in the population, we can still expect populations to be able to stably support only four species (in this case distinct phenotypes), as there are only four limited resources in the environment, one associated with each antigen. Theoretically this calculation means it would be impossible for the four specialists and the generalist to coexist. In fact, we would expect that a population can support either the four specialists in a stable state of equal proportions, or the generalist alone. We assume that these two stable outcomes will occur with an equal likelihood.

Figure 5.6 shows the results of this test in both the unlimited and the limited resource environments. In the unlimited resource environment we get the expected result that the population can support only one species and in some replicates it converges to a specialist, but in most the generalist takes over. However in the limited resource environment we find that the generalist dominates the population in all 10 replicates.

Why is the generalist so successful? There are several possible contributing factors, but the primary reason comes from the fact that the solutions are using sexual recombination,

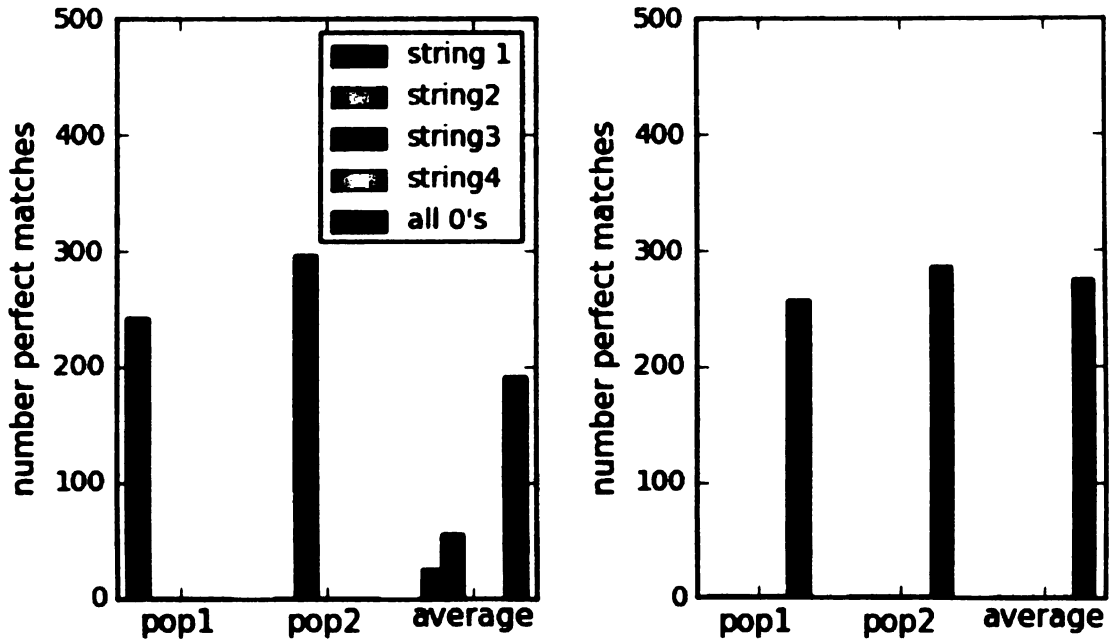


Figure 5.6: Overlapping Niches. Number of antibodies produced that perfectly match each of four antigens and one generalist represented by string of all 0's. Data are recorded after 500 generations of evolution in populations of 500 individuals. (a) Two typical populations and the average of 10 populations in the unlimited resource environment. (b) Two typical populations and the average of 10 populations using an environment with four limited resources, one associated with each antigen.

and one of the products of the crossover of any two specialists in this task is the generalist antibody of all 0's. Mutation-selection balance will therefore cause the specialists to produce the generalist, but the reverse is not true. When we perform these same experiments using the limited resource environment, but without any form of crossover, we find that 6 out of 20 replicates reach a stable state with all four specialists and no generalists.

Why does any imbalance remain? The generalist does have a second slight advantage. Specifically, once the population reaches a state with either the four specialists or the single generalist, this state is stable and unlikely to be disturbed; we found in experiments that invasion by a rare type (or types in the case of the four specialists) was uncommon. This result indicates that whichever state evolves to dominate the population first will likely remain dominant. The generalist antibody matches 12 bits of each of the antigens and has the ability to exploit any one of the four resources that is available at a higher level than the others. Therefore, for the specialists to dominate the population, they must *all* evolve at least to the point of matching 12 of the bits of their respective antigen before the generalist evolves. We found that if we start populations with equal proportions of each of the five genomes (the four specialists and the generalist), 16/20 populations reach a steady state with the four specialists and just four are swept by the generalist. Given that we started populations with 20% generalists, it is expected given a random walk that the generalist would dominate the population 20% of the time, which is exactly what we see here.

The fact that the limited resource environment encourages the generalist to evolve is, in part, a bi-product of the fitness function we used and the fact that the generalist and each specialist are assigned the same fitness, given equal levels of all resources. We can easily change the fitness function to bias selection in one direction of the other. We explore this dynamic by modifying the fitness function such that an individual's match factor is cubed instead of squared (see Equation (5.1)), leading to a greater increase in resources acquired for each extra bit matched.

$$S(x, v_i) = \begin{cases} 0 & \text{if } b_i < l/2 \\ (\frac{2b_i}{l} - 1)^3 & \text{otherwise} \end{cases} \tag{5.7}$$

The fitness of the specialist will still be

$$2^1 * 2^0 * 2^0 * 2^0 = 2 \tag{5.8}$$

as before, but now the fitness of the generalist will b

$$2^{.125} * 2^{.125} * 2^{.125} * 2^{.125} = 1.41 \tag{5.9}$$

and the specialists will have a selective advantage given equal levels of each resource.

Figure 5.7 shows the resulting populations after 500 generations of evolution. The four antigens are now perfectly matched in equal proportions by the population of antibodies. We also find that the generalist antibody represented by the string of all 0's is still maintained in the population at a low frequency, however this is not stable, but due only to the fact that it is constantly being generated by crossover between specialists. If we switch to an asexual population the generalist disappears completely.

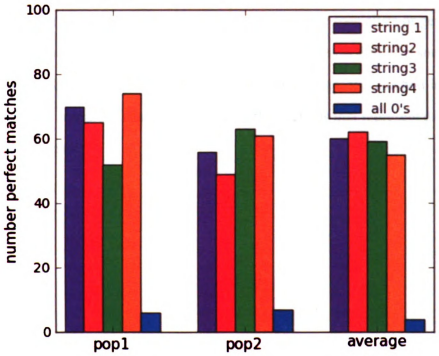


Figure 5.7: Selection for specialists. Number of antibodies produced that are generalists (represented by all 0's) four specialists that perfectly match each of four antigens. Data recorded after 500 generations of evolution in populations of 500 individuals. The bar chart depicts two typical populations and the average of 10 populations using a modified fitness function designed to add selection pressure for specialists.

5.5 Conclusion

This initial work has shown that natural competition for limited resources enabled populations to find and cover multiple niches in a bit-string matching problem that is a simple analogy to the human immune system. Populations were also able to find effective generalists by exploiting patterns in the set of antigens to be matched when there weren't enough antibodies to support perfect matching on each antigen. Varying the strength of selection for specialists in the fitness function allows us to select for specialists or generalists when niches are highly overlapping. The dynamics of competition for limited resources as a niching mechanism become more complex when niches overlap, this method is more clearly applicable when resources can be associated with functions that represent distinct peaks in the fitness landscape.

Chapter 6

A General Ecology-Based Evolutionary Algorithm and its Application to a Real-World Problem

In Chapter 4 we showed that the natural mechanism of competition for limited resources can increase the diversity of an evolving population and accelerate the evolution of solutions to complex problems, In Chapter 5 we applied these concepts to an evolutionary algorithm focused on a simple diagnostic problem and demonstrated that it could discover and maintain a diverse set of solutions. Here, we introduce the general form of an ecology-based evolutionary algorithm (Eco-EA) that uses this mechanism, and apply it to a real-world problem in software engineering.

6.1 Eco-EA

Competition for limited resources can be implemented in any evolutionary algorithm to increase diversity. The Eco-EA requires that for each trait to be evolved an associated limited resource must be created and governed by the parameters described in Chapter 3.

When an individual manifests a trait associated with a resource, it receives a fraction of the currently available resource, and its fitness is increased by a proportional amount. This resource sharing method differs from existing sharing methods in two ways:

1. The resources are directly tied to phenotypic functionality, as opposed to using a genetic or phenotypic distance measure. Functionality based resources are simpler because you do not have to determine a distance metric for sharing. Additionally, the algorithm can find and cover niches both close together and far apart in the same fitness landscape. The disadvantage is that you must known of what traits should be rewarded, but most real world problems have this property, at least in terms of low-level of building blocks.

2. The researcher can control how much resource is given to different functions (by setting the inflow rate higher or lower), such that if some domain knowledge is known it can be easily exploited. As I explore in Chapter 6, subtasks can be assigned less resources than larger pieces of a problem and the availability of resources can be made to vary both temporally and spatially.

6.2 Evolving Behavioral UML Models

I implemented the Eco-EA in Avida, and applied it to a real-world application of evolving UML models for controlling nodes in a remote sensor network. I modified Avida in a similar fashion as in Chapter 5; individuals are evaluated using an explicit fitness function to determine the speed at which they can produce offspring. However, unlike Chapter 5, individual genomes are not binary strings, but rather are sequences of special-purpose instructions that build a UML model, and I do not use crossover during replication. I used the same resource settings as in Chapter 5 for the limited resources environment, an inflow of 100 units of resource per update (where the gestation time of organisms is fixed such that 1 update $\sim$ 1 generation), and a maximum consumption value of 5.

6.2.1 Avida-MDE

For this study, I used a software engineering extension to Avida called Avida-MDE (Avida for Model-Driven Engineering), previously developed by Goldsby and Cheng (October 2008). Here I 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.* (2007) have proposed a modeling and analysis process for such high-assurance systems, where a system is represented by a class diagram that captures the structural elements and several behavioral models (Goldsby & Cheng, July 2008). 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. 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. We implemented a new genetic language 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 the previous papers by Goldsby & Cheng (2008). 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. We added several tasks to the Avida environment, which have previously been linked only to constant rewards. *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. Finally, *property tasks* are included to reward models for having all possible execution paths support a desired system property. For example, "no data are ever lost," "battery levels never drop below a threshold value," or "water level never exceeds a maximum value." Therefore, no matter what the system is performing, these properties are always maintained.

6.2.2 Grid-Stix

Avida-MDE was previously used to generate behavioral models for GridStix, a light-weight flood warning system that comprises a set of sensor nodes. GridStix is used to monitor the

water levels for potential flood conditions with the River Ribble in England (Hughes, 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, collecting data, and notifying nearby nodes.

Different scenario tasks captured different non-functional tradeoffs. Specifically, three tasks rewarded models that supported scenarios for setting different CPU 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 only 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 CPU 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 CPU 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 rewarded a model only 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, we required the models to support a minimum set of scenarios before we considered them 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 CPU speed and one transmission method), as well as three other required scenarios. These combinations of the three CPU speed scenarios and six 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 phenotypes.

6.2.3 Generating a Diverse Suite of Models with Different Non-functional Properties

Our first objective was to assess how well the Eco-EA version of Avida-MDE performs compared to the original, unlimited-resource 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 able to evolve only 14 of the 18 possible phenotypes, even across 40 runs.

We compared the efficacy of the Eco-EA version of Avida-MDE in evolving a diverse suite of models to Goldsby and Cheng's previous results. The key difference between the two approaches is how the NF scenarios are rewarded. In both versions of Avida-MDE, organisms can receive a fitness gain for only one scenario from each of the sets of NF scenarios (in the Grid-Stix study, one CPU speed and one transmission method). 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. The rest of the Avida-MDE tasks (including the required scenarios) are rewarded using the original fixed-amount method in the Eco-EA version as well; these tasks represent properties and behavior required in all models and therefore we do not want the fitness gained by performing them to be dependent on other organisms in the population.

We performed two 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. In future discussions we will refer to these two sets of experiments, as opposed to the original Avida-MDE experiments. We ran each experiment for 100,000 updates, or 24 hours, whichever came 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. In this pair of experiments all of the 20 Eco-EA

replicates evolved to satisfy the property task and reached the 24 hour limit, ending between 7,000 and 98,000 updates. Sixteen of the unlimited resource EA replicates reached the 24 hour limit, ending between 47,000 and 92,000 updates, and the other four ended at the 100,000 update cut-off.

We found that the Eco-EA version of Avida-MDE not only generated a more diverse suite of final model phenotypes, but that it also evolved any model satisfying the required functional property faster than the traditional, unlimited resource approach. The Eco-EA populations first found models satisfying the property at an average of 2,106 updates, and across all 20 populations found property-satisfying models of each of the 18 non-functional phenotypes within 4,000 updates of evolution. In contrast, the traditional approach using unlimited resources first evolved a model satisfying the property at an average of 3,747 updates, and even after 100,000 updates of evolution of all 20 populations had not found property-satisfying models representing all of the possible phenotypes ($p < .01$ that the first update a model evolved is significantly different between the two environments using a Mann-Whitney U test).

The Eco-EA version of Avida-MDE also yielded a significantly more diverse set of models in each individual population than the unlimited resource EA. Figure 6.1 shows the average number of unique phenotypes of models satisfying the required property found in 20 Avida populations evolved in each environment over time. As discussed earlier, many populations reached the 24 hour time limit before evolving for 100,000 updates so the average assumes each population maintains its final number of phenotypes through the end of 100,000 updates.

Every one of the 20 Eco-EA populations yielded property-satisfying models. The final populations contained coexisting models representing between eight and the full set of 18 different phenotypes, with a mean of 16.1 phenotypes per population. In contrast, only 16 of the 20 unlimited resource Avida-MDE populations evolved any property-satisfying models, with a maximum of 12 phenotypes in a single population. The average number of

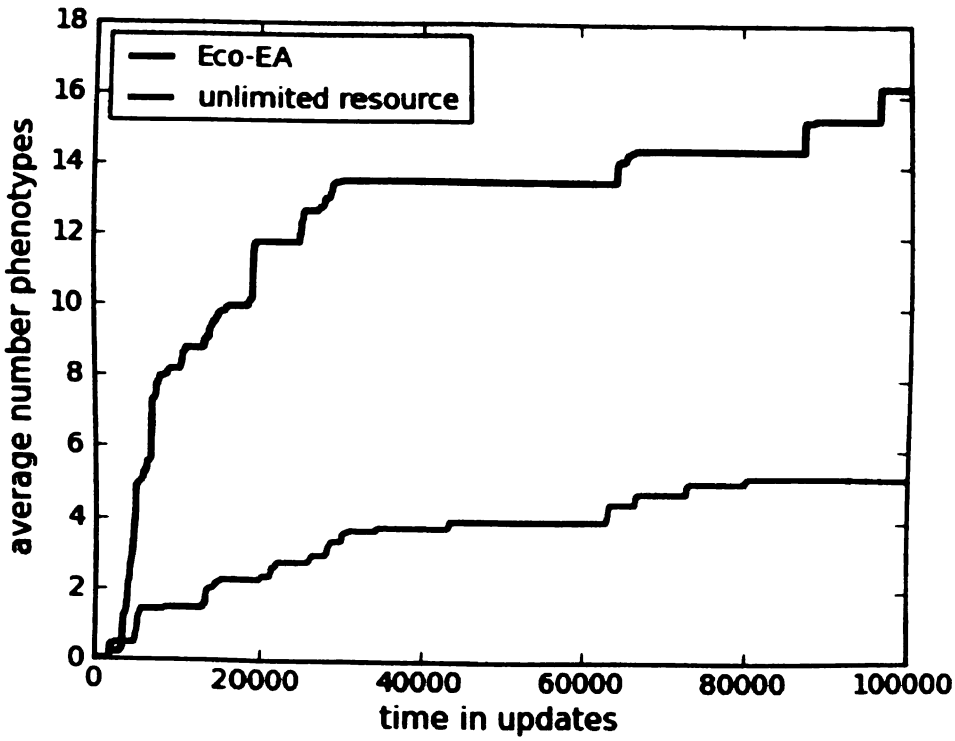


Figure 6.1: Phenotypic diversity of models satisfying the property. The average number of unique phenotypes for models satisfying the property found by each of 20 populations per environment over time. In the Grid-Stix problem there are 18 possible combinations of non-functional scenarios, each of which results in different non-functional behavior in the models. In the Eco-EA (limited resources environment), property-satisfying models representing most of the 18 non-functional phenotypic possibilities evolved in every population. In the traditional EA (unlimited resource environment), models satisfying the property evolved more slowly and fewer of the non-functional based phenotypes are found at all.

phenotypes found in the final populations of the unlimited resource EA was 5.2 ($p < .001$ comparing 5.2, s=3.7 to 16.1, 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.

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 two sets, a single CPU speed and a single transmission method, as well as the three other required scenarios. We performed five replicates in 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 rewarding for only a single phenotype, any model satisfying the required behavioral property appeared in only 5 of the 90 populations within 100,000 updates of evolution.

The theory that more scenarios yields more evolutionary pathways leading to faster evolution could also explain why the Eco-EA finds models satisfying the developer's requirements faster than the unlimited resource EA. Figure 6.2 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. The Eco-EA populations diversify quickly to con-

tain individuals supporting almost all combinations of NF scenarios, while the unlimited resource populations are stuck on just a few of the possible phenotypes, giving evolution fewer possible paths to a model satisfying the property.

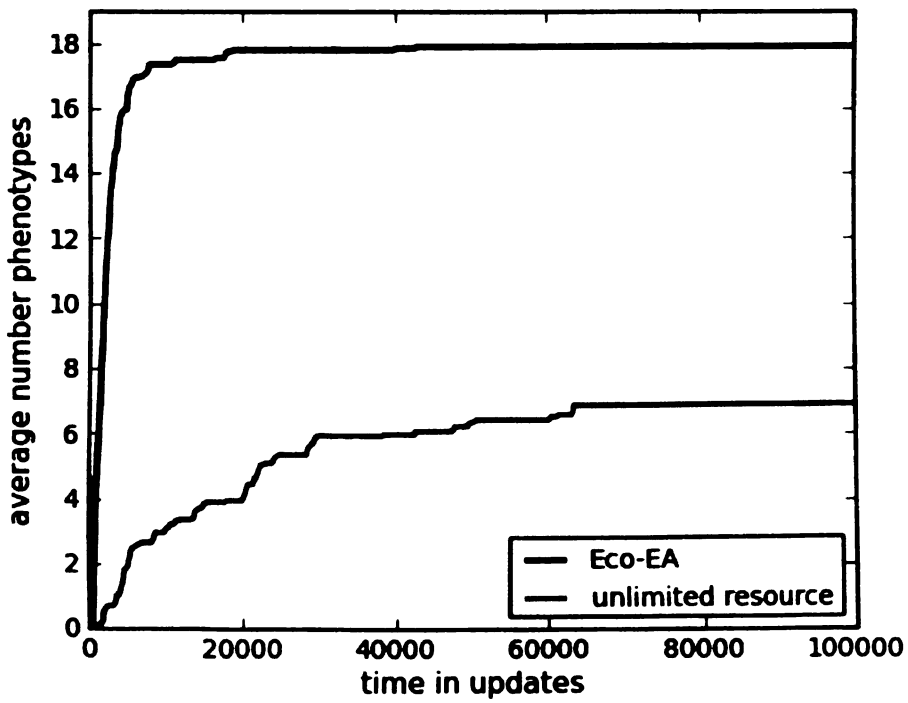


Figure 6.2: Phenotypic diversity of models supporting the required scenarios. The average number of unique phenotypes in each population for models supporting the required scenarios. Eco-EA populations diversify to cover almost all of the possible phenotypes prior to evolving models that satisfy the property, while the unlimited resource EA finds only a few phenotypes per population. These data suggest that there are fewer evolutionary paths to find a model satisfying the property in the unlimited resource EA.

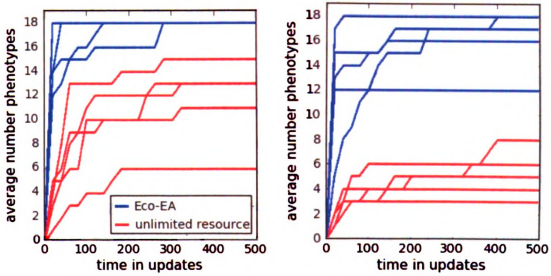
6.2.4 Adaptability of Models

Developers often design an initial model suited to a given set of conditions as a first step in building a full suite of models appropriate for a wide range of condition domains. We therefore performed additional comparisons between the Eco-EA and the unlimited resource EA, where the population was initially filled with copies of one individual that builds a

model already satisfying the required behavior. We randomly selected five 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 unlimited resource Avida-MDE, ensuring that we chose the same five phenotypes as the former set. We used each of the 10 chosen models to seed the initial populations of 20 replicate experiments where we continued evolution in the Eco-EA environment, and 20 more where we continued evolution in the unlimited resource environment, for a total of 400 additional runs.

We found two key results: 1) the Eco-EA environment generates a more diverse suite of models, in less time than in the original unlimited resource environment; and 2) the individuals evolved in the Eco-EA environment appear to be more evolvable themselves than those evolved in the unlimited resource environment. Figure 6.3 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 unlimited resource EA tends to evolve only phenotypes close in solution 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 generated all 18 possible phenotypes when seeded with any of the 5 models initially evolved using the Eco-EA, and the unlimited resource EA generated over 10 phenotypes when seeded with four of these models, while the most it ever found when seeded with models initially evolved in the unlimited resource environment was 8 phenotypes.



(a) Eco-EA Evolved Seed

(b) Unlimited Resource Evolved Seed

Figure 6.3: Phenotypic diversity of models when seed population with property-satisfying model. 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 five models originally evolved in the Eco-EA environment. While individual models yielded varying results, the Eco-EA quickly evolved all 18 phenotypes no matter which ancestor it was seeded with. The unlimited resource environment was never able to find all 18 phenotypes. (B) Similar results occurred when populations were seeded with models originally evolved in the unlimited resource environment. The Eco-EA generated all 18 phenotypes for only two of these initial models, but still generated more phenotypes in its worst case (12) than the unlimited resource EA generated in its best case (8). Each experiment let a population of 1,000 individuals evolve for 24 hours, ranging from 1,000 to 7,000 updates in Avida time (400 to 2,800 generations).

The increased evolvability of models initially evolved in the Eco-EA version of Avida-MDE can be seen more clearly in Figure 6.4, where the average results across all five seed models are shown for each of the four treatments. We again found that the Eco-EA version of Avida-MDE not only evolved a more diverse set of phenotypes more quickly than the unlimited resource approach across sets of all 20 runs, but it also yielded higher diversity in individual populations. When averaging all populations across all 10 seed models, the Eco-EA evolved an average of 17.1 phenotypes per population, while the unlimited resource EA evolved 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 population diversity also differed based on which environment was used to evolve the seed models. Averaging all final populations from both environments when seeded with the 5 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 unlimited resource environment ($p < .001$ comparing 14.8, s=1.7 to 10.7, s=2.2 using the independent group t-test for means).

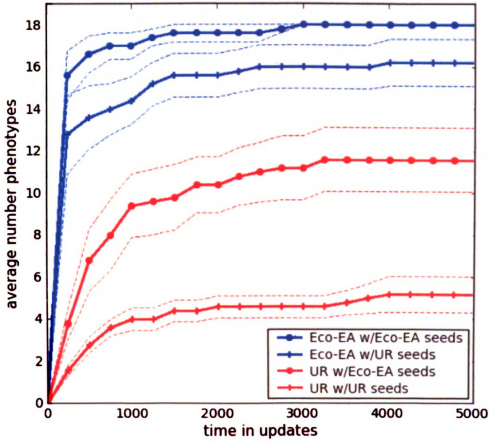


Figure 6.4: Average phenotypic diversity of models when seed population with property-satisfying model. Average of data with error bars (+/- 1 standard error) for each of four experimental treatments (All combinations of two types of seed models; those evolved in the Eco-EA environment or those evolved in the unlimited resource (UR) environment, and two environments for continued evolution; the Eco-EA and the unlimited resource). The line for each treatment represents the mean of the number of unique phenotypes found for each of five seed models. Each seed model was used to start 20 populations, and the number of unique phenotypes was counted across all 20 of those populations over time. The Eco-EA found, on average, a more diverse set of models across 20 populations than the unlimited resource EA no matter which type of models it was seeded with ($p < .001$, t-test). Both environments found a significantly more diverse set of models when seeded with models initially evolved using the Eco-EA than those evolved using the unlimited resource EA ($p < .001$, t-test).

6.3 Conclusion

We have shown that the Eco-EA yields several advantages over a traditional unlimited resource EA when applied to a real-world complex problem, including:

1. faster evolution of any solutions
2. evolution of a more diverse array of solutions
3. evolution of solutions with greater plasticity 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.

Chapter 7

Conclusion

In this final chapter I summarize my investigations in using the natural mechanism of competition for limited resources to increase both diversity and the rate of adaptive evolution in an evolutionary algorithm. I also explore future directions of research and potential applications of the Eco-EA to solve problems of greater complexity.

7.1 Summary

My major goal for this thesis was to explore the benefits of incorporating competition for limited resources into an evolutionary algorithm framework. I focused on how this mechanism can increase diversity to provide many evolutionary paths to a problem solution, an aspect of niching methods that has not been extensively studied in previous literature. I introduced "Eco-EA", a general form of an evolutionary algorithm that associates a limited resource with each desired trait or subtask to be evolved, and I showed that it yielded several advantages over traditional EA approaches, including: (1) significantly more rapid evolution of targeted complex functions; (2) discovery and maintenance of a diverse set of partial solutions that together solve a problem; (3) maintenance of a selection of high-quality final solutions for the researcher to choose from, often with slightly different properties; and (4) discovery of more evolvable solutions that can easily adapt to new environments.

In Chapter 3 I presented the specific implementation of competition for limited resources in the digital life system Avida, and studied the effects on population dynamics of the parameters that are used to control the limited resources. I showed that competition for limited resources is effective at increasing diversity over a broad range of these parameters, and that the general appropriate parameter settings are governed by a relatively simple set of equations.

7.1.1 Rapid Evolution of Targeted Complex Traits

In Chapter 4 I explored how competition for limited resources can lead to the faster evolution of a complex trait in Avida. I compared the performance of Avida with limited resources to a previous study on evolving complex functions using Avida with an unlimited resource. The study focused on evolving one complex task, EQU, and how simpler tasks are used as building blocks during its evolution. I found that associating a limited resource with each subtask as well as EQU itself led to more populations performing EQU at the end of a period of evolution. Some of this improvement was caused by the reward structure inherent to the limited resource environment; individuals could more readily trade off many subtasks in order to gain EQU and still receive an overall fitness increase. However we also showed that EQU actually appeared more often and earlier in populations evolved in the limited resource environment, regardless of whether it was maintained. We posit that the most likely reason for this increased appearance is that the higher phenotypic diversity caused by competition for limited resources creates more potential paths evolution can follow to produce the complex trait EQU.

7.1.2 Discovery and Maintenance of Diverse Problem Solutions

Much of the previous literature on sharing techniques in evolutionary algorithms is focused not on evolving a single solution to a complex problem, but rather on obtaining a final population of diverse solutions. These solutions could represent partial solutions that together

represent the overall solution to a problem, or could all be full solutions to a problem, but that have different properties. In Chapter 4 I explored how using limited resources in a simple EA can discover and maintain diverse solutions. I found that when niches were distinct with little or no overlap (i.e. an individual that performs one task well cannot perform any others well), populations evolved in a limited resources environment were able to find and cover multiple niches in a simple bit-string matching problem. Populations were also able to find good generalists by exploiting patterns found in large sets of tasks when it was impractical to maintain individuals perfectly performing each task. Highly overlapping niches tended to cause the populations to converge to generalists unless strong selection for specialists was introduced. I concluded that while this direction could be explored further, in general limited resources as a niching method is more clearly applicable when resources can be associated with functions that represent distinct peaks in the fitness landscape.

7.1.3 Applications to a Real-World Problem

The true test of any method designed to improve the performance of an evolutionary algorithm is to apply it to an actual problem in the real world. In Chapter 5 I applied the Eco-EA to a real-world application of evolving UML models for controlling nodes in a remote sensor network. I found that the Eco-EA did yield the two advantages discussed in Chapters 3 and 4; it more quickly found any final solution, and it discovered and maintained a more diverse array of solutions with different properties for the developer to choose from. I also found that the Eco-EA evolved solutions with greater plasticity that were easily adapted to succeed in different environments.

7.2 Future Research

The promise and hope of evolutionary algorithms is to evolve solutions to complex problems beyond what a human engineer could produce. However current evolutionary algo-

rithms have mostly fallen short of that goal; the complexity of problems they can solve is generally limited. We hope competition for limited resources will increase the complexity of problems that can be approached with an EA. The ideal Eco-EA would allow an engineer to use the domain knowledge they have without over-limiting the trajectory of evolution. In many cases an engineer may not know exactly how to solve a problem, but may have general ideas for potential substeps on the way to a full solution. Using the Eco-EA, each substep could be associated with a limited resource; if an idea turns out to be beneficial the part of the population targeting that resource will gain the benefit, however if an idea turns out to lead down a bad path toward an evolutionary dead-end, only part of the population will follow that path while the rest will still be exploring alternative directions to reach the final solution. Here I present some initial exploration on this concept of throwing out ideas and associating limited resources with each, using a diagnostic problem to show the potential benefits.

7.2.1 Multiple Pathways

We investigate the conditions that allow the ecology-based evolutionary algorithm to concurrently explore many diverse paths to a final solution in one population. We use a simple diagnostic problem to explore the dynamics of the system in a clear, tractable environment. The problem is to evolve a bit string where an individual is rewarded for how many consecutive 1's appear starting at the beginning of the string that are immediately followed by 0101. So the string 11110101... receives a bonus of 4, while the string 1011111110101... receives a reward of 0 (the 1's must be consecutive from the beginning of the string). Any sequence of zeros and ones can follow the 0101 without loss of fitness.

This problem is trivial, but presents a challenge for traditional EA populations because an organism that evolves a string with the pattern 0101 following even a small number of 1's will often sweep the population. At this point the population can get stuck on this local fitness peak, as multiple mutations are required to move the 0101 further down the string

and any single mutation is deleterious. Therefore an engineer trying to solve this problem might think of different subtasks that would be useful for organisms to achieve along the path to evolving the overall solution. One sub-task would simply be to evolve as many 1's as possible at the beginning of the bit string. This selective pressure seems advantageous as a string of all 1's requires just 2 mutations to receive the full bonus of 28. However, to test the theory that the limited resource system could withstand poor decisions by the engineer we also consider a subtask rewarding the organisms for evolving as many 0's in the string as possible.

We used a simplified version of Avida similar to the one used in chapter 4 to compare the limited resources environment to an unlimited resource environment. Each organism represented a binary sequence of ones and zeros, and we used an explicit fitness function to evaluate their performance on each task in the environment. These organisms had a fixed length of 32, so the maximum bonus for the overall task was 28 (gained by a string of 28 1's followed by 0101). The maximum bonus for the other two tasks was 32 (a string of 32 1's or 32 0's). The population was initially filled with 500 organisms representing random bit strings. In these experiments organisms reproduced asexually, without crossover. The parameters for the limited resources environment were set to be the same as those in the experiments in Chapter 4 (inflow $I = 100$, outflow $O = .01$, maximum consumption $m = 5$, consumption fraction $Cf = .0025$). The actual reward an individual received for each task was a multiplier to its execution speed determined by the following equations and the bonus described above for each task:

$$B = \frac{bonus}{maxbonus}^2 \tag{7.1}$$

$$A = min(B * C_f * R, m) \text{ if using limited resources} \tag{7.2}$$

$$A = B * m \text{ if using unlimited resources} \tag{7.3}$$

$$\text{Reward} = 2^A \tag{7.4}$$

We compared three different subtask structures in each of two environments (limited resources and unlimited resource) to an unlimited resource environment where neither of the subtasks were rewarded. All seven of the treatments reward the overall task of matching the desired pattern. Table 7.1 shows the three subtask structures that we tested.

Label	Tasks rewarded
Good subtask	Overall + maximize number 1's in string
Bad subtask	Overall + maximize number 0's in string
Both subtasks	Overall + maximize 0's + maximize 1's

Table 7.1: Three subtask treatments. The first rewards for the overall task and the subtask we expect to improve evolution of the overall task, the second rewards for the overall task and the subtask we expect to hinder evolution of the overall task, and the third rewards the overall task and both subtasks.

Figure 7.1 shows the number of populations out of 50 that evolved a perfect match to the overall pattern at each time point. The unlimited resource treatment that rewards only for the overall task finds a perfect match in 23 populations. Adding just the beneficial sub-

task to the unlimited resource environment improves this result to 34 populations, however if both of the potential subtasks are added only 22 populations find a perfect match, and if only the harmful subtask is added just two populations find a perfect match. The unlimited resource environment can maintain only one phenotype, and it is more likely that an individual in the initial population will have many 0's than that it will have many consecutive 1's followed by a specific four bit pattern. Thus many populations are swept by the higher-fitness individuals with many 0's when this subtask is rewarded, and the desired overall task is never obtained.

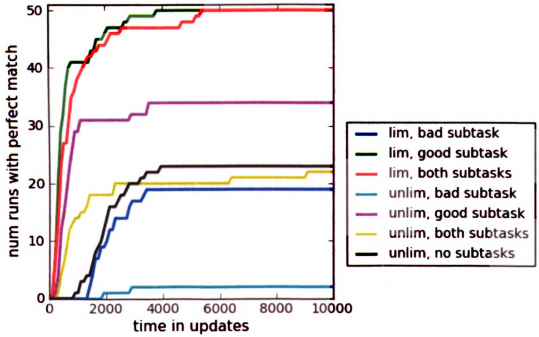


Figure 7.1: Effects of different subtasks on the evolution of the overall pattern. The number of populations out of 20 that have evolved a perfect match to the overall pattern at a given time. Seven different treatments are shown, one that rewards only the overall task, and three in each resource environment that reward the overall task and combinations of the two possible subtasks. The legend labels for each treatment show which resource environment is used, "lim" for limited resources and "unlim" for unlimited resource; labels for which tasks are rewarded are described in Table 7.1.

7.2.2 Conclusion

In contrast the limited resources environment evolves a perfect match to the overall pattern in all 50 populations in the two treatments where the beneficial subtask is rewarded, regardless of whether the harmful subtask is also rewarded, performing even better than the unlimited resource environment with the beneficial subtask only $p < .001$ 50 is significantly different from 34, Fisher's Exact test). Subpopulations in this environment are targeting all possible resources, and thus though one of those subpopulations is led away from the target pattern by the harmful subtask, the rest of the population is unaffected. Even when only the harmful subtask is rewarded and not the beneficial, the limited resources populations appear to do no worse than the single task unlimited resource populations, and much better than when only the harmful subtask is rewarded in the unlimited resource environment ($p < .001$ 19 is significantly greater than 2, Fisher's Exact test). Again, we see this result because only part of the population is led astray by the harmful subtask.

I have shown that in this simple problem the Eco-EA can improve the evolution of the final solution by allowing an engineer to throw any ideas for substeps to that solution into the environment, and that by associating each with a limited resource we can gain the benefit of good ideas and mitigate the negative effects of bad ideas. I plan to test this ability on the real-world software engineering problem discussed in Chapter 6. In that problem we used only scenarios that helped lead an individual to the desired overall property, however one could easily imagine a developer including a scenario that leads an individual away from obtaining the property, either through faulty logic in designing the scenario or simply error in inputting the scenario into the algorithm. I posit that the Eco-EA will not be overly affected by a few bad scenarios as the population will be exploring many paths at once, but that a traditional EA may suffer a serious drop in performance as many whole populations will target a bad scenario.

BIBLIOGRAPHY

Adami C, Ofria C, and Collier TC (2000). Evolution of Biological Complexity. *Proc. Natl. Acad. Sci.* USA 97, 4463-4468.

Armstrong, R. A. & McGehee, R. (1980). Competitive exclusion. *Am. Nat.* 115,151–170.

Bäck, T. (1996). Evolutionary Algorithms in Theory and Practice: Evolution Strategies. *Evolutionary Programming*, Genetic Algorithms, Oxford Univ. Press.

Chow, S.S., Wilke, C.O., Ofria, C., Lenski, R.E., Adami, C. (2004). Adaptive Radiation from Resource Competition in Digital Organisms. *Science* vol. 305, 84-86.

Cooper, T.F. and Ofria, C. (2002). Evolution of stable ecosystems in populations of Digital Organisms. *proc. Artificial Life VIII*, 227-232.

Darwen, P. and Yao, X. (1996). Every niching method has its niche: Fitness sharing and implicit sharing compared. *Parallel Problem Solving from Nature* 398-407.

Darwen, P. J. (1996). Co-evolutionary Learning by Automatic Modularisation with Speciation. PhD Thesis. University of New South Wales.

De Jong, K.A. (1975). An Analysis of the behavior of a class of genetic adaptive systems. PhD thesis, Dept. Computer Science, University of Michigan.

Fogel, D. (1998). Evolutionary Computation: The Fossil Record. (eds). IEEE Press.

Fonseca, C. M. & Fleming, P. J. (1995). An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation* 3(1), 1-16.

Forrest, S., Javornik, B., Smith, R.E., and Perelson, A.S. (1993). Using genetic algorithms to explore pattern recognition in the immune system. *Evolutionary Computation*, 1(3) 191-211.

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

Goings, S. and Ofria, C. (2009). Ecological Approaches to Diversity Maintenance in Evolutionary Algorithms. *IEEE Symposium on Artificial Life*, Nashville, TN.

Goldberg, D.E., Deb, K. and Horn, J. (1992). Massive Multimodality, Deception and Genetic Algorithms. *Parallel Problem Solving from Nature* 2, 37-46.

Goldberg, D. (1989). Genetic Algorithms in Search, Optimization and Machine Learning. *Addison-Wesley*, Reading Massachusetts.

Goldberg, D.E. and Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimization, *Proc. 2nd Int. Conf. Genetic Algorithms*, 41–49.

Goldsby, H.J., and Cheng, B.H.C. (2008). Automatically Generating Behavioral Models of Adaptive Systems to Address Uncertainty. *Proc of the ACM/IEEE Int. Conf. on Model Driven Engineering Languages and Systems*, Toulouse, France.

Goldsby, H.J. and Cheng, B.H.C. (2008). Avida-MDE: A Digital Evolution Approach to Generating Models of Adaptive System Behavior. *Proc. of the Genetic and Evolutionary Computation Conference*, Atlanta, Georgia.

Hastings, A. Disturbance, coexistence, history, and competition for space. *Theor. Popul. Biol.*18, 363–373.

Hector, A. *et al.* (1999). Plant diversity and productivity experiments in European grasslands. *Science* 286, 1123–1127 (1999).

Holland, J.H. (1975). Adaptation in natural and artificial systems. First edition.University of Michigan Press. Reprint MIT Press, 1992.

Horn, J. (2002). Resource-Based Fitness Sharing. *Proc. of the 7th Int. Conf. on Parallel Problem Solving From Nature*. Lecture Notes In Computer Science, 2439, 381-390.

Horn, J. (1997). The Nature of Niching: Genetic Algorithms and the Evolution of Optimal, Cooperative Populations, PhD thesis, Dept. Computer Science, University of Illinois.

Horn, J., Goldberg, D.E., and Deb, K.(1994). Implicit niching in a learning classifier system: Nature's way. *Evolutionary Computation*, 2(1), 37–66.

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

Huisman, J. & Weissing, F. J. (1999). Biodiversity of phytoplankton by species oscillations and chaos. *Nature* 402, 407–410.

Hu, J., Goodman, E. D. (2002). Hierarchical Fair Competition Model for Parallel Evolutionary Algorithms. *Proc. of the Genetic and Evolutionary Computation Conference*, GECCO-2002, New York, 772-779.

Konrad, S., Goldsby, H.J., Cheng, B.H.C. (2007). i2MAP: An Incremental and Iterative Modeling and Analysis Process'. *Proc. of ACM/IEEE Int. Conference Model-Driven Engineering Languages and Systems,* 451-466.

Lenski, R. E., Ofria, C., Pennock, R. T., & Adami, C. (2003). The evolutionary origin of complex features. Nature, 423, 129-144.

Mahfoud, (1995). Niching Methods for Genetic Algorithms, PhD thesis, Dept. Computer Science, University of Illinois.

Mahfoud, S. W. (1992). Crowding and Preselection Revisited. *Parallel Problem Solving From Nature,* 2, 27-36.

McCann (1980). The Diversity-Stability Debate, *Nature* 405:228-233 15.

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

Oppacher, F. & Wineberg, M. (1999). The Shifting Balance Genetic Algorithm: Improving the GA in a Dynamic Environment. *Proc. of the Genetic and Evolutionary Computation Conference*, 1, 504–510.

Potter, M.A. and DeJong, K.A. (2000). Cooperative Coevolution: An Architecture for Evolving Coadapted Subcomponents. *Evolutionary Computation*, 8(1), 1–29.

Schluter D. (2001). Ecology and the origin of species. *Trends Ecol. Evol.* 16, 372-379.

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

Shimodaira, H. (1999). A Diversity Control Oriented Genetic Algorithm (DCGA): Development and Experimental Results. *Proc. of the Genetic and Evolutionary Computation Conference*, 1, 603–611.

Smith, R.E., Forrest, S., Perelson, A.S. (1993). Searching for Diverse, Cooperative Populations with Genetic Algorithms. *Evolutionary Computation,* 1(2), 127–149.

Spears, W.M. (1994). Simple Subpopulation Schemes. *Proc. of the Third Annual Conference on Evolutionary Programming.*

Tilman, D. (2000). Causes, consequences and ethics of biodiversity. *Nature* 405, 208-211.

Tilman, D. (1982). Resource Competition and Community Structure. Princeton University Press, Princeton, N.J

Tilman, D., Wedin, D. & Knops, J. (1996). Productivity and sustainability influenced by biodiversity in grassland ecosystems. *Nature* 379, 718–720.

Tilman, D., Knops, J., Wedin, D., Reich, P., Ritchie, M. & Siemann, E (1997). The influence of functional diversity and composition on ecosystem processes. *Science* 277, 1300–1302.

Tilman, D. & Kareiva, P. (1997). Spatial Ecology: the Role of Space in Population Dynamics and Interspecific Interactions. Princeton Univ. Press, Princeton.

Tsutsui, S., Fujimoto, Y., & Ghosh, A. (1997). Forking Genetic Algorithms: GAs with Search Space Division Schemes. *Evolutionary Computation*, 5, 61–80.

Ursem, R. K. (2002). Diversity-Guided Evolutionary Algorithms. *Proc. of Parallel Problem Solving from Nature VII*, 462-471.

Xie, T. & Chen, H. (2001). Problem Decomposition-Based Scalable Macro-Evolutionary Algorithms. *Proc. of the 2001 Congress on Evolutionary Computation* CEC2001, 223-231