The Nature of Niching:,
Genetic Algorithms and the Evolution of Optimal, Cooperative Populations

Jeffrey Horn

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801

IlliGAL Report No. 97008
October 1997

(internet email) jeffhorn@uiuc.edu
(phone) 217/333-2346
(fax) 217/244-5705
(www) http://GAL4.GE.UIUC.EDU/illigal.home.html

Illinois Genetic Algorithms Laboratory (IlliGAL)
Department of General Engineering
University of Illinois at Urbana-Champaign
117 Transportation Building
104 South Mathews Avenue
Urbana, IL 61801-2996

THE NATURE OF NICHING: GENETIC ALGORITHMS AND THE EVOLUTION OF OPTIMAL, COOPERATIVE POPULATIONS

BY

JEFFREY HORN

A.B., Cornell University, 1985
M.S., University of Illinois at Urbana-Champaign, 1995

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1997

Urbana, Illinois

Abstract

Genetic algorithms (GAs) with *fitness sharing* have been analyzed and successfully applied to problems in search and optimization, while GAs using various types of *resource sharing* have been incorporated into classifiers, immune system models, artificial ecologies, artificial economies, etc. Both types of sharing are based on the same observation of nature: dividing a finite resource among competing organisms limits the size of populations dependent on that resource. If multiple resources are involved, each resource can be considered a *niche*, and each subpopulation exploiting a niche can be considered a *species*. By treating population slots as a finite resource, we obtain the traditional, fixed-size population *simple GA*. By further treating the figure of merit (i.e., the function to be optimized) as a limited resource, we obtain fitness sharing. If our problem domain manifests explicit resources, such as rewards for correct classification of examples, we can enforce resource sharing.

Both fitness and resource sharing have been studied separately, and usually under the assumption of non-overlapping niches, or *perfect sharing*. Little effort has been made to understand the more complicated situation of niche overlap. Yet the resolution of niche overlap is critical to successful, useful niching. Non-overlapping niches can be covered by non-competing species, while heavily overlapped niches induce competition between species in which only the best will survive to represent the overlapping resources. Somewhere in between the two extremes of overlap must lie a critical boundary between cooperation (all species survive) and competition (one dominant species survives). We seek this boundary for the simplest case of niche overlap: two niches.

We first define the general mechanism of sharing, and investigate where sharing-induced-niching fits into the larger frameworks of context-dependent function optimization, natural ecologies, and models of cooperation and competition. We then map fitness sharing to resource sharing, and show that they are identical when there is no niche overlap (i.e., perfect sharing). We go on to analyze the three cases: perfect sharing, fitness sharing (with overlap) and resource sharing (with overlap). We find that even under severe selective pressure with high degrees of niche overlap, a stable equilibrium population is quickly found and indefinitely maintained, a population consisting of diverse species covering the best niches. We find that maintenance of the sharing equilibrium degrades gradually as niche overlap and fitness discrepancies increase. We create a control map for the two-niche case, plotting the boundary between surviving niche pairs (cooperation) and untenable niche pairs (competition). Controlling this boundary will allow us to use sharing to evolve the types of cooperation and competition appropriate to the problem at hand.

We extend the above analysis by looking at some aspects of the general case of multiple overlapping niches. We discover that calculating the equilibrium point for three or more niches under resource sharing can be computationally expensive, as evidenced by the difficulty that GA selection has in reaching it. We present some evidence that equilibrium can thus represent solutions to hard problems, and that selection plus sharing might be a computationally intensive yet efficient algorithm (even without the exploration operators of recombination and mutation).

In addition to analyzing the existence, resilience, nature, and meaning of sharing equilibrium, we also explore its utility. We apply various sharing methods to specific problems from three domains: function optimization (search), multi-objective optimization, and layout/packing problems. We find that sharing is a robust, flexible, and extensible technique, whose simplicity and similarity with nature give us further confidence in its general applicability. We hope our results ameliorate some of the over-reaching criticisms frequently used to dismiss sharing as a niching method, by demonstrating that sharing is not computationally expensive, that its performance degrades gracefully with decreasing a priori knowledge of niche distribution, and that simple and intuitive techniques can be found to extend sharing's domain of utility.

For my family: Gabriele, Thorin, and Ellissa.

Acknowledgments

David E. Goldberg was a constant source of insight and guidance on most of the topics in this paper. I thank him for his roles of advisor, supporter, and mentor through the most difficult periods of my PhD quest. It was his suggestion long ago that niching (and in particular sharing) form a foundation for *cooperative behavior*, which has always been my main interest. Thus he channeled my vague desires into a productive conduit for research.

I thank Georges Harik and Dirk Thierens for helpful discussions on population optimization and mixing times, respectively. I thank Kalyanmoy Deb, who inspired me daily with his optimism, energy, discipline, and analytical skills. I will miss the friendship of everyone at the Illinois GA Laboratory (IlliGAL), especially Hillol Kargupta, Sam Mahfoud, Erick Cantu-Paz, Brad Miller, and Fernando Lobo. With all of its students, papers, international visitors, librarians, and computing power, the IlliGAL is a great place to study GAs, especially for a beginner.

I acknowledge the critical input of Deborah Thurston, on the topics of decision analysis and MAUA. I also acknowledge the help of Wayland Eheart and his students Sanmugavadivel Ranjithan, Pascal Stork, and Scott Cieniawski at UIUC's Department of Civil Engineering in implementing their ground water monitoring problem. Angus R. Simpson (University of Adelaide, Australia) graciously provided his extensive code for the NYC tunnels problem, and spent much time and energy explaining the problem to me during his visit to the IlliGAL in 1993. Mary A. Johnson, Department of Mechanical and Industrial Engineering at UIUC, pointed me toward the work on quasi-stationary Markov chains. Good friend and IlliGAL member-for-life Nicholas Nafpliotis was my spirited collaborator on all of the niched Pareto GA work. My wife, Gabriele Burkhard, provided a professional yet personal editorial analysis and proofreading that improved this manuscript "just in time".

On a deeper level I thank my original family: siblings Lisa, Brian, and Jennifer and parents Anna and Ralph, for their encouragement and support. When I chose to leave a decent consulting job and my native culture on the mid-Atlantic seaboard, they let me know that I wasn't leaving my family behind!

On a more practical level, I gratefully acknowledge support provided by the U.S. National Aeronautics and Space Administration (NASA) under Contract NGT-50873, administered by the Johnson Space Center as part of the NASA Graduate Student Researchers Program. I also received support from the U.S. Army under Contract DASG60-90-C-0153, and from the U.S. Air Force Office of Scientific Research (AFOSR) under Grants number F49620-94-1-0103, F49620-95-1-0338, and F49620-97-1-0050[1].

[1]The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the AFOSR or the U.S. Government.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

This thesis is about the evolution of cooperation. Since evolution implies competition, this cooperation must emerge from competition. Indeed it is based on competition for survival and is defined in the context of strong selective pressure. Most implementations of the *genetic algorithm* (GA) do not incorporate explicit cooperation of any type, but rather converge quickly to a uniform or near-uniform population of a single high quality solution. It seems difficult in general to improve upon the *simple GA* by introducing cooperation precisely because of the difficulty of maintaining a cooperative group of individuals while selection is trying to choose only the best from among them.

A rather simple form of cooperation stands out among others as being particularly robust, particularly useful in a wide range of applications, and particularly successful in those applications. This form of cooperation is the localization of competition around finite, limited resources (*niches*), resulting in the lack of competition *between* such areas, and causing the formation of *species* for each niche. The localization of competition is introduced by simply *sharing* resources among the individuals competing for it. Individuals not competing for the same set of resources can be seen as cooperating, in a rather implicit fashion. We argue that the subtle, indirect nature of such cooperation , which leads us to call it 'implicit", is the very reason for considering it *foundational*. It seems to be a "level-0" type of cooperation, necessary for all other "higher" levels of cooperation. It is this basic cooperation that allows complementary specialists to coexist and diverse ecosystems to thrive.

The use of sharing to promote niching and speciation in the GA is based on a natural metaphor. But successful niching in nature provides more than motivation. What happens in nature can be and should be a guiding principle in the design and application of GA sharing methods. In this thesis a strong emphasis is placed on resource sharing in nature, and how abstractions of such natural procedures work in artificial GAs. In particular, we expend considerable effort relating the extremely successful and well-known implementation of *fitness sharing* back to its "roots" in the more natural *resource sharing*. That is, in pure function optimization no obvious resources exist except the function value (fitness) itself, and so this is shared. While fitness sharing has enjoyed success, it must be remembered that it is part of a more general and very natural technique of sharing which can be applied with equal success to a range of problems outside of pure function optimization, including classification, multi-objective decision making, layout problems, immune system algorithms, artificial ecosystems, economic modeling, etc.

The goal of this thesis is to tie together the work on sharing, both fitness sharing and all types of resource sharing, and to create a more unified view of the sharing process across domains. We explore the very nature of niching (that is, the creation and maintenance of distinct species of similar individuals), from the point of view of *niching equilibrium*. We define and measure the unique diverse population distribution that is dynamically stable under the GA's selection pressure. We ask hard questions about this equilibrium: does it really exist?, how stable is it (how long does it last)?, how long does it take to reach it?, does it represent true cooperation?, can it represent a solution to a problem?, what can it be used for?, etc. Along the way to answering some of these questions we discover some surprising results, such as phase transitions between cooperation and competition, the complexity of three-species interactions, or the "long-distance" communication of long chains of cooperative individuals.

But all of these interactions, phenomena, and analyses are based on an algorithm as simple as the simple GA: the division (sharing) of finite resources. The basic axioms of sharing are based on an ecological metaphor:

1. equal division of resources (among all equally deserving competitors), and

2. conservation of finite resources (the total of all resource dividends is constant).

Thus the sharing methods discussed here can be seen as a most effective and universal instance of what has been called *computational ecology* or *ecological computation* (Hubermann, 1988).

In the remainder of this chapter, we present more specific background on natural algorithms such as genetic algorithms and niching, then summarize the goals and contributions of the dissertation. Lastly we present the overall organization and structure of the remainder of this thesis.

1.1 Background: Algorithms

In this section we place genetic algorithms in the class of *natural algorithms*, emphasizing the inspirations from nature, and the importance and risk of relying on natural metaphors.

1.1.1 Natural Algorithms

Genetic algorithms, along with neural networks, simulated annealing, and others, are members of a fairly young class of algorithms in computer science: *natural adaptive algorithms*. Such algorithms are attempts to abstract essential methods presumably "discovered", or "designed", by natural evolution, and to apply them to difficult, open problems, such as optimization, classification, or artificial intelligence. While simulated annealing emulates the physical process of annealing metallic crystals, neural networks and genetic algorithms are biologically inspired. Artificial neural networks are based on biological neural networks found in creatures with advanced nervous systems. Genetic algorithms (GAs) on the other hand are abstractions of the general process of biological evolution.

1.1.2 Genetic Algorithms

Genetic algorithms (GA) have steadily gained respect and rigor as robust, stochastic algorithms for search, optimization, and machine learning (Goldberg, 1989; 1994). This popularity is based on a growing history of solving difficult real-world engineering and design problems, but also on substantial gains in theoretical understanding of the algorithm. The "simple GA" represents a successful abstraction of an essential algorithm from nature, combining three key ingredients: (1) high selective pressure to promote superior solutions, (2) high rates of "mixing" (recombination) to juxtapose the various components of good solutions to form better solutions, and (3) the use of fairly large populations to absorb the stochastic effects of the other two mechanisms (Goldberg, Deb, & Thierens, 1993; Thierens & Goldberg, 1993).

1.1.3 Niched Genetic Algorithms

Another mechanism to abstract from nature and formalize as an algorithm is the natural formation of species corresponding to niches. In nature, niching involves the formation of distinct species exploiting different *niches* (resources) in the environment. In an artificial system like the GA, niching means the formation of subpopulations in a fixed size population, with each subpopulation specializing at a subtask of the problem at hand (e.g., optimizing a subfunction of the overall fitness function, or covering a subset of the examples to be classified).

Not only is niching another example of a natural algorithm, but it is also a strong candidate for a fourth ingredient to add to the simple GA because it is relatively simple, efficient, and generally applicable. It can promote *cooperative* populations that work together to solve a problem. Whereas the simple GA is purely competitive, with the best individuals quickly taking over the population, *niched GAs* converge to a population of diverse species (niches) that together "cover" a set of resources or rewards.

1.1.4 A Niching Scenario

To better convey the simplicity and naturalness of sharing, we summarize the situation in nature that are we are trying to abstract and simulate:

Imagine k distinct species competing for the same finite resource r (e.g., food, water, or sunlight). Furthermore, assume that all species are evenly matched in this particular competition. How should the resource be allocated? It seems natural to divide the resource evenly among all k competitors. Thus each receives an $\frac{r}{k}$ size piece of the resource[1]. Now imagine a set of m different resources, $R = \{r_0, r_1, r_2, ..., r_{m-1}\}$. Species that "cover" a resource r_i for which little competition exists receive a relatively large "payoff". Assume that reproduction is carried out in proportion to individual fitness, where fitness is the sum an individual's payoffs from the sharing of resources. Individuals with a greater than average total of payoffs are thus allocated more offspring than are individuals with lower totals. The population is thus encouraged to speciate (specialize) so as to evenly cover the available resources.

[1] Or, if the resource is not divisible into k portions, the entire quantity of the resource could be given to one of the competitors at random. But over many such competitive trials, say t trials, we expect each species to garner an equal amount, $\frac{t*r}{k}$, of the total resource $t * r$.

This simple act of *sharing* resources induces a remarkably useful and powerful population behavior: *niching*. A *niche* can be defined generally as a subset of resources in the environment. A *species* on the other hand can be defined as a type or class of individuals that takes advantage of a particular niche. Thus niches are divisions of environment, while species are divisions of the population (and of the space of all possible individuals). When the terms *niche* and *species* are used interchangeably, it is assumed that there is a one-to-one correspondence between them. For example, species **A** is said to "cover" niche **A** (conversely, niche **A** is said to be exploited by species **A**).

Niching, as induced by the simple act of sharing resources (payoff), seems to be a "natural algorithm", and, as part of another natural algorithm, the *genetic algorithm*, it is the subject of this thesis. We explore the power and limitations of niching, and demonstrate the practicality of this simple algorithm over a broad range of problems, from the domains of search, function optimization, multi-objective optimization, design, and machine learning.

1.2 Purpose and Scope

The title of this thesis captures the major emphases of the contained work. We stress the *nature* of niching in a genetic algorithm, mapping the *sharing algorithms* used in artificial genetic search to extant models of niching and speciation from the field of theoretical ecology. We argue that the same niching mechanisms work in nature and in niched GAs. We also show that under certain preference orderings, the steady-state equilibrium population under niching can be considered "better" than all other possible populations (and therefore *optimal*). Third, we emphasize that the *sharing* method of localized competition for local resources (fitness) leads to strong cooperative relationships, even though the mechanism of cooperation is indirect. Finally, we address the *evolution* of such cooperative groups. Even without the exploration induced by recombination or mutation, the niched GA under pure selection is still searching the space of possible population distributions, looking for the best, most cooperative population made up of individuals from its current population.

The overall objective of this work is to explore the power and complexity of the relatively simple act of sharing finite resources. The current high level of interest in the "evolution of cooperation" is usually focused on more complicated and explicit interactions among organisms, such as mutual defense, or exchange of resources or information between individuals. But a surprising degree of sophistication can be found in the emergent relationships induced by simply sharing resources. And yet despite the dynamical complexity of niched GAs, the niching mechanism is simple enough to be understood and its effects modeled and predicted. This comprehension allows us to harness the natural algorithm of niching and immediately apply it to practical problems.

Below we present the more specific goals and contributions of this work.

1.2.1 Goals

This thesis takes on some methodological goals as well as the more concrete goals of analysis and application.

1.2.1.1 Methodology: Abstracting and Designing Natural Algorithms

On the methodological level, we view the analysis, modeling, and design of niching as an example of the relatively new practice of abstracting essential techniques from nature and incorporating them into our repertoire of effective, predictable computer algorithms[2]. Although far from a precise and definitive methodology, the general path taken here in modeling natural niching and designing artificial niching contains some general lessons in what works and what doesn't. These lessons concern the use of tools such as asymptotic Markov chain analysis, expected population distribution tracking, facet-wise modeling, control maps, and critical timing analysis. We attempt to summarize such pointers near the end of this dissertation.

1.2.1.2 Concrete Goals: Niching as a Natural Algorithm

The primary goals of this work are logically ordered:

1. demonstrate that **niching** is beneficial in general,
2. show that **sharing** is a natural algorithm to implement niching, and
3. begin constructing a bounding, prescriptive **model** of niching.

[2]As opposed to taking physical *designs* from nature, a more or less successful human strategy dating from prehistory.

The first goal is accomplished via analysis and applications. In the analysis of niching we show that a "sharing equilibrium" exists, and that it is quickly reached and quickly restored, and is therefore stable over a long period of time. We include some preliminary results showing that at such equilibrium the niched GA can be expected to maintain schemata in direct proportion to schema average fitness, along the lines of Holland's *schema theorem*. We conjecture that this "fitness proportionate schema maintenance" might avoid the race between selection and mixing (recombination), and thereby allow sufficient time for proper juxtaposition of building blocks under selective pressure. In the application section we discuss the empirical results of our own carefully controlled experiments that indicate improved search and optimization through niching.

The second goal of this thesis is to demonstrate sharing as a natural way of inducing the specialization of niching. By careful analysis of *implicit niching* (via resource sharing), we show how the natural and intuitive approach of simply dividing up scarce resources, such as credit or reward, among competing individuals leads to powerful and robust niching. We map implicit niching, and resource sharing, to *explicit niching* via *fitness sharing*, and discuss how the former inspired the latter. The applications section provides empirical evidence of niching's success.

The third goal is to make substantial progress in the development of a more or less complete model of niching. We limit most of our analyses to the two-niche case, but within that restriction our results are general to all possible combinations of niche fitnesses and niche overlaps. We attempt to define a control map for niching, plotting the boundary between cooperation (i.e., niching success) and competition (i.e., niching failure). And we outline ways of generalizing the analyses and their results to more general cases of multiple (> 2) niches.

1.2.2 Contributions

In meeting the general goals described above, the dissertation makes some concrete contributions to our understanding of niching in particular and emergent cooperation in general:

1.2.2.1 Stability of Steady-State Populations

There exists a *steady-state* (SS) or "equilibrium" population that is unique and stable under niching and selection. For a simple GA, the only equilibria are uniform populations, but with sharing the equilibrium distributions in general are *not* uniform ones, but rather consist of diverse sets of high quality (i.e., high fitness) individuals. By stable, we mean that the combination of selection and sharing induces a "restorative pressure" that drives the population to the equilibrium distribution, in both the *convergence* from the initial (e.g., random) population distribution, and also in the quick restoration of equilibrium after any deviation caused by the stochastic selection operator.

The stability of this steady-state equilibrium means that cooperation can be maintained under severe selection pressure. This implies that cooperative groups can be sought via intensive genetic search (i.e., high selection pressure along with a significant rate of recombination). This is directly applicable, for example, to learning classifier systems, allowing us to apply *search-intensive concept induction* (Giordana & Saitta, 1996).

1.2.2.2 A Predictive Model of Niching

Niching performance can be boundedly predicted. That is, given some idea of the initial population distribution, we can predict the expected steady-state distribution over any given time period. This predictive ability in turn allows us to identify and set a few key niching parameters to probabilistically guarantee maintenance of desired niches. In particular, we identify the niche fitnesses (specifically the ratio of fitnesses) and the degree of niche overlap as the two independent variables of a niching control map. This control map divides up the space of possible niching situations (i.e., combinations of niche overlap and fitness) into cooperative and completive regions, along the boundary between niching success and niching failure.

1.2.2.3 Complex Relationships Under Niching and Selection Alone

Subtle and complex cooperative and competitive relationships among individuals are possible under niching and selection alone. These include multiple levels of nested cooperation and competition, and dynamic problem decomposition via teams of "specialists".

1.2.2.4 Exploration under Niching and Selection Alone

Complex, evolutionary *learning* is taking place even under selection alone. That is, even without recombinative search taking place, the combination of selection and niching (via sharing) explores a complex space of population distributions, in which the steady-state equilibrium is *not* easily and quickly found.

1.2.2.5 Diverse Populations Can Represent (Distributed) Solutions

It is possible for the steady-state population distribution represents an distributed representation of a solution, or "answer", to the problem at hand. We present a number of problems in which high quality (if not optimal) solution can be extracted from the equilibrium distribution (rather than from a single individual).

1.2.2.6 Niching Helps Search

Even without an explicit need to maintain a diverse population, niching can be beneficial. In the optimization of a single scalar objective, niching can prevent premature convergence, effectively side-stepping the race between selection and recombination (Thierens, 1995). We give some theoretical support to the notion that niching can maintain desirable schemata (i.e., building blocks) beyond their expected extinction times due to drift. We also run several carefully controlled experiments on artificial and open problems, giving empirical evidence of the more robust nature of niched GA search.

1.2.2.7 Optimal Populations

The idea of evolution, whether artificial or natural, as an optimizer of organisms is controversial enough (De Jong, 1992; 1993), but the proposition that evolution (at least selection) somehow optimizes a *population* of organisms might be considered radical. Yet if we could model evolution as a population optimizer, we might be able to predict population space trajectories, and basins of attractions for locally or globally optimal populations. Under certain assumptions, we show that *proportionate niching* optimizes a simple multiplicative function (as opposed to an additive function) of the population's fitnesses. In addition, the steady-state population distribution by definition can be considered to be an *evolutionarily stable strategy* (ESS) (Smith, 1982), and therefore an optimal *mixed strategy.*

1.2.2.8 Niching is Broadly Applicable

Niching is readily specialized to a number of very diverse problem domains, guided by our predictive model of niching and by the metaphor of theoretical ecology . In many cases, the niched GA approach represents the first and only attempt to solve the problem with a coop-

Overall, Hierarchical Organization of Dissertation

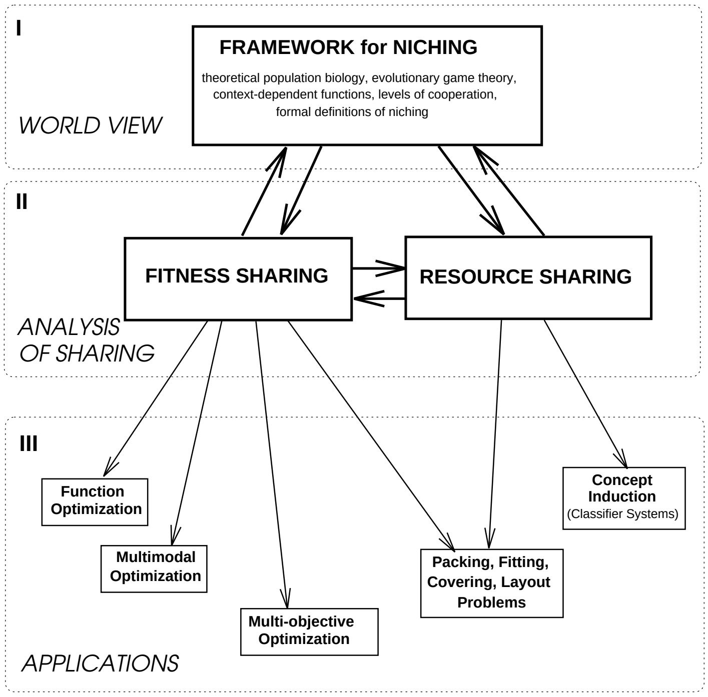

Figure 1.1: Three levels of discussion.

erative population (as a distributed representation of the solution). This dissertation covers example applications in multimodal function optimization, search (for function optimization), classification, multi-objective optimization, and layout/packing problems.

1.3 Organization

The topics of this dissertation fall into three levels of discussion,

- **I – Frameworks (Abstract Level)**
- **II – Models of Sharing (Analytical Level)**
- **III – Applications (Practical Level)**

as shown in Figure 1.1.

At the highest, abstract level, *I. Frameworks*, we strive to place GA niching in the larger contexts of existing work from other disciplines (e.g., theoretical ecology, agent-based artificial intelligence), arguing for niching as a natural and ubiquitous process. At level II, the analytical level, we present a detailed analysis of niching, and propose predictive models. At level III, the practical level, we summarize, suggest, and demonstrate practical applications of niched GAs to several problem domains.

As Figure 1.1 indicates, these levels of discussion are interrelated. For example, we discuss ecological framework issues in the analysis and modeling chapters, and we refer back to both the framework and the model when designing and applying niching algorithms at the practical level (III). Still, the linear organization of the chapters mostly follows the levels of discussion down the hierarchy, with chapter 3 presenting the framework (level I), chapters 4, 5, 6, and 7 covering the analysis and modeling of niching, and chapters 6, 7, and 8 containing application write-ups.

Below we present a more detailed summary of the remaining chapters.

1.3.1 Chapter 2: Background

The second chapter provides general background on genetic algorithms and niching, and the analysis of both. An overview of the simple GA is given first, followed by a review of niching in GAs in general. The niching overview covers the need for niching, and reviews the various major alternative niching methods that have been developed in the literature. A detailed look at one method of niching, *sharing*, comes next. The discussion of sharing includes both fitness sharing, for multimodal optimization, and resource sharing, for *implicit niching* in algorithms such as classifier systems, immune system models, and artificial life. Complete specifications of both kinds of sharing algorithms are given.

1.3.2 Chapter 3: Frameworks for GA Niching

In chapter 3 we place GA niching in some larger contexts. For example theoretical ecology includes several models of species interactions which can tell us a great deal when we map GA niching into them. We also offer some models of our own, such as a layered model of cooperation and competition, and a hierarchy of context-dependent (population-based) functions. Together these different views of niching demonstrate the critical role of niching in any model or method of cooperation. Niching appears to be foundational to higher levels of more direct cooperation. And niching is shown to exhibit many of the complex interactive phenomena usually expected only from more sophisticated and direct types of cooperation. Such niching phenomena include specialization, commensal symbiosis, mutualistic symbiosis, Lotka-Voltera oscillations, long-term evolutionary stability, and nested competition and cooperation.

Next in this chapter we map fitness sharing to resource sharing (*explicit niching* to *implicit niching*. We show that the fitness sharing is truly an attempt, as Goldberg and Richardson (1987) stated, to apply the more obvious and natural *resource sharing* to the domain of function optimization, in which there are no explicit resources to be shared. This mapping allows us to discuss later the general aspects of sharing, common to both fitness and resource sharing, and to note the differences, where the niching models diverge for the two types of sharing.

Finally, we discuss resource sharing in the learning classifier system as a specific, concrete example of resource sharing.

1.3.3 Chapter 4: The Existence and Maintenance of an Equilibrium

Chapter 4 looks at equilibrium in the two-niche case. Two species are minimally sufficient for exhibiting important niching phenomena, such as dynamic equilibrium states, and the transition from cooperation to competition as niche overlap is increased. Under the two-niche assumption, we examine the existence and nature of niching equilibrium states, in which a diverse population is held in some kind of dynamic steady-state. Building a full Markov chain model, we discover the limitations of asymptotic analysis in modeling niching, and propose a more useful model of long-term transient states, such as the equilibrium state of niching. We use these models to predict niche loss times as population size and niche overlap are varied, finally deriving a closed-form expression approximating niche maintenance/loss times. We then seek the *phase transition* from "cooperation" to competition, as niche overlap is increased from none to complete overlap.

1.3.4 Chapter 5: Convergence to Niching Equilibrium

Next we analyze niche restoration (or niche convergence) times, which is the time needed to converge to the steady-state. We turn to a different modeling technique, expected population distribution analysis, to derive a closed-form expression for niche convergence times (under varying population size and degree of overlap). The full Markov chain models are used to test the approximations. We examine how to use the convergence times together with the niche maintenance times derived in chapter 4 to produce fully predictive models of niching success and failure. The two critical niching time models, niche maintenance and niche convergence, can be integrated to generate a control map for the two-niche case, providing bounds on the combinations of population size, fitness ratio, and niche overlap for which we can say niching will fail or that niching will work.

1.3.5 Chapter 6: The Meaning of Niching Equilibrium

In this chapter we pause to consider the implications of niching equilibrium. Chapters 4 and 5 convince us that niching equilibrium is real, but is it meaningful? We consider the question of whether niching equilibrium represents some optimal population distribution; that is, whether or not a niched GA is optimizing some function of the entire population. We also look at the complexity of computing the equilibrium for multiple niches under resource sharing, and ask whether the niched GA is performing some useful computation simply by converging to equilibrium. Finally, we begin to investigate the apparent benefit that sharing provides to simple GA search. We take a first look at what happens to schemata (building blocks) under niching.

1.3.6 Chapter 7: Further Analyses of Niching

This chapter includes some ideas on extending the two-niche results of chapters 4, 5, and 6 to the general $k > 2$ niche case. After this brief foray into the intricacies of the k-niche case, a more formal approach to modeling k-niches is suggested and partially explored. First we show how the general k-niche situation can be decomposed readily into separate, independent clusters of interacting (overlapping) niches. Next we propose two types of clusters that we suggest as bounding cases: a *chain* of niches, and a *clump* of niches. Finally, we explore the use of the two-niche results to model and bound the behavior of the two "bounding" cases of niche clusters.

Next, the issue of the computational cost of sharing is addressed, and we answer some recent concerns that sharing, at $O(N^2)$ is an inefficient way to evolve cooperation. We also attempt to answer some recent criticisms of the limitations of sharing by demonstrating the flexibility and robustness of sharing on a hard, massively multimodal problem, introducing some new sharing techniques along the way.

1.3.7 Chapter 8: Applications

In this chapter we present several applications of fitness sharing to difficult and open problems. The application domains include multimodal function optimization, multi-objective optimization, and layout/packing problems. Several of the insights and other results from earlier chapters are illuminated and applied here.

1.3.8 Chapter 9: Conclusions

The dissertation closes with a summary of the major contributions, including (1) new insights about niching (e.g., limitations), (2) the utility of the proposed models (with guidelines for their application), and (3) suggestions for major new avenues of research (adding more direct cooperative interactions on top of a firm foundation of robust niching.

Chapter 2

Background

This section contains a brief introduction to genetic algorithms, niching in general, fitness sharing and resource sharing as mechanisms for niching, and to the learning classifier system (LCS) as an example of resource sharing.

2.1 Genetic Algorithms

In this section we briefly examine the nature and basic operation of the simple GA. The reader is referred to (Goldberg, 1989a) for an in-depth introduction to GAs and a thorough grounding in basic GA theory.

Throughout the history of GAs, since the 1975 publication of the first edition of John H. Holland's book (Holland, 1992), these algorithms have been applied to combinatorial optimization problems[1]. Such problems are defined by a vector of discrete decision variables $\vec{v}$ and a scalar objective function $f(\vec{v}) \rightarrow \Re$ to be optimized (i.e., minimized or maximized). The decision variables $\vec{v}$ are typically mapped to (encoded as) a bit string s consisting of ℓ binary variables (bits). The objective function $f(\vec{v})$ is usually scaled so as to be non-negative, perhaps inverted (if originally a minimization problem) to ensure a *direct* relationship between the objective function value and the desirability of the solution, and then used as the GA's *fitness function*. Thus each possible solution s is a particular setting of the decision variables

[1]Including continuous numerical optimization problems that have been mapped into combinatorial problems via discretization of the continuous decision variables.

corresponding to a particular string s which has a corresponding fitness $f(s)$ to be maximized or minimized.

One major application of GAs is to "black-box" function optimization. We assume a function $f(\vec{x}) \rightarrow \Re$ mapping a vector of decision variable settings to a real-valued (scalar) objective value (figure of merit). This objective function is to be optimized (minimized or maximized), but we know little or nothing else about f. To apply the GA to the problem, we first develop an encoding of the decision variables as a (generally binary) string (chromosome). The fixed length (ℓ-bit) binary encoding means that any of the 2^ℓ possible chromosomes (bit settings) represents a candidate solution to the problem, and can be assigned a *fitness* (usually some direct function of the objective function f). The GA maintains a fixed size N population of chromosomes, to which it applies standard, basic GA operators such as selection, crossover, and mutation.

2.1.1 GA Operators

The basic, well-known operators of the simple GA include *selection* (e.g., deterministic binary tournament selection), *crossover* (e.g., two-point), and *mutation* (e.g., bit-wise). More complicated, specialized and advanced operators include *niching* (e.g., fitness sharing) and *reordering* (e.g., inversion). I briefly describe examples of each of the three basic operators below.

Binary tournament selection operates on a population of individuals by selecting two individuals from the current population at random and comparing their fitnesses in a *binary tournament*. The individual with higher fitness is copied into the new population (i.e., the next generation). This process is repeated N times to create a new generation.

Two-point crossover operates on a pair of individuals (parents), call them **A** and **B**, replacing them by a pair of their offspring, call these **a** and **b**. The offspring are created by selectively copying bits of each parent into the offspring as follows. Two different *cutting points* are chosen at random from among the ℓ possible cutting points (between bit positions). The bits from parent **A** that are between the two chosen crossover points are copied over the corresponding bits from parent **B**, thus creating child **a**. Child **b** is created by the complementary process of copying the bits between the crossover points from parent **B** over the corresponding bits in parent **A**. The two children then replace the two parents in the population. Crossover happens with a probability p_c. Thus with probably $1 - p_c$ the crossover will not take place and the two children will be exact copies of their parents.

Bit-wise mutation acts on a single individual, by flipping each of the ℓ bits independently with a probability p_m. This probability is usually kept very low, to minimize disruption of converged genes and to reduce the "randomness" and noise of search. Again in order to focus on the processing of building blocks, we assume no mutation ($p_m = 0$) in all of the GA experiments.

2.1.2 The Main Loop

We assume a *generational* GA, in which the entire current population (generation) of N individuals is replaced by a new population (next generation) created by applying the three operators described above to the current population. The initial population (generation 0) consists of N randomly generated individuals. Each individual is then evaluated for fitness, as explained above. With fitnesses assigned, the three operators can be applied to create the next generation. Following (Goldberg, 1989a), we hold $N/2$ pairs of binary tournaments. Each pair of tournaments thus selects two individuals. We apply two-point crossover to these two parents, and (optionally) apply bit-wise mutation to each of the resulting offspring. The resulting (possibly mutated) offspring are then placed in the new population. The sequential application of these three operators is repeated for a total of $N/2$ (assuming even N) times, generating a new population of size N (generation 1). The new population replaces the old population, and we repeat the process on the new population. Thus, the population of generation $t + 1$ replaces the population of generation t.

This loop is repeated until some stopping criterion is reached, such as finding an individual with fitness within some small range δ of the ideal, or until the population has converged to near or complete uniformity (i.e., it is composed entirely, or nearly so, of copies of a single individual/solution).

2.2 GAs with Niching

In this section we introduce the idea of *niching*, motivating our discussion of various niching methods by first suggesting why niching is useful in evolutionary computation. We only briefly mention alternative niching methods (several of these are covered in more detail by Mahfoud

(1995a; 1995b)), focusing more on sharing techniques for niching. In particular, we introduce fitness sharing and resource sharing, the main targets of our analysis in this dissertation.

2.2.1 The Need For Niching

In a GA, selection drives the evolving population toward a uniform distribution of N copies of the most highly fit individual. Mutation and non-stationary fitness functions might stave off 100% convergence, but it is unarguable that the first-order effect of the first-order operator, selection, is the loss of low quality diversity. In many applications of the GA, including classifier systems, uniform convergence is undesirable. In multi-objective GA problems, we might want to find a number of solutions with different tradeoffs among the multiple objectives (Horn & Nafpliotis, 1993). Even with single objective (scalar fitness function) GAs, we might want to avoid premature convergence, or discover alternative "runner-up" solutions, by maintaining high quality diversity (Goldberg & Richardson, 1987). In the LCS, we ask the GA to search through the space of all possible rules to find and maintain a diverse, cooperative subpopulation.

To prevent the best individual in the population from replacing all copies of competing rivals, some kind of *niching* (a.k.a. *speciation*) is necessary. Niching induces *restorative pressure* (Horn, 1993), to balance the *convergence pressure* of selection.

Alternatively, some have argued, we could decrease selection pressure so that the population doesn't converge in the time frame (number of generations) of interest or simply can never overcome the noise of disruptive operators like mutation. But reducing selective pressure also reduces the speed and efficiency of stochastic search algorithms such as the GA. With insufficient selection, we get too much exploration and not enough exploitation. It has been found that a combination of high selective pressure and high rates of "mixing" (i.e., recombination, for exploration) works well (Goldberg, Deb, & Thierens, 1993; Thierens & Goldberg, 1993). Some researchers (e.g., Collins & Jefferson, 1991; Davidor, 1991) have recommended distributed populations for the discovery and maintenance of diverse *demes* (i.e., schemata). Although such schemes introduce intriguing new dynamics to GA search, it is clear that their effect on selection is second-order at best. Distributed populations only prolong the inevitable collapse of the population distribution by a number of generations that grows linearly with population size and number of subpopulations.

In addition to these simple approaches for maintaining diversity, many more complicated mechanisms have been introduced to artificial evolution. In artificial life models, for example, individuals interact with each other in many different ways, such as preying, trading, and communicating in primitive protocols. The LCS has been embellished with many types of mechanisms (e.g., message lists, bidding auctions, and taxation codes) to encourage the formation of complex interdependent social groups that can express much more complicated and sophisticated behaviors and concepts than can individuals (e.g., default hierarchies, rule chains, and rule trees). All of these attempts to more realistically model social interactions from ecosystems, economies, societies, and cultures, lead to very complicated systems with dynamics that vary widely between specific implementations. Such systems are therefore difficult to separate into analyzable pieces.

In this dissertation we start simply. By looking only at "simple" niching (via sharing) we introduce a low level type of interaction that nonetheless can induce complex "social dynamics", such as intertwined competition and cooperation among individuals and groups of individuals, simultaneously. Yet niching by itself (including normal GA selection) is a simple and general enough mechanism to allow us some chance for modeling it and actually learning something general from it.

2.2.2 Various Nichers

A number of niching mechanisms have been proposed and used over the last couple of decades. One of the earliest was Cavicchio's *preselection* (Cavicchio, 1970; Mahfoud, 1992), in which offspring could only replace one of their parents. De Jong's *crowding* (De Jong, 1975; Mahfoud, 1992) had the same flavor, in that new individuals replaced less-fit, but similar, solutions in the old population. Boltzmann tournament selection has also been shown to have niching effects (Goldberg, 1990; Mahfoud, 1991), and recently immune system models (Smith, Forrest, & Perelson, 1993), which are very similar to the stimulus-response (S-R) LCS, have been gaining attention for maintaining multiple solutions. In this paper, we limit our comparison to *fitness sharing*, introduced by Goldberg and Richardson (1987), studied in detail in (Deb, 1989; Horn, 1993;Mahfoud, 1993), and challenged by a massively multimodal problem in (Goldberg, Deb, & Horn, 1992). We also consider the closely related *resource sharing* which induces niching implicitly (thus it has been called *implicit niching* by Horn, Goldberg, and Deb (1994)). Fitness and resource sharing are indeed so similar in their behavior, that a unified model of *sharing* for niching is possible, and will be explored later in this thesis.

2.2.3 Fitness Sharing

Fitness sharing accomplishes niching by degrading the *objective* fitness (i.e., the unshared fitness) of an individual according to the presence of nearby (similar) individuals. Thus this type of niching requires a distance metric on the phenotype or genotype of the individuals. In this study, unless otherwise stated, we use the Hamming distance between the binary encodings (genotypes) of individuals. We degrade the objective fitness $f_i \equiv f(i)$ of an individual i by first summing all of the *share values* $sh(d)$ of individuals within a fixed radius σ_{sh} of that individual, and then dividing f_i by this sum, which is known as the *niche count* $m_i = \sum sh(d)$ for that individual. More specifically, if two individuals, i and j, are separated by Hamming distance $d_{i,j} \equiv d(i,j)$, then we add a share value

$$sh(d_{i,j}) = \begin{cases} 1 - (\frac{d_{i,j}}{\sigma_{sh}})^{\alpha_{sh}} & \text{if } d_{i,j} < \sigma_{sh} \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

to both of their niche counts, m_i and m_j. Here, σ_{sh} is the radius of our *estimated niches.* Individuals separated by σ_{sh}, or more, do not degrade each other's fitness. The parameters α_{sh} and σ_{sh} are chosen by the user of the niched GA based on some *a priori* knowledge of the fitness landscape or a specific user objective (e.g., minimum separation σ_{sh} of alternative solutions). The effect of varying α_{sh} has not been studied to the same extent as has been the effect of changing σ_{sh}. For this reason, α_{sh} is often set to one, yielding the *triangular sharing function.* Figure 2.1 shows a family of power curves for the sharing function as α_{sh} varies. The share value, or contribution to the niche count, always decreases monotonically from one to zero as the distance between the pair of individuals goes from zero to σ_{sh}. Varying α_{sh} does affect the "shape" of this decreasing function and hence the "shape" of the niche. Choosing σ_{sh} is trickier, since we need to have some idea of the size and separation of the niches.

For an individual i, the niche count m_i is calculated as

$$m_i = \sum_{j=1}^{N} sh(d_{i,j}), \tag{2.2}$$

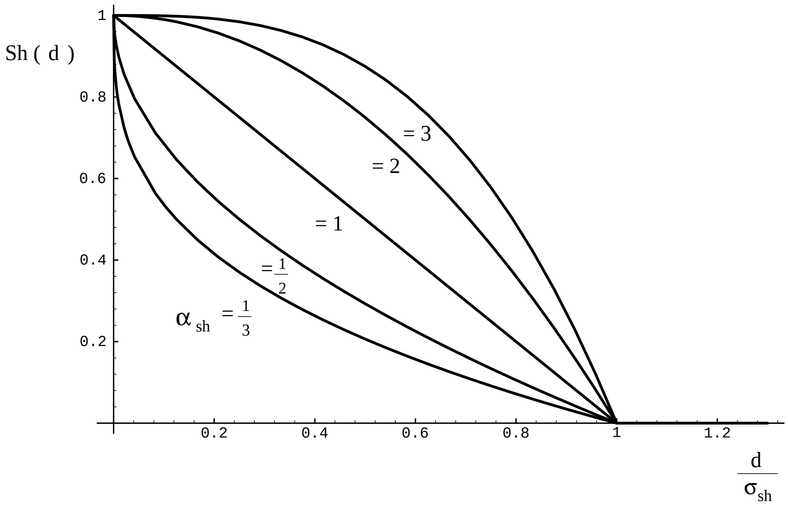

Figure 2.1: The sharing function, a function of distance d, as α_{sh} varies.

where N is the size of the population. The *shared fitness* $f_{sh,i} \equiv f_{sh}(i)$ of individual i is then given by

$$f_{sh,i} = \frac{f_i}{m_i}. \tag{2.3}$$

Fitness sharing tends to spread the population out over multiple peaks (niches) in proportion to the height of the peaks. GAs with proportionate selection and fitness sharing have been successfully used in solving a variety of multimodal functions (Deb, 1989).

2.2.4 Resource Sharing

A natural niching effect is implicitly induced by competition for limited resources (i.e., finite rewards). The basic algorithm common to all resource sharing systems is quite simple and intuitive:

1. For each of the finite resources r_i, divide it up among all qualified individuals contending for it, in proportion to their various merits (that is, the relative strengths of their claims. Thus two equally deserving individuals should be allocated equal amounts of the resource. If the resource is discrete, and cannot be evenly divided, then randomly choose among equally deserving individuals. This random choice results in an *expected* uniform distribution among equally deserving candidates.

2. For each individual, add all rewards/credits earned in the first step, and use this amount, perhaps scaled, as the fitness for GA selection.

3. After a new generation is produced, replenish/renew the resources and start over at the first step above.

4. Continue above loop until some stopping criterion is met.

The idea of splitting up a limited resource among all competing individuals seems so evident in nature, and is so simple to implement, that resource sharing is often incorporated in adaptive, or simulated, systems almost automatically

2.2.4.1 Applications and Instances of Resource Sharing

This simple and natural scheme has been abstracted into a number of population-based algorithms, including:

- Learning classifier systems (LCS),
- Immune system models,
- Multi-agent systems, and
- Ecological simulations.

We think the resource sharing approach is especially well-suited to several major, open problem domains,

- Classification,
- General covering problems, and
- Layout problems.

The reason for our belief in the appropriateness of sharing for such hard problems is the observation that sharing attacks two major difficulties of population-based solutions to hard problems: credit assignment, and problem decomposition.

2.2.4.2 Classifier Systems as a Detailed Example of Resource Sharing

An example of *resource sharing* occurs in most implementations of the *Michigan-style* learning classifier system (Horn, Goldberg, & Deb, 1994). In an LCS, the population consists of classification rules, or simply *classifiers*. These rules attempt to classify examples (either from some training set or some test set). The rules are rewarded for correct classifications of examples, earning credit for each correct classification of an example[2]. The sum of credits earned, over all examples, is used for each rule's fitness. This fitness is then used in normal GA selection.

In the Michigan LCS, individuals (classification rules, or *classifiers*) compete for the rewards (or credit) given for proper classification of a finite number of examples. Several researchers have shown that simply dividing up an example's reward/credit among all rules that successfully classify that example (i.e., sharing), effectively and robustly maintains a diverse set of rules that together "cover" the examples (e.g., Booker, 1982, 1989; Wilson, 1986, 1987, 1994, 1995). Thus LCS sharing is an instance of resource sharing in which the resources are the rewards/credits for the examples. Henceforth, we will consider the examples themselves to the resources to be shared, in order to simplify our discussion. Thus this strategy is often known as *example sharing* (McCallum & Spackman, 1990; Neri & Saitta, 1995, 1996).

The sub-goal of resource sharing is to *cover* (exploit) as much of the resources as possible. The only type of interaction between individuals is competition for the same resource, and the natural mechanism for handling such competition (and encouraging search for uncovered resources) is *sharing* of contested resources. Thus similar individuals (species) share common resources by dividing them up among themselves. This simple method induces *niching* or *speciation*, an emergent phenomenon.

The notions of competition and niche overlap are easy to visualize in the case of resource sharing. In Figure 2.2, the large rectangle represents the space of all *examples* given to the LCS for learning. The size of a circle represents the number of *examples* covered by the corresponding rule, and hence its accuracy. The overlaps of circles represent overlaps of coverage among rules, and thus contain the examples "shared" by two or more rules.

To illustrate the actual sharing of resources that leads to implicit niching: let f_A and f_B be the objective fitnesses for rules **A** and **B** respectively. The objective fitness could be taken

[2]And perhaps earning penalties for incorrect classifications. But we postpone considering such extensions until later in this dissertation.

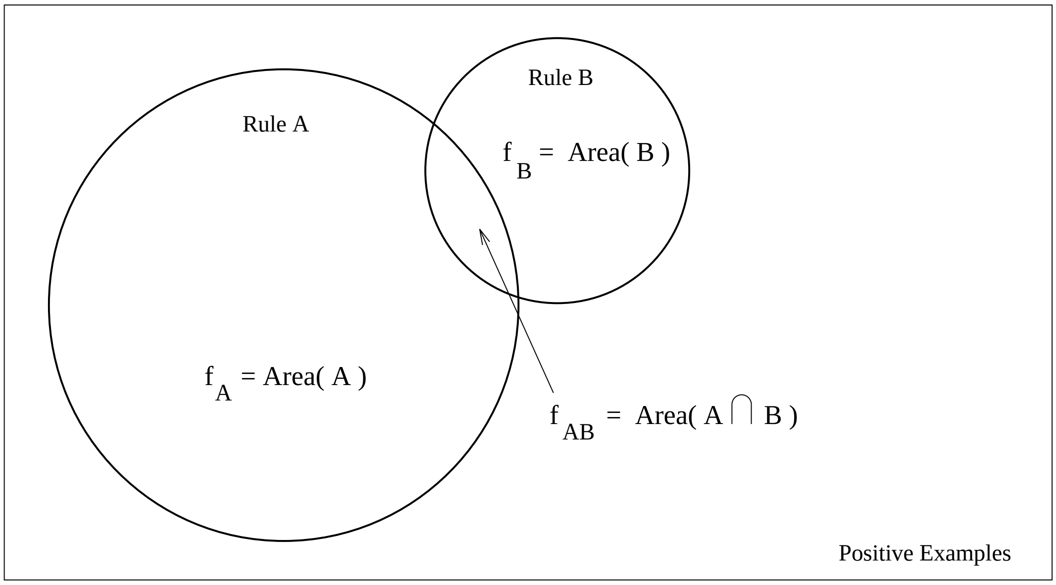

Figure 2.2: In the case of the learning classifier system (LCS), *implicit niching* is induced by rules competing to classify examples. We can use diameter in the space of examples to indicate a rule's coverage, which is also its *objective* (unshared) fitness.

as the number of examples covered[3] by that rule, in the case of binary classification. Let f_{AB} be the amount of resources in the overlapping coverage of rules **A** and **B**. That is, f_{AB} is the amount of resources shared by **A** and **B** (e.g., the number of examples covered by both). Let n_A, n_B be the number of copies of rules **A** and **B**, respectively, in our population. Then we can calculate the shared (expected) fitness of rule **A**:

$$f_{sh,A} = \frac{f_A - f_{AB}}{n_A} + \frac{f_{AB}}{n_A + n_B}. \tag{2.4}$$

Similarly for rule **B**,

$$f_{sh,B} = \frac{f_B - f_{AB}}{n_B} + \frac{f_{AB}}{n_A + n_B}. \tag{2.5}$$

The LCS's niching mechanism is given more study in the next chapter. In particular, more effort is spent to properly isolate the niching mechanism (and its effect) from the many other complex types of interaction frequently introduced to the LCS by implementors and users.

[3] In other words, classified correctly.

Also in Chapter 3, comparisons are made between fitness and resource sharing are initiated, and which continue throughout the dissertation.

2.3 Summary

In this chapter we have tried to motivate the further analysis of niching via sharing by first discussing how effective the simple GA searches large and difficult search spaces. Then we noted the inability of the simple GA to maintain diversity in the population for any significant amount of time. We then discussed the success to date of fitness sharing and various types of resource sharing as methods for maintaining useful, meaningful diversity in the form of multiple *niches*. Sharing has been implemented in several forms, which have been applied and studied separately. Often, resource sharing is buried in the complex details of elaborate approaches to problems, such as in the learning classifier system. We hope to motivate the abstraction of sharing from evolutionary computation, to be studied here in relative isolation, as an important GA operator in and of itself. And we hope to motivate a unified analysis of sharing abstracted from its different forms (e.g., fitness sharing, resource sharing, example sharing, antigen sharing, credit sharing, etc.).

Chapter 3

Frameworks For GA Niching

To further motivate the study of niching in GAs, we place it in several larger contexts, in which it forms a basis for further analysis of more complex interactions. For example, resource sharing is a fundamental factor controlling population size in theoretical ecology. As another example, in the optimization of "context-dependent fitness functions", fitness sharing appears to be a very simple, basic type of context-dependency. In this chapter we make a few tentative, but provocative, connections with models from theoretical ecology. We also show how the simple act of sharing (resources or fitness) induces a hierarchy of different levels of cooperation and competition among individuals and groups of individuals, producing complex behavior we might have associated only with more explicit forms of interaction. Next we propose a categorization of different possible *context-dependent fitness functions*, ordered into a lattice, in which niching forms the first rung up the ladder from the strictly context-independent fitness of the simple GA. Finally, we place fitness sharing and resource sharing side-by-side, exposing their similarities and differences, and making clear that both are two specializations of the more general procedure of simply *sharing*.

3.1 Theoretical Ecology

GA niching is inspired by the natural phenomena of speciation and specialization in natural ecosystems. In the study of natural ecologies, a great deal of abstract analysis, mathematical modeling, and testing of models has been published. Here we explore connections to the very simplest, most widely accepted models of "real niching" in natural ecologies.

GA theorists are in an enviable position, at least enviable from the point of view of population biology theorists. Many of the most basic models of theoretical population biology, such as exponential and logistic growth equations, or the Lotka-Volterra competition model, are of very limited applicability in biology. They are generally considered to be too simplistic to be of any predictive value and are well-studied mostly out of mathematical interest (they are more readily manipulated) and as a foundation for more realistic models. But for the simple, abstract ecosystems of our GAs, these models can prove most applicable.

As an example of a simple model from theoretical ecology that is of great use, we discuss below the logistic growth model (e.g., Hofbauer & Sigmund, 1988) as a model for the simple GA. In the next chapter we expand on this mapping by relating competition models to GA niching.

3.1.1 The Logistic Growth Model

Before applying resource competition models to the niched GA, we must first take into account a nearly universal assumption of GAs, simple or not: The fixed population size N. Perusing some of the basic theoretical ecology literature, it is quite clear that their models in general do not assume such a fixed, finite size (e.g., the basic model for unbounded, exponential growth). Different models assume different mechanisms for limiting population growth, if it is limited at all. For example, Roughgarden considers a whole class of *density-independent* factors that check population growth, such as weather and other environmental fluctuations. He also considers several different types of *density-dependent* factors (that is, those whose effects on future population numbers are a direct function of the current population number), only one of which is *resource depletion*. Other density-dependent factors include foraging effort, social interaction, and intensity of predation (since predators can focus on the most abundant prey type). Thus we must be explicit in accounting for our fixed population sizes if we want to apply extant models from theoretical ecology.

It seems that the most natural explanation for our use of finite, fixed population sizes is the need to use a shared, finite resource: computational power. (Ease of implementation and analysis I would argue are secondary motivations.) Due to memory and/or processing time constraints, we must limit our population size. We can consider *population slots* to be a single shared resource for which the population members compete. If we assume N units of this discrete resource, allocate either zero or one unit to members, and then select all and only those individuals with one resource unit, we have enforced the fixed population size.

The limitation on population growth due to the shared use of a single finite resource is accounted for in the simple model of logistic growth. Hofbauer and Sigmund (1988, p. 35, equation 6.8) give the following recurrence relation for a logistic growth of a population with non-overlapping generations:

$$y' = Ry(1 - \frac{y}{K}), \tag{3.1}$$

where y and y' are the the numbers of the population at time t and $t+1$ respectively. The value R is the *initial rate of growth* while K represents the maximum population size allowed by this model (not necessarily N)[1].

This model can be directly applied to a simple generational GA that uses binary tournament selection, as we show below. (Mappings to other selection methods seem to be more involved, and will not be considered here.)

3.1.2 Logistic Growth in the Simple GA

Assume we have a fixed size population of N members and only two possible species **A** and **B**. Let n_A^t and n_B^t be the number of copies of **A** and **B** respectively, at time (generation) t. Thus $n_A^t + n_b^t = N$, $\forall t \leq 0$ (note that $t = 0$ denotes the first, randomly initialized generation). Furthermore, without loss of generality, let us assume $f_A > f_B$, so that under pure selection (i.e., no mutation or crossover) we expect **A** to take over the population ($n_A \rightarrow N$, $n_B \rightarrow 0$) rather rapidly under tournament selection.

We consider for now only binary tournaments and a generational (non-overlapping population) GA[2]. Since **A** will win every tournament in which it is included, the probability of **A** being selected as a result of a tournament between two randomly chosen competitors is $p(A) = 1 - (1 - n_{A,t}/N)^2$. Given $n_{A,t}$ copies of **A** at generation t, the expected number of

[1]Hofbauer and Sigmund (1988) note that $0 \leq y < K$ are the limits on y imposed by the model. Also $1 < R < 4$ are the restrictions on R. If $R \leq 1$, there is no growth, or negative growth, while if $R \geq 4$, then y' can be $> K$ for some $y < K$, which would be a violation of the first restriction.

[2]See (Goldberg & Deb, 1991) for a more detailed discussion of this derivation and extensions to tournament selection with larger tournament size s and probabilistic tournaments, as well as to other types of selection.

copies of **A** at the next generation $t+1$ is

$$n_{A,t+1} = N(1-(1-\frac{n_{A,t}}{N})^2).$$

Rearranging the above equation yields

$$n_{A,t+1} = 2n_{A,t}(1-\frac{n_{A,t}}{2N}). \tag{3.2}$$

Comparing this to equation 3.1 above, we can see that $R=2$ and $K=2N$.

The initial growth rate, $R=2$, is due to the selection pressure of binary tournaments between randomly chosen individuals. Each copy of **A** expects to be in two tournaments. Initially, when the population is mostly made up of **B**s, each **A** can expect its opponent to be a **B** and so it can expect to win both its tournaments and be selected for reproduction twice. Thus the early growth of **A** is exponential with base two: $\approx 2^t$, with $n_{A,t}$ doubling with successive t (again, for small t and small $n_{A,t}$.

As for K, note that this equation is valid only for $0 \leq n_{A,t} < 2N$, that is, $y < K$ in Equation 3.1 above. Otherwise, we get $y' \leq 0$. Thus K represents the maximum population size for this model. But as long as the initial population size $n_{A,0} < K$, and the growth rate $R < 4$, our dynamical system cannot leave the region $0 \leq n_{A,t} < 2N$.

Subtracting $n_{A,t}$ from both sides of Equation 3.2 and rearranging, yields

$$n_{A,t+1} - n_{A,t} = \Delta n_A = n_{A,t}(1-\frac{n_{A,t}}{N}), \tag{3.3}$$

which directly corresponds to Roughgarden's (1979) difference form of the logistic equation:

$$\Delta n = r(1-\frac{n}{K})n. \tag{3.4}$$

According to Roughgarden:

> The symbol K here is called the carrying capacity because it is a measure of the amount of renewable resources in the environment in units of the number of organisms these resources can support.

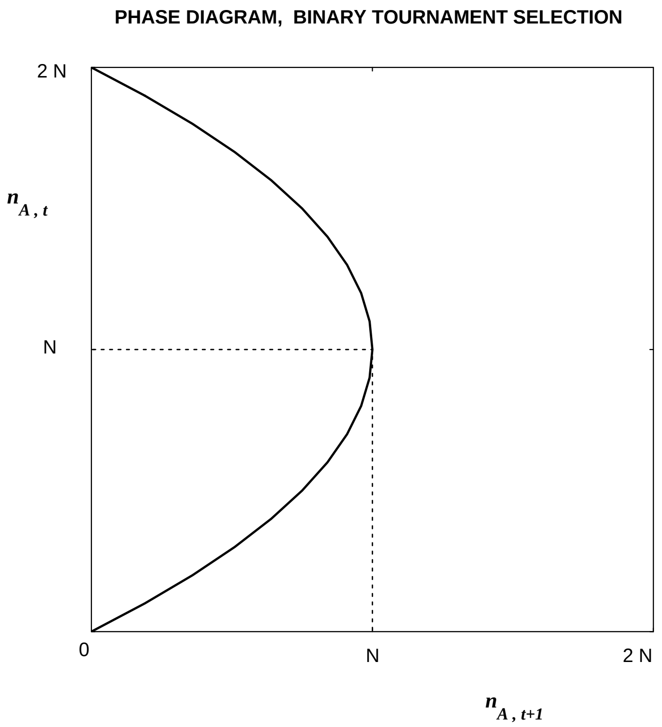

Figure 3.1: A phase diagram for a resource depletion: a natural analog to the simple GA. The expected population trajectory is shown. Clearly, $n_t = N$ is the equilibrium.

Comparing Equation 3.4 to Equation 3.3, we can see that Roughgarden's $K = N$. Indeed, in both Equation 3.3 and in Equation 3.2 the fixed points $(n_{A,t+1} = n_{A,t})$ are clearly at $n_{A,t} = N$.

So the equilibrium, or steady-state population, in the single finite resource model is the same as the fully converged finite population of the simple GA under selective pressure. In other words, we can view the artificial, strict constraint of fixed population size in the simple GA as a convenient short cut to the full simulation of the natural sharing phenomenon among members of a single species.

Figure 3.1 illustrates this correspondence. In the figure, we plot the recurrence relation of Equation 3.2: that is, the expected next generation $n(A, t+1)$ given the present population $n(A, t)$. The sharing of the finite resource drives the population to N members. As we will see later in reviewing and graphing a Markov analysis of the simple GA, the fixed size population restriction of the GA can be seen as simply a limitation of the behavior of a natural population to the lower left quadrant of the phase space in Figure 3.1. Indeed, the plotted Markov transition matrices for a simple GA will bear a striking resemblance to this lower left quadrant of Figure 3.1. But that comes later.

3.2 Levels of Cooperation and Competition

Although the terms "cooperation" and "competition" are imprecise and often undefined except in some intuitive way, it is quite clear that these are useful concepts. It is generally agreed that cooperative and competitive relationships can co-exist, that cooperation and competition can occur on several levels (e.g., between individuals, groups, and species), and that cooperative and competitive interactions can take place in very complicated and intricate ways. For example, cooperative individuals can outperform non-cooperative (lone) individuals, so that competition fosters cooperation. In turn, groups of cooperative individuals can compete with each other, so that cooperation fosters (group) competition. It is also generally agreed that cooperation and competition combined correctly can be extremely effective at problem solving (e.g., optimization, learning, design), but that specifying and fostering the correct balance of cooperation and competition is difficult. Lately, increased interest in *emergent cooperation* has been fueled by the renewal of *distributed artificial intelligence* (DAI) (Bond & Gasser, 1988) for application to "agent-based computation", and "work group problem solving" on the world-wide internet.

Here we have a chance to define and discuss cooperation and competition, emergent in our niched GAs, in very concrete terms. I will not, however, try to cast this framework as anything more than speculative and suggestive. In particular, I do not expect to definitively classify cooperative and competitive relationships in terms of levels and types. Rather my purpose is to point out the surprisingly rich diversity of cooperative/competitive relationships engendered by the rather simple interaction prescribed by sharing. Even before adding other types of interaction to our GA, such as communication, we find several levels of cooperation and competition with sharing by itself.

Figure 3.2 depicts several levels of cooperation and competition in a niched GA. At some basic level, call it level 0, all individuals are competing for the same resource: population slots, as discussed earlier. At this level, every individual is competing against every other, as individuals,

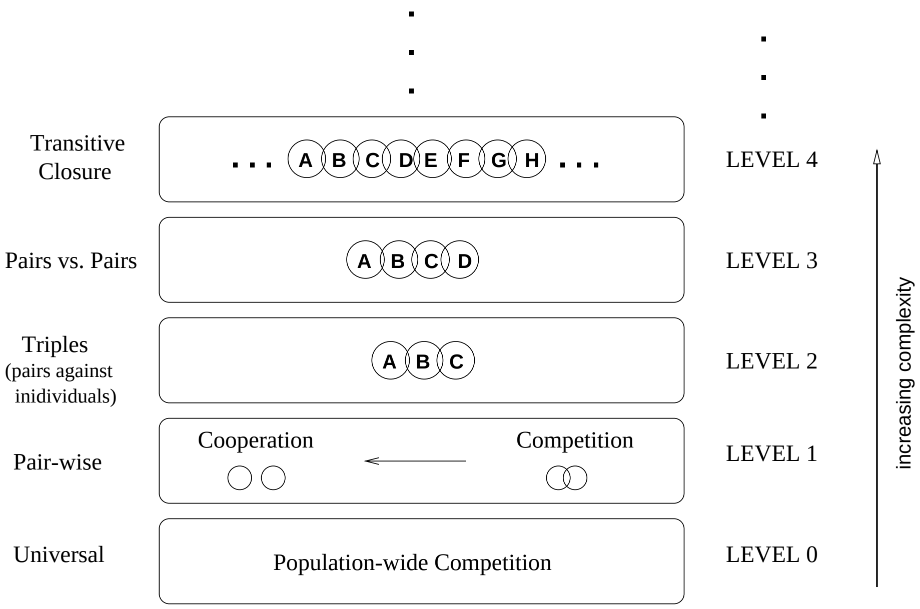

Figure 3.2: Multiple levels of cooperation and competition, building on each other, induced by the simple act of sharing in a GA.

not as species. The competition at level 0 is universal and there is no competition. At level 1, pairs of individuals either have overlapping niches or non-overlapping niches. Overlapped niches imply competition for some common resource. Thus the more overlap, the greater the competition. A complete lack of overlap, it can be argued, represents an "implicit" or "default" type of cooperation, defined solely as the lack of (level 1) competition. This is all quite familiar. Pairs of individuals can be categorized as either cooperating or competing. But level 1 competition and cooperation lead to another type of cooperation (a higher level, in my opinion). At level 2, two species, **A** and **C**, are level 1 cooperators with each other, and both also are level 1 competitors with the species **B**. By having a common level competitor and being level 1 cooperators, species **A** and **C** are cooperating more strongly than at level 1. Here at level 2 they are ganging up on **B**. Increasing the number of copies of **A** reduces the number of copies of **B** which in turn increases the number of **C**s. And vice versa. Thus level 2 cooperation is more direct, in that one species can increase the (shared) fitness of the other by being more populous, and vice versa.

Now if **A** and **C** are directly cooperating (level 2) and **C** and **D** are competing (level 1), then **A** and **D** can be said to be competing (any enemy of yours is an enemy of mine!), and we call this “level 3 competition”. Similarly, **A** can be said to be cooperating at level 3 with any other competitor of **D**. We can proceed by induction (any friend of yours is a friend of mine, etc.), and so at level 4 we simply take the transitive closure of these lower level cooperative and competitive relationships to define large groups of cooperating individuals, competing with each other, a powerful phenomenon that actually occurs in niched GAs, as we will demonstrate. The emergence of long chains of cooperating species, and the other implications of multiple overlapping niches, will be examined later in in this thesis when we discuss scaling up of our results to multiple niches/species.

3.3 Context Dependent Functions

Although we focus on GAs with niching, the questions we attempt to answer are pertinent to more general GA optimization of *context dependent fitness functions.* Such functions require that an individual's fitness be evaluated in the context of the current environment. In particular, individuals can affect each other's fitness. This is the case not only with niched GAs, but with learning classifier systems (LCS), immune system models, and artificial life.

Very little analysis of context-dependent function optimization is to be found in the literature. One of the few examples is Dawid's paper (1994). Dawid extends previous Markov chain analyses of genetic algorithms operating on “state independent fitness functions” to any type of *state dependent fitness function.* That is, he looks at the general case of a fitness function that varies in some way, any way, with time t (including, therefore, niching, in which individual shared fitnesses vary with the current population distribution) . He limits his analyses to asymptotic results, a limitation we discuss later. In this section we sketch the beginnings of a possible hierarchy of context-dependent fitness functions, and place Dawid's work as well as the current analysis of niching within this framework.

3.3.1 A Function Hierarchy

First, let us define our usual *context-independent fitness function* $F(x)$ from the simple GA, where x is some individual (i.e., chromosome). Thus $F(x)$ is static, deterministic and depends only on the individual being evaluated. At the most general level, a context-dependent fitness function $F(x, C)$ is a function of both the individual and its context (or *state*) C. The context C

is determined by (1) the current population, (2) effects of previous populations, and (3) outside influences. The current population can be considered part of the context, while the effects of the past (e.g., previous populations) are recorded in the current physical *environment*, including artifacts created and left behind (e.g., message lists , memory stores). Finally, outside influences include random fluctuations and events in noisy and nondeterministic fitness functions. Thus the context C is a catch-all category for all other factors besides the current individual.

Let us break C down into the more essential independent variables: the current population, the physical aspects of the environment, and time. Thus our most general context-dependent function becomes $F(x, \vec{P}, E, t)$, where $\vec{P}$ is the current population distribution, E is the *environment* (that is, the physical record of effects from previous generations and previous outside influences), and t is time, in generations. So we have rewritten the context as a triple: $C = (\vec{P}, E, t)$, where we incorporate into E the dependency on history (i.e., the physical effects of previous populations and previous outside influences). Also, we incorporate into F the application of current (at time t) "outside influences" to the context-dependent fitness[3] of x.

Typically, the individual x is evaluated at time t in the current physical environment E_t: $\text{fitness}(x) = F(x, \vec{P}, E_t, t)$, and the static environment is then updated via some function $\mathcal{E}$ of the current context: $E_{t+1} = \mathcal{E}(\vec{P}, E_t, t)$. (Note: the updating of the population $\vec{P}$ from generation t to $t+1$ is left to the GA itself!)

If we accept this model $F(x, \vec{P}, E_t, t)$ of context-dependency, then Figure 3.3 illustrates a tentative function hierarchy. Actually, what is shown here is a portion of a partially ordered space of functions, where the ordering relation is "generality versus specificity". And I have chosen a Hasse diagramming convention of placing the a more general function above a more specific one and drawing a connecting line between them. Thus if two functions are connected by a line, the lower one is a specific instance of the upper one. The subset of the partially ordered space that is shown here is actually a lattice, with a single unique most general function $F(x, \vec{P}, E, t)$, and a single, unique most specific (i.e., least general) function, $F(x)$.

We are most interested in the special case of $F(x, \vec{P}, E, t)$ which is independent of both E and t: $F(x, \vec{P})$. This is the *strictly-population-dependent* class of functions. It is the class of

[3]There are of course many alternative, reasonable function definitions, such as $F'(x, \vec{P}, \vec{H}, t)$ in which $\vec{H}$ is a *history vector* (i.e., list of all previous populations $\vec{P_{t'}}$ for $0 \leq t' \leq t$), and F' therefore computes the current environment E based on history $\vec{H}$. But the form $F(x, \vec{P}, E, t)$ seems closest to our implementations.

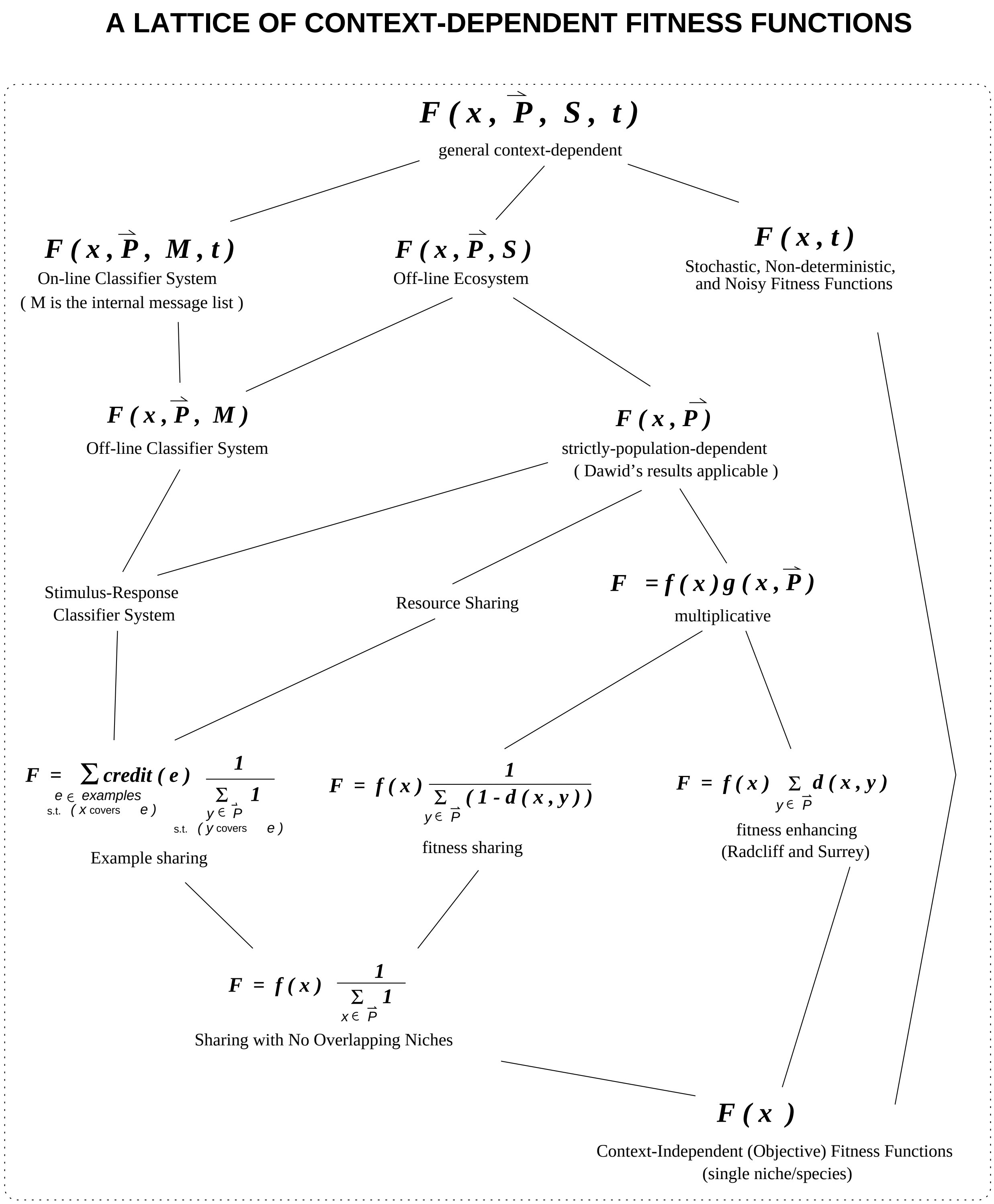

Figure 3.3: A lattice of context-dependency, with the most general case being a function dependent on the current population *and* the physical environment, as well as non-deterministic outside effects. The least general case is dependent solely on the individual x.

context-dependent functions that Dawid (1994) examines, although he uses the more general term *state dependent fitness function*, a term we will avoid as it conflicts with the terminology adopted above. Dawid's Markov chain analysis is general to all functions of the form $F(x, \vec{P})$. Indeed, this class of functions is well-suited for a straightforward Markov analysis because a GA optimizing such a function has the *Markovian property* (Hillier & Lieberman, 1990); that is, the transition function is dependent solely on the *current* state, and not on any previous states (where the state is defined as the population distribution in the current generation).

Other classes of functions could be modeled using more complex Markov analyses. For example, the class of functions taking into account outside influences (e.g., weather), which we model here as time-dependent, $F(x, \vec{P}, t)$, can be modeled using *non-stationary Markov chains* (i.e., using non-stationary transition probabilities) (Hillier & Lieberman, 1990). We might use *non-deterministic Markov chains* (Hillier & Lieberman, 1990), with probabilistic transition probabilities, to model noisy fitness functions. For functions *finitely dependent* on previous generations, $F(x, \vec{P}, E)$, in which the environment E depends only on the last k generations, we could recover the Markovian property by expanding the number of states so as to model all possible (finite) histories (Hillier & Lieberman, 1990). (For M different possible populations, there are M^k different possible "histories" consisting of k successive populations.) To make such a situation seem reasonable, we could imagine an environment in which the effects of individuals gradually fade away. For example, in many learning classifier systems, the messages posted by individual rules on the message list have finite expirations.

Returning to the class of *strictly-population-dependent* functions $F(x, \vec{P})$, we realize that Dawid's Markov analysis does apply to the entire class, but that it is a purely asymptotic analysis. His results only show that even with population dependent functions, the long-term distribution is still clustered around homogenous states (i.e., uniform populations) as has been shown to be true for strictly static (i.e., context-independent) functions $F(x)$. Dawid does not explore the albeit transient diverse equilibrium states such as niching equilibrium.

The strictly-population-dependent class also includes both fitness and resource sharing and indeed probably all types of niching functions. For example, in the case of fitness sharing: $f'(x) = \frac{f(x)}{g(x,\vec{P})}$, where $f'(x)$ is the shared (or degraded) fitness of individual x, $f(x)$ is the objective (i.e., unshared) fitness of x, and $g(x, \vec{P})$ give the niche count m_x, as $\sum_{y \in P} Sh(x, y)$. Thus fitness sharing can be called a *multiplicative* subclass of $f(x, \vec{P})$, in which the decomposition consists of an "objective fitness" term strictly dependent on x only, multiplied by some function of the entire population $\vec{P}$. Later in this dissertation we will find that Radcliffe and Surrey's (199X) niching algorithm also utilizes a similar but distinct multiplicative instance of $f(x, \vec{P})$. However, it will turn out that resource sharing, unlike fitness sharing, is **not** multiplicative. We might further characterize (i.e., subclassify) fitness sharing, since it is strictly a division in which the divisor is a simple summation of pairwise interactions between population members. (Furthermore, the pairwise interactions are linear when $\alpha_{sh} = 1$.) Thus we might call fitness sharing an example of *multiplicative, pairwise additive strictly-population-dependent* fitness functions! Some of the later results generalize to any member of such a subclass.

3.4 Niching in the LCS

In this section we focus on the sharing of examples in a LCS (i.e., the sharing of credit for proper classification of examples during training of a LCS). We isolate the mechanism and effect of the sharing-induced niching in the generally complex LCS. This is not a straighforward task, as classifier systems in general tend to be large and complicated, with many augmentations tailored to the classification problem at hand, with many levels and types of interaction between rules in the population.

3.4.1 Background: Analyzing Resource Sharing in the LCS

The learning classifier system[4] (LCS) is particularly difficult to analyze because of its complexity. Even if we can all agree on a canonical "simple LCS", that model is bound to be much more complex than the "simple GA". The primary reason for the additional complexity is the multiobjective nature of the task at hand. The most basic LCS is trying to find the (1) smallest set of rules that (2) best solves the example problem while (3) generalizing well to all similar problem instances. In terms of classification, this means searching for a concise, accurate, and robust concept description, where a concept description is a group of rules.

In an effort to accomplish such multiple objectives simultaneously, the research community has introduced a number of simple, elegant, and/or intricate mechanisms for competition and

[4]See (Holland, 1971, 1975, 1985, 1992; Booker, 1982; Goldberg, 1983, 1989a; Wilson, 1986; Wilson & Goldberg, 1989).

cooperation among individuals. Such mechanisms include internal message lists for inter-rule communication, Holland's bucket brigade (Holland, 1985) for temporal credit allocation, and specificity-based bidding to allow default hierarchies to form. It is these additional mechanisms for rule interaction that induce the complex cooperative and competitive relationships among rules that in turn make the analysis of the LCS so much more difficult than that of GAs.

In a simple GA, each individual is evaluated according to a single scalar fitness function independent of other members in the population. We can thus view the task of the simple GA as optimization of the sum, or average, of the population's fitnesses. The optimum population consists entirely of copies of the best individual. But when individuals influence each other's fitness evaluation, the task of the GA can no longer be modeled as an optimization of total population fitness. Indeed, the multiple objectives mentioned above cannot be combined, in general, into a single, scalar measure of population fitness. Such evolution has gone by the name of *coevolution*, *co-adaptation*, and *context-dependent function optimization*. Coevolution is more natural, more realistic, potentially more powerful, and certainly more complex than the convergence of a simple GA.

To understand the complex relationships that can and do emerge among rules in the LCS, and among individuals in coadaptive systems in general, we believe it is necessary to analyze each type of rule interaction in isolation. Here we concentrate on the *weak cooperation* induced by rule competition for limited resources (i.e., reward sharing). The sub-goal of weak cooperation is to *cover* (exploit) as much of the resources as possible. The only type of rule interaction is competition for the same resource, and the natural mechanism for handling such competition (and encouraging search for uncovered resources) is *sharing* of contested resources. Thus similar rules share common resources by dividing them up among themselves. This simple method induces *niching* or *speciation*, an emergent phenomenon that we suggest is prerequisite to all other types of cooperation (i.e., *strong* cooperation).

Though a simple mechanism, resource (or fitness) sharing induces complex behavior in the evolution of the population. Indeed, a simple LCS with sharing alone is worthy of study because it retains the first two of the multiple objectives listed above by finding the *individually best* rules that *together cover* as much of the resources as possible (with a finite population). Yet the behavior of the LCS with sharing alone is poorly understood. This is especially true when rules overlap (compete for the same resources) to varying degrees, thus making it difficult for the GA in the LCS, and indeed for the human designer, to distinguish cooperation from competition.

But sharing has already been introduced to the simple GA (Goldberg & Richardson, 1987) in the form of *fitness sharing*. Since 1987, several other studies have increased our knowledge of the effects of sharing on the GA. By isolating fitness sharing in the LCS we can relate fitness sharing to the LCS and come to a better understanding of this one critical subfunction of the overall LCS.

3.4.2 Background: Previous Analyses

Smith and Valenzuela-Rendón (1989) identified the general need for niching in the LCS. They applied explicit fitness sharing to a small LCS using Hamming distance as a metric on the space of rules. They were successful in maintaining a set of diverse rules that together covered the examples and solved the problem. However, they did run into the *separation* problem inherent in fitness sharing and identified in (Goldberg, Deb, & Horn, 1993). Briefly, the separation of desirable and undesirable individuals can be a problem for fitness sharing because of the fixed niche radius σ_{sh}.

Most recently, Smith, Forrest, and Perelson (1993) analyzed implicit niching in the immune system model. They noted that a similar niching process takes place in the LCS. They went on to show that the immune system model does indeed exhibit "emergent fitness sharing" with many of the properties of GA explicit fitness sharing. The immune system model is very similar to the S-R LCS, with the exception that rules cannot use "don't care" wildcard symbols (i.e., "#"). Generalization is induced by taking a limited size random sample of rules (a.k.a. antibodies) to compete to cover an example (antigen). Much of their analysis was thus focused on the sampling process and its effects. In this paper we avoid the issue of rule generality and concentrate on the implicit niching induced by arbitrarily breaking ties between rules competing for a resource. This arbitrary tie-breaking is used in the immune system model as well as the GA. It leads to an expected even division of a resource among all individuals competing for it, and is the key to sharing, implicit or explicit.

3.4.3 The Need for Niching in the LCS

In the LCS community it is commonly believed that the GA is used so sparingly, and with so many other specialized operators at work, that it is virtually impossible for a single rule to take over the population. And this is often the case. Specialized selection operators such as elitist selection, or protective group measures such as the formation of classifier corporations, or any kind of cooperative-reinforcement fitness function such as the bucket brigade or epochal credit assignment, all help counteract convergence. But this observation does not help us understand or predict the behavior of the LCS.

The LCS in its simplest form, arguably a stimulus-response LCS for binary classification as we outline below, should be able to maintain a diverse group of high-quality rules that together classify the examples at hand. Furthermore, if we try to maintain diversity by lowering selection pressure, we deprive the LCS of the power of GA search. Therefore, maintenance of rule sets must take place over a time frame that is at least an order of magnitude greater than the convergence time of the GA. The only way to maintain such high-quality diversity in the face of high selection pressure is to balance convergence with a restorative force, such as "niching pressure" (Horn, 1993).

3.4.4 Methodology: Isolating Resource Sharing in the LCS

In this section we describe the critical process of isolating one LCS mechanism, implicit niching, and its effect on LCS behavior. To do so, we idealize the LCS by eliminating or simplifying operators. At the same time, we must simplify the corresponding classification task which the resulting LCS, with its remaining idealized mechanisms, should accomplish. Our simplifications, idealizations, and assumptions are guided by intuition and the methodology of functional decomposition.

The basic method of functional decomposition has been successful to date in the analysis of the simple GA (Goldberg, 1993, 1994). Goldberg (1993, 1994) separates the fundamental functions of a simple GA as building block generation, isolation, growth, and mixing. In the case of the LCS, we separate the function of weak cooperation from strong cooperation, the function of rule maintenance from the function of rule search, and the functions of accuracy and covering from that of generalization. We do not however, separate the issues of accuracy and covering, since these appear to be so closely related that their separation would make subsequent analysis meaningless[5].

In the following subsections, we first summarize the key assumptions. We then cast the resulting, simplified LCS learning task as a set covering problem. Finally, to further illustrate the nature of our simplifications, we view the learning task as a search through the schema space of the class membership function, and as a well-known machine learning problem.

3.4.5 The Assumptions, in Brief

We begin our process of functional decomposition by making the following assumptions about the LCS:

- Ternary alphabet
- Stimulus-response classifiers
- Binary classification
- Equal specificity of all rules

Our first simplification is a common one in LCS analysis as well as LCS applications: we use a ternary alphabet. The classifier conditions are coded from the set $\{0, 1\#\}$, where # is the "don't care", or wildcard, character. Responses are limited to the alphabet $\{0, 1\}$. We assume k attributes to describe the environmental input. Each attribute is encoded by ℓ_i bits, $i \in 1..k$. So our environmental input vectors will have $\ell = \sum_{i=1}^{k} \ell_i$ bits total. We can concatenate all our attributes into a single bit string representing an individual example (or more generally, an environmental input vector). Equivalently, we can imagine that we have ℓ binary attributes in our problem. Thus each rule's condition is a string of length ℓ taken from $\{0, 1, \#\}$ (e.g., ###00#1011##0#11).

Our second simplification is also a common one in LCS analysis: we limit ourselves to a *stimulus-response* (S-R) classifier system. In a S-R LCS there is no internal message list and hence no passing of messages between classifiers. The output of each classifier is a single

[5]E.g., the most general classifier, ###...#, covers all examples but on most problems it would be very inaccurate.

response to a single input from the environment. With no message list, the LCS cannot evolve "strong" cooperation. That is, rules cannot communicate directly with each other, nor can they pay or charge each other credit. We thus avoid the question of delayed reward, eliminating the need for complex credit assignment mechanisms, such as the bucket-brigade algorithm. What we are left with is *weak* cooperation, in which rules cooperate indirectly to *cover* the different reward situations (e.g., correct classifications of examples).

We further restrict our classification task to binary classification (i.e., single class membership). The goal of the LCS is to learn to classify instances as either members of a class (or concept) or not members. The output of each classifier is either a "1" (member of class) or a "0" (not a member). If we also assume a default rule ($\#\#\#...\# \Rightarrow 0$), then the task of the LCS is to find accurate rules that classify the exceptions (i.e., the positive examples). Therefore, we can assume that all rules besides the default have an output of "1". We can then leave the response (output) out of the encoding, and search only among the possible condition vectors (e.g., 110###0#0###101#).

Finally, we make a very important simplification to eliminate the issue of specificity versus generality. We assume that all rules in the population have the same number of "don't care" characters #. With each rule applying to an equal volume of the total classification space[6], there can be no preference for specific or general rules. All reasonable fitness functions for individual rules reduce to a function of accuracy only. And since all rules have equal volumes of coverage (applicability), any reasonable fitness function based on accuracy must be a monotonically increasing function of the number of examples covered by the rule. So we can simply use the number of examples covered (again, assuming all examples are positive ones) to order the rules according to fitness, even though we don't know the exact, possibly nonlinear, fitness function.

3.4.6 What's Left? A Hard Problem: Set Covering

After all the above simplifications and assumptions, it is essential that we verify that we still have a difficult, interesting, albeit abstract, problem.

When we limit ourselves to binary classification and assume a default class (i.e., we are really only trying to identify one class), we are left with an instance of the set covering problem.

[6]The entire classification space is of size 2^ℓ, and is represented by ##...#.

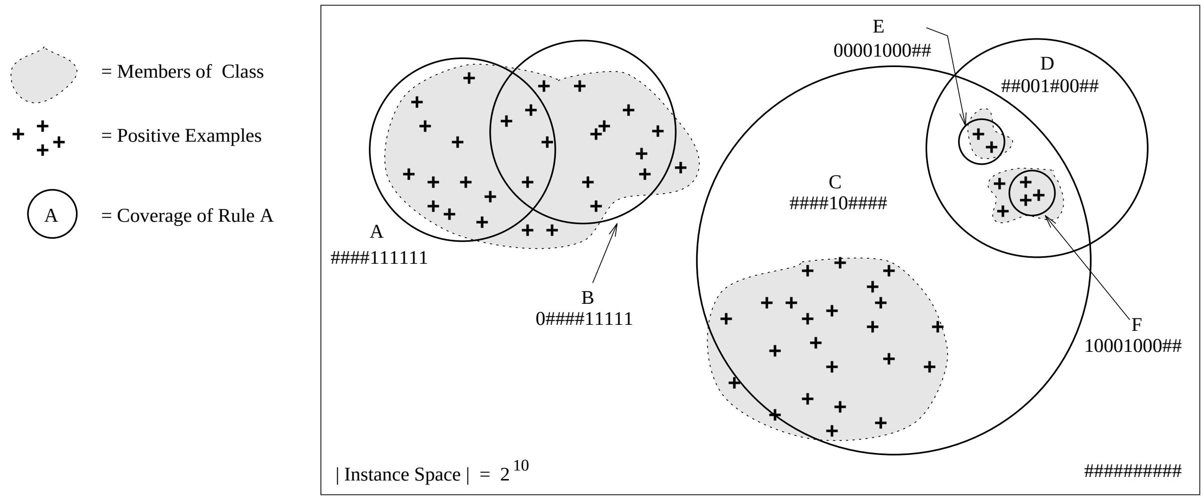

Figure 3.4: Binary classification as a set covering problem. The task of the LCS is to find classifiers (rules, represented as circles) to cover the examples (positive only) of the unknown concept (single class, represented as shaded regions).

We are trying to cover the positive examples with a small set of accurate rules. Figure 3.4 illustrates our version of set covering. The large rectangle represents all possible 2^ℓ instances, where ℓ is the number of bits used to encode each instance. In Figure 3.4, we use $\ell = 10$ bits. The entire instance space is thus described by the rule (########## $\Rightarrow$ 1) or simply ##########. The actual concept to be learned is shown by the shaded regions. The LCS is given only a subset of these instances as (positive) examples. The LCS is constrained to a rules syntax of $\{0, 1, \#\}^{10}$. We represent this constraint by a circle for the "coverage" of a rule. Rules with more # characters cover more of the instance space, and are represented by larger diameter circles. As Figure 3.4 shows, rules can vary in specificity (coverage of instance space) as well as in number of examples[7] covered. Furthermore, rules can overlap to almost any degree in coverage of instances and/or examples.

In general, the binary classification task of the LCS is to find a *small* subset of *accurate* rules that *cover* the examples. These are three separate objectives. However, the second objective, individual rule accuracy, is problematic. How, exactly, do we measure accuracy? Clearly accuracy increases with the number of examples correctly classified and decreases with the number classified incorrectly. But these two numbers represent two conflicting objectives

[7]For the rest of this paper we omit the word "positive", and assume all examples are positive ones.

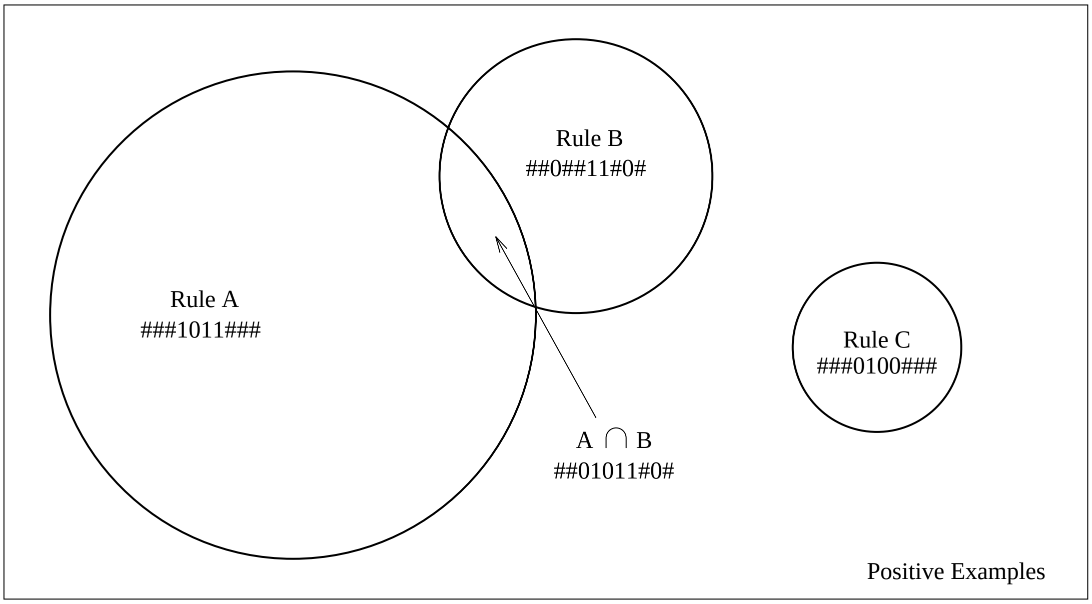

Figure 3.5: By assuming only rules of equal specificity (number of #'s), we can use diameter in the space of examples to indicate a rule's accuracy as "coverage of examples", which is also its *objective* (unshared) fitness.

themselves. One rule might have both a higher number of correct classifications and a higher number of incorrect classifications than another rule. Which is preferred? In addition, we often want rules with greater predictive power, that is, more generality because of their greater coverage of instance space. We might prefer less accurate but more general rules to more accurate, more specific rules. So the objective of rule accuracy is itself multiobjective. It is tied up in the issues of generality-specificity versus accuracy. These are critical issues, but we believe they are separable from the issues of rule set size (conciseness of concept description) and coverage.

To avoid considerations of generality versus specificity, and hence reduce accuracy to a single objective that can be measured by a scalar, we consider only rules of the same order (number of don't cares). All such rules have the same specificity. Since each rule has the same coverage of instance space, their accuracies can be computed as simply the number of examples covered. Maximizing the number of examples covered maximizes accuracy. In terms of Figure 3.4, the LCS is further constrained to using circles of a specific, constant size. This observation prompts us to redefine our Venn diagram terms and use the diameter of the circles to represent accuracy. In Figure 3.5, the large rectangle represents the space of all *examples* given to the LCS for learning. The size of a circle represents the number of *examples* covered by the corresponding rule, and hence its accuracy. The overlaps of circles represent overlaps of coverage among rules, and thus contain the examples "shared" by two or more rules.

3.4.7 A Search Through Schema Space

It is illustrative to take another view of the classification problem at hand. What makes the LCS version of the set covering problem unique is the restriction on the syntax of the concept descriptors (rules) to the ternary alphabet $\{0, 1, \#\}$. This same restriction allows us to recast the covering problem as a search through a schema space. We note that our rules are schemata in the space of instances. By schemata, we mean the similarity templates denoting hyperplanes in the (usually) binary search spaces of GAs (Holland, 1975, 1992). Thus rule ##00#1 in a 5-bit problem would cover all examples with 0's for the third and fourth attributes and a 1 for the last attribute, just as the schema ##00#1 "contains" all strings with 0's in bit positions 3 and 4, and a 1 in position 5. We are choosing, from the set of 3^ℓ possible schemata, a minimal set of highly fit schemata. Figure 3.6 illustrates a portion of an example search space.

In Figure 3.6, the search space is represented as a hierarchy, with each level labeled according to the order (number of fixed bits) of the schemata at that level. A line from one schema to another indicates containment of the lower schema (higher order) by the upper schema (lower order). Note how the intersection (overlap) of one schema with another is itself another schema. At the bottom of the hierarchy are the highest-order schemata, order-ℓ, which are the examples themselves. We can view the binary class membership function as the fitness function over the instance space. Every order-ℓ schema (i.e., instance) has a fitness indicating to what degree it is a member of the class to be learned. In our case, we assume a binary class membership function. Thus our fitness function is binary as well: $f(i) = 0$ if instance i is not a member of the class, and $f(i) = 1$ if it is. More generally, we could use a "fuzzy" class membership function $f(i) \in \{0..1\}$.

The fitness of a rule, or schema, is a function of the number of examples covered (or contained) by the rule, and the number of non-examples (misclassifications) covered or contained. The number of non-examples covered is simply the total coverage of the rule minus the number of examples covered. The total coverage (of instances) of the rule can be calculated from the

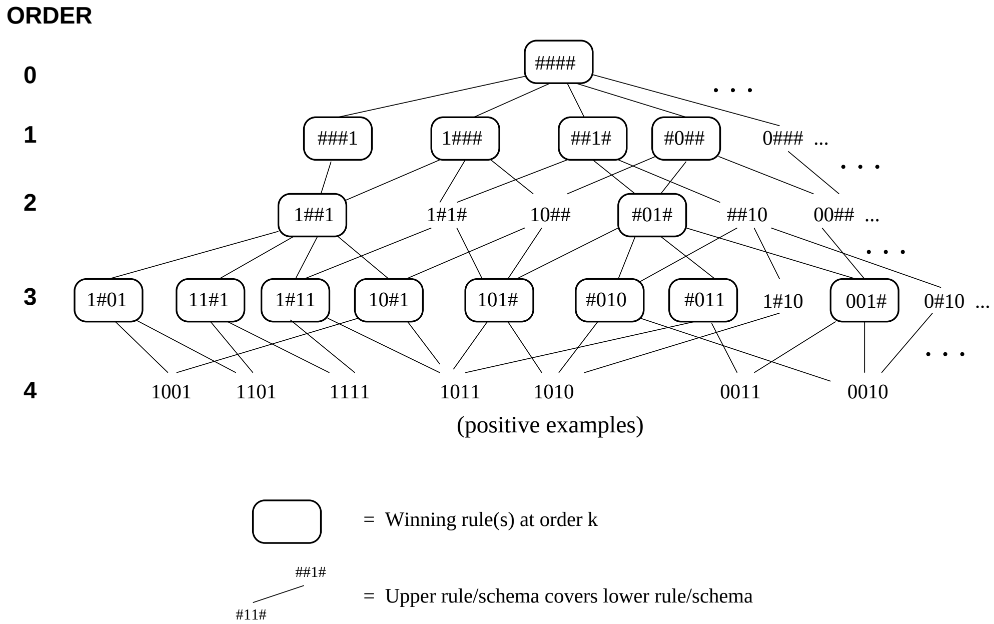

Figure 3.6: The LCS binary classification problem as a search through schema space. Each rule is a schema, or hyperplane in the instance space. Instances, such as examples, are schemata of order ℓ. Here, $\ell = 4$.

order o of the rule: $2^{\ell-o}$. In GA schema analysis, schema average fitness $\overline{f}_{schema}$ is defined as the average fitness of all strings (i.e., instances) contained by a schema. In the case of our binary classification LCS, $\overline{f}_{schema} = |examples|/2^{\ell-o}$. That is, the schema average fitness for schemata corresponding to a rule is the number of examples covered divided by the coverage of the rule. We can extract the number of covered examples from the schema average fitness since we know the order of the schema. We simply multiply $\overline{f}_{schema}$ by $2^{\ell-o}$. This illustrates the close correspondence between the LCS search space and the schemata of the class membership function[8]. A GA is thought to search schema space implicitly, while an LCS clearly searches this space explicitly. However, the goal of the GA is to find the best string (order-ℓ schema) by implicitly processing schemata, while the goal of the LCS is to find the "best" set of schemata, by implicitly processing strings (examples).

Our simplifying restriction to rules of a single order can be viewed in terms of the schema hierarchy as limiting search to one level of the hierarchy. All schemata within a level have the

[8]This quotient, schema average fitness $\overline{f}_{schema}$, is a candidate measure of fitness, although not of much use since some of the least general rules, those that describe only one example each, would be the most fit with a fitness of 1.

same order, hence any comparison of their fitness is essentially a comparison of the number of examples covered. Thus we can leave aside the issue of generality (schema order) versus accuracy, and unambiguously name the winners *at a particular order* k as those rules/schemata with the most examples. Figure 3.6 indicates some of the winners at each order with circles. At order two, for example, the two rules 1##1 and #01# together cover all the examples shown here in our 4-bit problem[9]. Note that these two winners at order two overlap in coverage, competing for the example 1011.

3.4.8 Learning DNF

Finally, we can cast the S-R LCS binary classification task as a well-known problem in machine learning, specifically similarity based learning (SBL) or learning from examples (Michalski, Carbonell, & Mitchell, 1983). Calling the ℓ condition bits binary attributes a_i, $1 \leq i \leq \ell$, the task of our S-R LCS is to learn a (concise) disjunctive normal form (DNF) description of the target concept. Each rule represents a *monomial*, or conjunction of attribute settings (e.g., 1##0 maps to $a_1\bar{a_4}$). The population of rules represents a disjunction of such conjunctions (e.g., $a_1\bar{a_4} + \bar{a_2}a_3a_4 + \bar{a_3} + ...$). The target concept is an arbitrary DNF formula, and the LCS is given a stream (finite or continuous) of examples according to some unknown distribution. If the distribution of the evaluation (or test) set of the LCS is the same as that of the example stream, then we can judge our LCS according to the distribution-free or probably approximately correct (PAC) learning models of Valiant (1984) and others. The schema hierarchy in Figure 3.6 can be seen as a *version space* (or *lattice*, if the empty set is added at level $\ell+1$ as the common "child" of all level ℓ nodes). Our restriction to rules of a single order k is equivalent to a restriction of concept descriptions to k-DNF formulae (i.e., each disjunct has exactly k attributes). Learning k-DNF formulae from positive examples is a known hard problem (Pitt & Valiant, 1988).

3.5 Fitness Sharing versus Resource Sharing

We now consider the relationship between resource sharing and fitness sharing. We will find that they are similar yet distinct. This comparison and contrasting of the two types of shar-

[9] Not only are these two rules individually better than any other order two rule, but they would probably beat most other rules at other orders according to almost any measure of accuracy, since neither rule makes any classification errors.

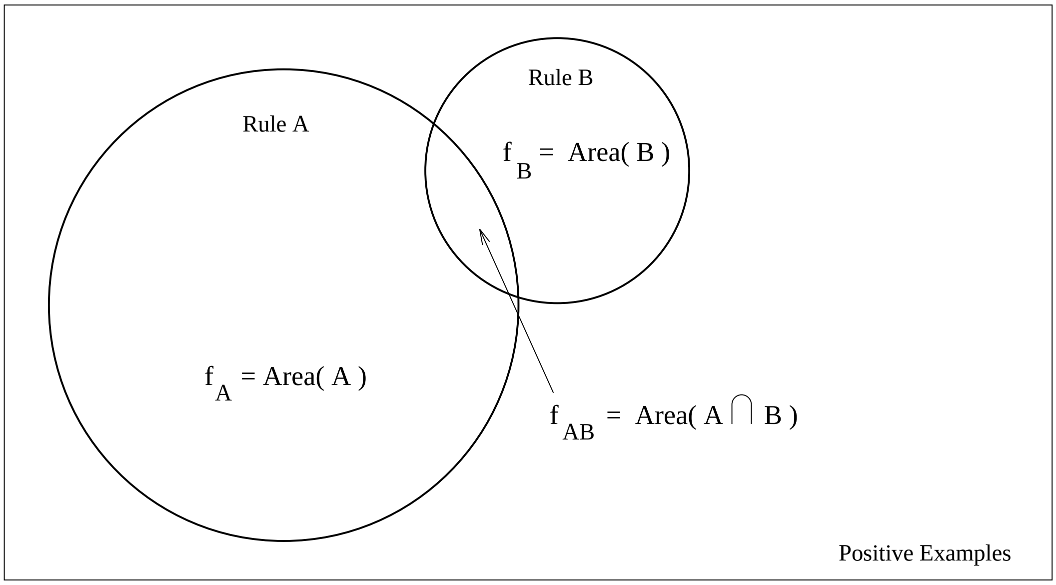

Figure 3.7: Our definitions for the objective fitness of rules, and the "fitness" of the overlap in rule coverage. Fitness might be measured in number of examples covered. Here area is proportionate to fitness.

ing is a constant theme throughout this thesis. To make our discussion of resource sharing more concrete, we use a specific but very common instance of resource sharing: the sharing of examples in a learning classifier system (LCS) discussed above.

3.5.1 Review: Examples as Resources

We briefly review the sharing of examples in the LCS. We focus on the effect two rules have on each other. In fitness sharing, two nearby individuals degrade each other's fitness according to the distance between them, as we described in the background chapter. How do two similar rules (i.e., rules with overlapping coverage) affect each other's fitness in the LCS?

Recall Equations 2.4 and 2.5: let f_A and f_B be the objective fitnesses for rules **A** and **B** respectively. The objective fitness could be taken as the number of examples covered by that rule, in the case of binary classification. Let f_{AB} be the amount of resources in the overlapping coverage of rules **A** and **B**. That is, f_{AB} is the amount of resources shared by **A** and **B** (e.g., the number of examples covered by both). Figure 3.7 illustrates some of these definitions.

Let n_A, n_B be the number of copies of rules **A** and **B**, respectively, in our population. Then we can calculate the shared (expected) fitness of rule **A**:

$$f_{sh,A} = \frac{f_A - f_{AB}}{n_A} + \frac{f_{AB}}{n_A + n_B}. \tag{3.5}$$

And similarly for **B**:

$$f_{sh,B} = \frac{f_B - f_{AB}}{n_B} + \frac{f_{AB}}{n_A + n_B}. \tag{3.6}$$

Next we consider a special case: $f_{AB} = 0$.

3.5.2 No Overlap Means No Difference

We note that in the case of no overlap, $f_{AB} = 0$, the above shared fitnesses reduce to

$$f_{sh,A} = \frac{f_A}{n_A} \qquad f_{sh,B} = \frac{f_B}{n_B}.$$

There are the same expressions as those for two-niche fitness sharing with no overlap. The niche count of an individual **A** under fitness sharing with no overlap is simply the number of copies of that individual. Thus fitness sharing and resource sharing are equivalent when there are no overlapping niches. To put it the other way, the two methods of sharing differ *only* when niches overlap.

3.5.3 Overlapping Niches

First let's also define the fitness ratio $r_f \equiv \frac{f_A}{f_B}$ and the ratio of overlap $r_o \equiv \frac{f_{AB}}{f_A}$. For the moment, let us assume that both rules cover the same number of examples, $f_A = f_B$ and therefore $r_f = 1$. Then the ratio r_o can vary between 0 and 1, as we increase the overlap in coverage. We can now look at how the presence of additional copies of **B** affects the fitness of rule **A**.

3.5.3.1 A Niche Count for Resource Sharing

Recall that in fitness sharing, the shared fitness of an individual **A** is equal to its objective fitness divided by its total niche count m_A:

$$f_{sh,A} = \frac{f_A}{m_A}. \tag{3.7}$$

We can force our expression for shared fitness with resource sharing, in Equation 2.4, into a form similar to the right hand side of Equation 3.7:

$$f_{sh,A} = \frac{f_A}{\frac{f_A n_A (n_A + n_B)}{f_A n_A + f_A n_B - f_{AB} n_B}}. \tag{3.8}$$

Now we have expressed resource sharing as a degradation by division of the objective fitness, as in fitness sharing. But in its current form, the denominator in Equation 3.8, call it D, is difficult to relate to the niche count m_A in fitness sharing. At this point in our analysis, however, we are more interested in **B**'s contribution to the degradation of f_A, than we are in the analog of total niche count for **A**.

3.5.3.2 A Sharing Function For Resource Sharing

In fitness sharing, **B**'s contribution to the denominator (i.e., to niche count m_A) is the term $sh(d_{A,B})$ in the summation $m_A = \sum_{X \in Pop} sh(d_{A,X})$. The total contribution for n_B copies of **B** is thus $n_B * sh(d_{A,B})$. If we take the partial derivative of the denominator with respect to n_B, we get the contribution to the denominator per copy of **B**:

$$\frac{\partial m_A}{\partial n_B} = sh(d_{A,B}). \tag{3.9}$$

Turning to resource sharing, we can also take the partial derivative of the denominator D from Equation 3.8 with respect to n_B:

$$\frac{\partial D}{\partial n_B} = \frac{f_A f_{AB} n_A^2}{(f_{AB} n_B - f_A n_A - f_A n_B)^2}. \tag{3.10}$$

Dividing numerator and denominator by f_A^2 and n_A^2, we get

$$\frac{\partial D}{\partial n_B} = \frac{\frac{f_{AB}}{f_A}}{(\frac{f_{AB}}{f_A}\frac{n_B}{n_A} - \frac{n_B}{n_A} - 1)^2}. \tag{3.11}$$

Substituting $r_o = f_{AB}/f_A$ and defining $r_n \equiv \frac{n_B}{n_A}$, we get

$$\frac{\partial D}{\partial n_B} = \frac{r_o}{(r_o r_n - r_n - 1)^2}. \tag{3.12}$$

Equation 3.12 can be considered the contribution to the "niche count" of **A** under resource sharing. This quantity corresponds to the sharing function $sh()$ of fitness sharing. Comparing the expression in Equation 3.12 to that in Equation 2.2, we see that the "LCS niche count contribution" $\frac{\partial D}{\partial n_B}$ depends on the numbers of rules **A** and **B** in the population, specifically the ratio between those numbers. In fitness sharing, the contribution $sh(d_{A,B})$ depends only on the distance between the two individuals. We can view the degree of overlap, r_o, as the "distance" between two rules in an LCS. When $r_o = 0$ the rules are maximally distant, in that they are outside of each other's niches. At $r_o = 1$, the rules occupy the same niche. Although r_o is clearly not a metric, it serves the same role as the metric in fitness sharing, namely an indicator of similarity of purpose in solving the problem at hand.

3.5.3.3 Comparing Sharing Functions

In Figure 3.8, we plot $\frac{\partial D}{\partial n_B}$ as a function of "distance" $1 - r_o$ for various values of r_n. The curves of Figure 3.8 are strikingly similar to those of the sharing function in Figure 2.1, with the sharing exponent $\alpha_{sh} \leq 1$. Specifically, both $sh()$ and $\frac{\partial D}{\partial n_B}$ are monotonically decreasing functions of niche overlap (or distance) that start at one for complete overlap and decrease steadily to zero for no overlap (i.e., beyond σ_{sh} in the case of fitness sharing). Thus $(1 - r_o)$ in resource sharing seems to correspond directly to the term $(\frac{d_{A,B}}{\sigma_{sh}})^{\alpha_{sh}}$ in fitness sharing.

In fitness sharing, however, $sh()$ is independent of n_A and n_B, while $\frac{\partial D}{\partial n_B}$ varies with both. Holding n_A constant (e.g., at $n_A = 1$), Figure 3.8 shows how $\frac{\partial D}{\partial n_B}$ changes with increasing n_B. At $n_B = 0$, $\frac{\partial D}{\partial n_B}$ looks exactly like the triangular sharing function, which is $sh()$ for $\alpha_{sh} = 1$. This curve represents the contribution to the degradation of $f_{sh,A}$ of the first copy of **B**. As n_B

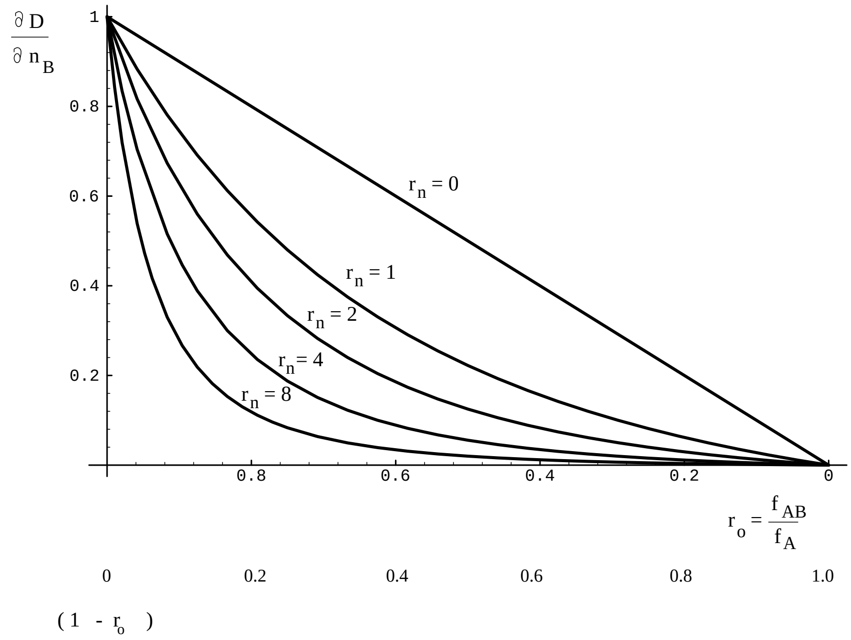

Figure 3.8: The LCS analog of the sharing function $Sh()$, as a function of "distance" $(1 - r_o)$, for different population ratios $r_n = \frac{n_B}{n_A}$. With more copies of **A** than of **B** (smaller r_n), the LCS "sharing" function becomes increasingly triangular.

increases the contribution of each new copy of **B** decreases, and $\frac{\partial D}{\partial n_B}$ as a function of r_o looks like $sh()$ with increasingly small α_{sh}, where $0 < \alpha_{sh} < 1$.

Figure 3.9 plots $\frac{\partial D}{\partial n_B}$ as function of both r_n and r_o. We consider the behavior of $\frac{\partial D}{\partial n_B}$ as depicted in Figure 3.9. Like fitness sharing, resource sharing degrades objective fitness according to how much of a particular resource two individuals have in common. In both forms of sharing, this degradation varies from one (added to the denominator) when two individuals are identical (in their coverage), to zero when both individuals are so different that they have no resources in common. But in resource sharing, the degradation per copy of **B** *decreases* as the number of **B**s grows, at least when the overlap is not complete (i.e., less than 100%). The natural explanation of what is going on here is that **B** can only degrade that portion of **A**'s fitness that comes from the shared resources. Thus when **B** does not completely overlap **A** there is a limit to how much it can degrade **A**'s shared fitness. Each new copy of **B** means that **A** gets an ever decreasing share of the overlapped region. This shrinking share represents an ever decreasing portion of **A**'s total shared fitness $f_{sh,A}$, asymptotically approaching zero. For large n_B, that

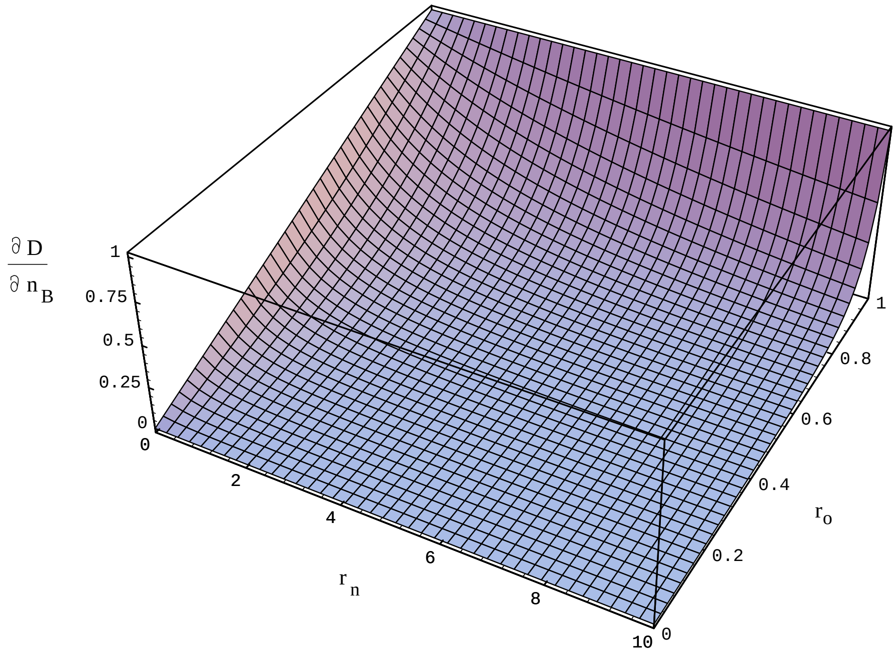

Figure 3.9: The behavior of the resource sharing function is summed up in this plot of population ratio $r_n = \frac{n_B}{n_A}$ versus fitness overlap ratio r_o.

is when $r_n = \frac{n_B}{n_A} >> 1$, the niche represented by the overlap $\mathbf{A} \cap \mathbf{B}$ is essentially filled up, at least from **A**'s point of view, and **A** then relies on other resources (i.e., $\mathbf{A} - (\mathbf{A} \cap \mathbf{B})$).

The above explanation is intuitive and natural. It leads us to speculate that if we want to better preserve individuals with slightly overlapping niches in fitness sharing, we should also limit the amount of fitness degradation caused by a single type of individual as that individual receives many copies. And it seems that a natural way to accomplish this is to decrease the exponent α_{sh} as the nearby competitor multiplies in number.

We can generalize our results so far. For example, what about rules of unequal fitness, $f_A \neq f_B$? Without loss of generality, let us assume $f_B < f_A$, thus $r_f > 1$. Since r_f does not appear in Equation 3.12, the surface in Figure 3.9 does not change. However, the ratio of overlap r_o is now bounded by $\frac{1}{r_f}$, corresponding to total overlap, where the coverage of **A** contains the coverage of **B**. Thus the figures for $\frac{\partial D}{\partial n_B}$ would be cut off at $r_o = \frac{1}{r_f}$ rather than at $r_o = 1$.

3.5.3.4 Further Analysis of the Resource Sharing Function

We can further generalize our results for the behavior of the resource sharing function by allowing for the presence of other rules that partially overlap rules **A** or **B** or both, in coverage. The general effects of the presence of these other rules on the resource sharing function can be summarized qualitatively. Since we are only looking at changes in the sharing function due to changes in the numbers of **A**s and **B**s, we can assume that all other rules, **C, D, ...**, have a constant number of copies. Their effect on $f_{sh,A}$ then is simply to reduce the *original fitness* f_A and f_{AB} to be shared by **A** and **B**. These other rules also reduce the effect of sharing in regions in which they overlap coverage. For example, if the coverage of a rule **C** overlapped by f_{AC} with f_A, then the amount split up solely among copies of **A** would be reduced from $f_A - f_{AB}$ to $f_A - f_{AB} - f_{AC}$. In addition, the degradation of f_{AC} by additional copies of **A** would be spread out among **C**'s shares of f_{AC}, rather than just among **A**'s shares, thus reducing the degradation of $f_{sh,A}$ from what it would be in the absence of **C**.

Although the presence of these other rules in the same niches covered by **A** or **B** does affect the resource sharing function, these effects do not appear to change the overall shape of the function. Furthermore, the effects of other rules on $\frac{\partial D}{\partial n_B}$ are probably less than the effects of **A** and **B** themselves, as a comparison of second-order partial derivatives should bear out (e.g., in the above example of rules **A**, **B**, and **C**, $\frac{\partial^2 D}{\partial n_C \partial n_B} \ll \frac{\partial^2 D}{\partial n_A \partial n_B}, \frac{\partial^2 D}{\partial^2 n_B}$ for the values of n_A, n_B, n_C of interest).

So we concentrate our remaining analysis on the interaction of two rules **A** and **B** in the presence of these two rules only. Looking again at Figure 3.9, we see that the sharing fitness sharing function varies considerably with the ratio r_n of numbers of **A**s to **B**s. But there is one "slice" of this surface, yielding a two-dimensional sharing function plot, that is of particular interest. That slice lies on the curve $r_n = r_{eq}$, where r_{eq} is the *equilibrium point.* In previous studies of niching (Deb, 1987; Horn, 1993), the authors have proposed and investigated such stable points, where the "niching force" and selection force are in balance. Fitness sharing in particular has been shown to induce such an equilibrium (a.k.a. stable or fixed point). Finding an equilibrium distribution of the population, and showing that such a state is indeed a stable one, for both fitness and resource sharing, is the subject of the next chapter.

3.6 Summary

Sharing is not an isolated or anomalous algorithm. Rather it is a fundamental component of some larger contexts, such as the context of natural ecologies. Sharing can also be seen as a first level of cooperation, in a more general model of layered cooperation and competition. In addition, sharing seems to be a very basic type of context-dependent fitness function. In comparing our GAs to ecologies, we find that a simple "global" sharing (of population slots) is already incorporated in our GAs, performing the essential task of keeping our algorithms practical in their use of memory space, in the same way that natural populations are bounded in size. It seems only a small step from there to differentiate multiple diverse resources and so induce multiple cooperative subpopulations (species). This seems the next logical level of complexity as we progress toward more sophisticated "GA ecologies". In addition, more complex types of cooperation, such as mutual aid in resource exploitation, seem to be built upon the more basic type of cooperation involving simple sharing of common resources through localized competition. Finally, we can place sharing in a formal taxonomy of all imaginable fitness functions in which an individual's fitness depends on the context of the current environment and population. Sharing seems to be one of the simplest of all possible such context-dependent functions, and so seems an excellent starting point for the more general analysis of context-dependent function optimization.

Chapter 4

The Existence and Maintenance of an Equilibrium

In this chapter we try to answer the question "Is there really a *niching equilibrium*?", that is, a population distribution that is more or less constant under selection pressure. To answer this question we first examine the *single generation expected behavior*, which is the expected change in population distribution from one generation to the next. After showing that yes, there exists a population distribution $\vec{P}$ such that $E(\vec{P}_t) = E(\vec{P}_{t+1})$, we then look at how meaningful that equilibrium is. More specifically, we ask how steady is such a steady-state population distribution? How stable is such an equilibrium? (To use a classic metaphor, a pencil balanced on its point is at equilibrium, but surely not a stable one!) A simple Markov model will help us answer these questions analytically. After determining the conditions under which niching equilibrium is stable, we move onto the next chapter, in which we examine how GA selection gets to equilibrium, if it ever does. That is, we explore *convergence* to equilibrium.

In this chapter we analyze what might be called the "base case" of niching: two niches. That is, the entire population is composed of two "species": **A** and **B**. The niches occupied by these species can have different fitnesses (resources) and can overlap. We will find that by careful modeling we can understand and predict the general behavior of two-niche populations, under both fitness sharing and resource sharing. It might not be possible to model, understand, and predict three-niche, and $k > 3$-niche situations, nor is it clear to what extent the two-niche models developed and used here will prove useful for general k-niche modeling, but such questions are considered in chapter 6.

4.1 What is Equilibrium?

Perhaps the greatest distinguishing qualification between the niched and the simple GA is the existence of an equilibrium point that consists of a diverse, non-uniform population. By equilibrium, we mean that under selection, there is an active "force" maintaining the population distribution at that point. A small perturbation in the population, away from equilibrium, brings an immediate corrective pressure back toward equilibrium. Thus in a simple GA selection drives the population distribution toward a uniform one consisting of copies of the best (most fit) individual in the current population. Such a uniform distribution is *stable* under selection. It is even stable under recombination and mutation, if we assume only a very small possibility of a better individual being discovered (which seems justifiable, given the lack of genetic diversity at such an equilibrium).

Such an "active" equilibrium, in which forces act to restore equilibrium after perturbation of the system away from it, is known formally as *stable equilibrium*, one of three types of equilibrium:

- stable equilibrium,
- unstable equilibrium, and
- neutral equilibrium.

In contrast, an equilibrium in which a small perturbation leads to larger perturbations, thus driving the system *away* from the equilibrium point, is known as *unstable equilibrium* (e.g., the pencil balanced on its point). Later we will encounter a few examples of unstable equilibria in niched GAs, in very special circumstances. A third kind of equilibrium, *neutral equilibrium* is also seen in GAs.

Genetic drift (that is, the case in which all individuals in the current population have the same objective fitness) has many equilibrium points under selection, but none are stable or unstable. There is no restorative pressure for ANY population distribution. Nor is there any selective pressure *away* from any population distribution.

For niched GAs that use any kind of sharing, resource or fitness, equilibrium (under selection only) will occur when the shared fitnesses of all population members are equal (Goldberg & Richardson, 1987; Deb, 1989). In the two niche case, the general equilibrium condition is:

$$f_{sh,A} = f_{sh,B}. \tag{4.1}$$

Below we substitute into Equation 4.1 the three different formulae for shared fitness, depending on the case: perfect sharing (no niche overlap), fitness sharing (with overlap), and resource sharing (with overlap). We show that a unique steady-state is induced by sharing, allowing us to calculate an equilibrium point as a function of fitness ratio r_f, population size N, and overlap (r_o or σ_{sh}).

4.1.1 Equilibrium under Perfect Sharing

With no niche overlap, individuals of species **A** do not affect the sharing of f_B, and vice versa. The shared fitnesses are simply $f_{sh,A} = \frac{f_A}{n_A}$ and $f_{sh,B} = \frac{f_B}{n_B}$. Substituting these into Equation 4.1 above,

$$\frac{f_A}{n_A} = \frac{f_B}{n_B},$$

we can solve for the *equilibrium ratio* $r_{eq,n} \equiv \frac{n_B}{n_A}$:

$$r_{eq,n} = \frac{n_B}{n_A} = \frac{f_B}{f_A} = \frac{1}{r_f}. \tag{4.2}$$

We can introduce the inverse ratio of fitness here, $r'_f \equiv \frac{f_B}{f_A} = \frac{1}{r_f}$, and rewrite the above as

$$r_{eq,n} = r'_f.$$

(Henceforth we will use either r_f or r'_f in our formulae and graphs based on whichever yields the simplest or clearest expressions or plots. For example, note that since we designated **B** to be the lesser fit niche, $f_B \leq f_A$. Therefore as f_B decreases, $r_f = \frac{f_A}{f_B}$ increases from one to asymptotically approach infinity, while r'_f decreases from one to asymptotically approach zero. For some theoretical models, it might be more convenient to have a parameter with finite bounds, such as r'_f, while for others the left-bounded r_f might be more intuitive.)

From the above equation we can see that at equilibrium two non-overlapping niches will be filled in direct proportion to their fitnesses, as was shown by Goldberg and Deb (1989), and Deb (1989). Thus we might call sharing *fitness proportionate niching* (noting that its "fitness proportionate equilibrium" is independent of the selection method). Also of use later will be the number of **A**s at equilibrium, call this $n_{A,eq}$, and the proportion of **A**s at equilibrium, call it $P_{A,eq} = \frac{n_{A,eq}}{N}$. These can be calculated easily, recalling that $n_B = N - n_A$ in the two-niche case:

$$n_{A,eq} = \frac{r_f}{1 + r_f} N, \tag{4.3}$$

and so

$$P_{A,eq} = \frac{n_{A,eq}}{N} = \frac{r_f}{1 + r_f}. \tag{4.4}$$

Under perfect sharing, an equilibrium always exists, since for any fitness ratio r_f, the proportion $P_{A,eq}$ must be in the range of 0 to 1 ($0 < P_{A,eq} < 1$). But note that for finite populations of size N, $n_{A,eq}$ might not be an integer, in which case n_A will likely fluctuate closely around $n_{A,eq}$.

We now recalculate these quantities ($n_{A,eq}$ and $P_{A,eq}$) under the two different sharing methods: fitness and resource.

4.1.2 Equilibrium under Fitness Sharing

Deb (1989) calculated equilibrium for the more general fitness sharing case (i.e., niche overlap). Again, substituting shared fitnesses into Equation 4.1 above,

$$\frac{f_A}{m_A} = \frac{f_B}{m_B}.$$

Recall the calculation of niche counts: $m_i = \sum_{j \in pop} Sh(i,j) = \sum_{j \in pop}(1 - d(i,j)/\sigma_{sh})$. (We simplify our calculations by assuming $\alpha_{sh} = 1$, yielding the common triangular sharing function.) For our two niches **A** and **B**, let us set $d(A,B) = 1$, without loss of generality. Since $n_A + n_B = N$, the niche count of **A** becomes

$$m_A = n_A * Sh(A,A) + n_B * Sh(A,B).$$

Now since $d(A, A) = 0$, then $Sh(A, A) = 1$, while $Sh(A, B) = 1 - \frac{1}{\sigma_{sh}}$. Thus the niche count becomes

$$m_A = n_A + (N - n_A)(1 - \frac{1}{\sigma_{sh}}).$$

Similarly for **B**,

$$m_B = (N - n_A) + n_A(1 - \frac{1}{\sigma_{sh}}).$$

Substituting m_A and m_B into Equation 4.1 above, yields

$$\frac{f_A}{n_A + (N - n_A)(1 - \frac{1}{\sigma_{sh}})} = \frac{f_B}{(N - n_A) + n_A(1 - \frac{1}{\sigma_{sh}})}. \tag{4.5}$$

Solving for n_A and calling it $n_{A,eq}$ for "n_A at equilibrium", results in

$$n_{A,eq} = \frac{r_f \sigma_{sh} - \sigma_{sh} + 1}{1 + r_f} N, \tag{4.6}$$

and in

$$P_{A,eq} = \frac{r_f \sigma_{sh} - \sigma_{sh} + 1}{1 + r_f}. \tag{4.7}$$

Note that when niche overlap disappears, $\sigma_{sh} = d(A, B) = 1$, and the above equations reduce to those of perfect sharing above (Equations 4.3 and 4.4). As Deb (1989) and Deb and Goldberg (1989) note, this equilibrium exists only for a limited range of overlap σ_{sh}. For example, if we assume that $f_A > f_B$, which implies $r_f > 1$, then $P_{A,eq}$ must be strictly less than 1 for all σ_{sh} such that $1 \leq \sigma_{sh} \leq \frac{r_f}{r_f - 1}$. And when $\sigma_{sh} = \frac{r_f}{r_f - 1}$, then the above equation yields $P_{A,eq} = 1$, so if $\sigma_{sh} > \frac{r_f}{r_f - 1}$ then $P_{A,eq} > 1$, which is impossible. Thus there exists a critical overlap, a function of fitness ratio r_f, at which point niching collapses (because the equilibrium point disappears).

4.1.3 Equilibrium under Resource Sharing

We calculate an equilibrium point for resource sharing as a function of r_f, r_o, and N. Again we start with the general equilibrium condition, Equation 4.1, in which shared fitnesses are equal:

$$f_{sh,A} = f_{sh,B}. \tag{4.8}$$

Substituting the formulae for shared fitness under resource sharing, from Equations 2.4 and 2.5 in Chapter 2,

$$\frac{f_A - f_{AB}}{n_A} + \frac{f_{AB}}{n_A + n_B} = \frac{f_B - f_{AB}}{n_B} + \frac{f_{AB}}{n_A + n_B}.$$

Remembering that $n_A + n_B = N$, we reduce the above to

$$\frac{f_A - f_{AB}}{n_A} + \frac{f_{AB}}{N} = \frac{f_B - f_{AB}}{n_B} + \frac{f_{AB}}{N}.$$

The common term $\frac{f_{AB}}{N}$ cancels. Solving for $\frac{n_B}{n_A}$,

$$\frac{n_B}{n_A} = \frac{f_B - f_{AB}}{f_A - f_{AB}} = \frac{\frac{f_B}{f_A} - \frac{f_{AB}}{f_A}}{1 - \frac{f_{AB}}{f_A}}. \tag{4.9}$$

Substituting our ratios $r'_f \equiv \frac{f_B}{f_A}$ and $r_o \equiv \frac{f_{AB}}{f_A}$, rearranging, and naming the result $r_{eq,n} \equiv \frac{n_B}{n_A}$ (at equilibrium),

$$r_{eq,n} = \frac{r'_f - r_o}{1 - r_o}. \tag{4.10}$$

From Equation 4.10 we see that the equilibrium point changes with overlap, just as with fitness sharing. With no overlap, that is when $r_o = 0$, then $r_{eq,n} = r'_f$. And at complete overlap ($r_o = r'_f$) the equilibrium point is 0 (i.e., the population will converge to a uniform one, consisting of all **A**s).

Again, let us extract from the equation for $r_{eq,n}$ the actual *number* and *proportion* of **A**'s in the population, n_A and $P_A = \frac{n_A}{N}$, which we will need later:

$$r_{eq,n} = \frac{n_B}{n_A} = \frac{N - n_A}{n_A} = \frac{1 - P_A}{P_A} = \frac{r'_f - r_o}{1 - r_o}.$$

Solving for n_A and calling it $n_{A,eq}$:

$$n_{A,eq} = \frac{1 - r_o}{1 - 2r_o + r'_f} N. \tag{4.11}$$

This is the number of copies of **A** we would expect to find at equilibrium, given a particular ratio of fitness r'_f and ratio of overlap r_o, and a fixed population size N. Dividing both sides by N gives P_A, which we'll call $P_{A,eq}$:

$$P_{A,eq} = \frac{1 - r_o}{1 - 2r_o + r'_f}. \tag{4.12}$$

This is the *proportion* of **A**s we would expect to find at equilibrium (and is independent of population size N). Both $n_{A,eq}$ and $P_{A,eq} = \frac{n_{A,eq}}{N}$ are simply *predictions* describing the expected equilibrium based solely on the assumed condition that at equilibrium all (shared) fitnesses, of all individuals, are equal. This simple model of equilibrium does not consider any other complex, dynamic aspects of selection, such as the noise in stochastic selection methods like RWS, or the changes in niche counts over time (generations). In the next section we investigate how well this straightforward, simple model of equilibrium predicts the actual situation at equilibrium. To model the time-dependent (i.e., dynamic) aspects of sharing and selection, we turn to Markov chains. (Later, in Chapter 6, we will examine how even the simple, straightforward calculation of expected numbers/proportions from the equilibrium condition, as presented in this section, can become a complex, and even difficult calculation, when multiple (≥ 3) niches are considered under resource sharing.)

4.2 Markov Models of Sharing

In this section, we show how previous work has used Markov chains to model the simple GA. We then modify those chains to model niched (sharing) GAs. We examine how these models differ, reflecting the different sharing algorithms and assumptions being modeled. In the following section we will use these models to rigorously answer questions about the stability and maintenance of sharing equilibrium.

4.2.1 Markov Models of Simple GAs

Modeling the simple GA as a Markov chain has been successful in understanding several basic behaviors (e.g., De Jong, 1975; Goldberg & Segrest, 1987; Vose, 1990, 1992; Nix & Vose, 1992; Davis & Principe, 1993; Ngo & Marks, 1993; Suzuki, 1993). In particular, modeling the simple GA as a finite Markov chain (Goldberg and Segrest, 1987) leads to exact expressions for expected drift times, times to convergence, and probabilities of premature convergence. These results are exact for single allele chromosomes only, but provide bounds on expectation for more realistic problem spaces. Markov chains can exactly model each generation of a GA by combining the effects of the various stochastic sources (such as initial population generation, selection, crossover, mutation, and even noisy fitness functions) into a single transition probability matrix. To date, the few studies that have used Markov modeling have necessarily kept to artificially small population sizes and string lengths (e.g., Goldberg & Segrest, 1987), or else have worked with matrix notation only, avoiding the generation and direct manipulation of the matrices themselves (e.g., Nix & Vose, 1992). We discuss the Goldberg and Segrest (1987) model in some detail, as it is their model for a simple GA that we extend to include niching.

4.2.1.1 The Goldberg and Segrest Model

Goldberg and Segrest (1987) kept their model manageable by dealing with only a "single-locus" genome. That is, the genome consists of only one binary position. Thus the only possible individuals are "1" and "0" (i.e., the species **A** and **B** for us). Such a limitation allows for an intuitive numbering of states. Given a fixed population size of N, there are $(N+1)$ possible states i, where i is the population with exactly i ones and $(N-i)$ zeros. Goldberg and Segrest defined an $(N+1)$ by $(N+1)$ transition matrix $P_{trans}[i,j]$ mapping the current state i to the next state j.

We repeat here Goldberg and Segrest's calculation of the transition probabilities for their model of a single-locus, simple GA using proportionate selection with no mutation. However, we substitute the names **A** and **B** for the two distinct possible individuals, rather than their "1" and "0" respectively. This substitution is in keeping with our emphasis on the general case of two species, rather than the case of a single-locus genome. Thus i now means the number of copies of **A** in the current population, and $N-i$ is the number of copies of **B**, and similarly for j and the next generation:

$$i \equiv n_{A,t}, \qquad j \equiv n_{A,t+1}.$$

Under proportionate selection, we choose a member k of the current population to reproduce (i.e., to be in the next population) with probability proportional to its fitness relative to the

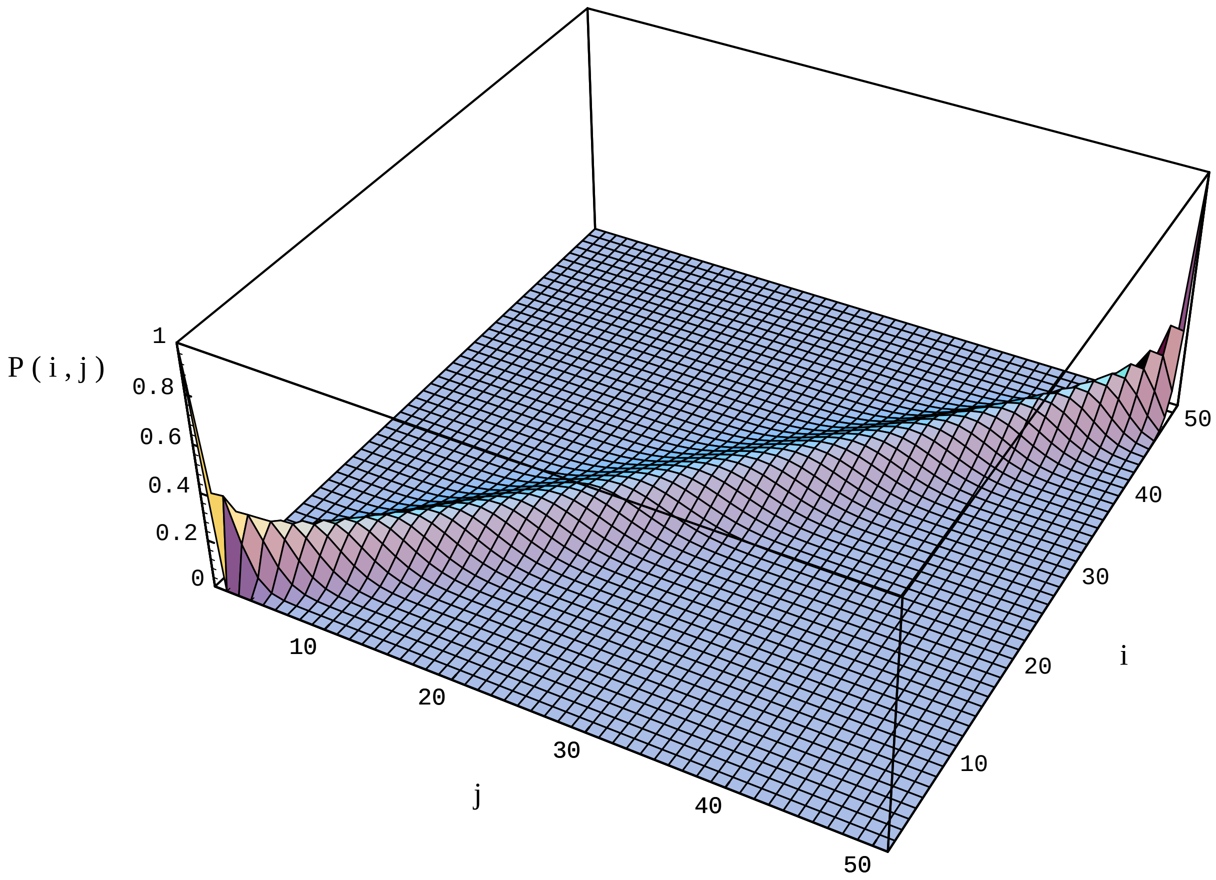

Figure 4.1: The transition matrix for a simple GA with $N = 50$ and $r_f = 1$ (genetic drift). $P_{trans}[i, j]$ is the probability of transiting from current state i to next state j in a single generation. Note the absorbing states $i = 0$, 50.

total fitness of the population[1]. Thus, with probability

$$p_A = \frac{f_A}{\sum_{i=1..N} f},$$

we choose an individual **A** (where f_A is the fitness of **A** and $\sum f$ is the sum of the fitnesses of all individuals in the current population).

If there are n_A exact copies of an individual **A**, then the probability of an **A** (any **A**) getting selected is

$$p_A = \frac{n_A f_A}{\sum f}.$$

In a population with only two distinct types of individuals (i.e., two niches) **A** and **B**, with $n_A + n_B = N$, we can write the denominator $\sum f$ as simply $n_A f_A + n_B f_B$. Then the probability of choosing an **A** for the next generation's population is

$$p_A = \frac{n_A f_A}{n_A f_A + n_B f_B}. \tag{4.13}$$

Similarly for p_B:

$$p_B = \frac{n_B f_B}{n_A f_A + n_B f_B}. \tag{4.14}$$

With r_f as the fitness ratio $\frac{f_A}{f_B}$, then $p_A = \frac{r_f n_A}{r_f n_A + (N - n_A)}$. Letting i be $n_{A,t}$, then $p_A = \frac{i r_f}{i r_f + (N-i)}$. The probability of choosing a **B**, p_B is of course $1 - p_A$ or $p_B = \frac{N-i}{i r_f + (N-i)}$. The probability of going from a state with i **A**s to a state with j **A**s is

$$P_{trans}[i,j] = \binom{N}{j} (p_A)^j (p_B)^{N-j}. \tag{4.15}$$

We can substitute for p_A and p_B:

$$P_{trans}[i,j] = \binom{N}{j} \left(\frac{i r_f}{i r_f + (N-i)} \right)^j \left(\frac{N-i}{i r_f + (N-i)} \right)^{N-j}. \tag{4.16}$$

Equation 4.16 defines a complete transition matrix (and hence a complete Markov chain/process) for any population size N and fitness ratio r_f. In Figure 4.1 we plot the transition probabilities

[1]Remember that in a *generational* GA, such as the simple GA, we make N selections each generation to replace the entire population at time t with a new (but similar) population at time $t+1$.

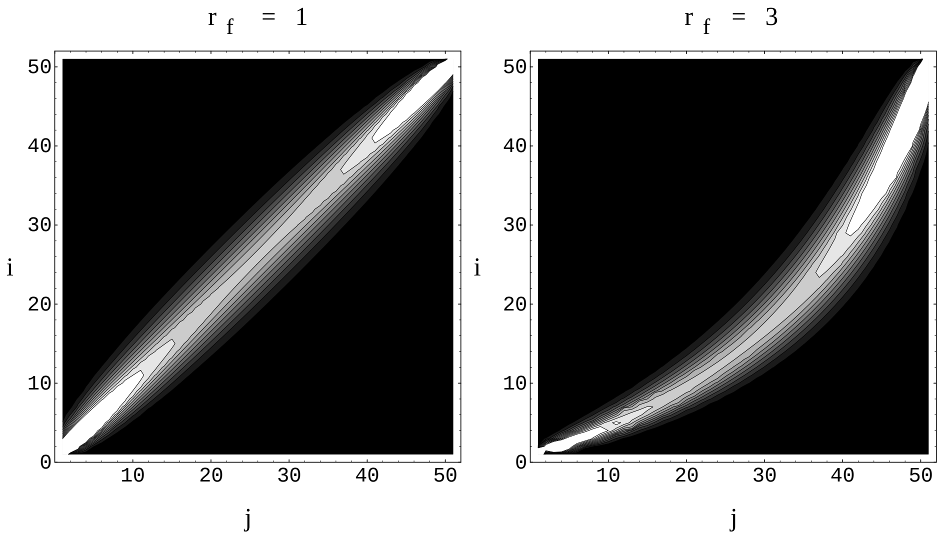

Figure 4.2: Contour plots of transition matrices for the simple GA with $N = 50$ and $r_f = 1, 3$. On the left is a contour plot of the surface in Figure 4.1, showing the case of genetic drift. On the right is a matrix illustrating selection pressure.

for a population of size $N = 20$, and a fitness ratio of $r_f = 1$. On the left of Figure 4.2 is a contour plot of the surface plot in Figure 4.1. The centering of the distributions on the main diagonal is due to the fact that $r_f = 1$. There is no selection pressure toward the all-**A**s or all-**B**s states. With $r_f = 1$, Equation 4.16 reduces to the equation for pure *genetic drift*, a major focus of the Goldberg and Segrest study.

An important feature to note in Goldberg and Segrest's model of selection pressure alone is that the two states 0 and N, corresponding to all-**B**s and all-**A**s respectively, are absorbing states, and thus have transition probability rows and columns consisting of a single spike of probability one ($P_{trans}[i, i] = 1$). Goldberg and Segrest used Equation 4.16 to investigate expected times to absorption for the drift case ($r_f = 1$).

We are interested in visualizing the force of selection when there is a preference ($r_f \neq 1$), before we add niching pressure. Figure 4.2, right, shows the transition matrix for $r_f = 3$. The "ridge" of higher probabilities moves off the main diagonal when $r_f \neq 1$, thus favoring the higher fit individual. The presence of the ridge in the lower or upper triangles of the matrix indicates a pressure toward more or less **A**s, respectively. Comparing Figure 4.2 to the phase diagram for resource depletion model of Roughgarden, in Figure 3.1, we see that the expected behavior of the fixed population simple GA is essentially the same as that of a variable size population model under resource sharing[2]. Thus we argue that the fixed population size of GAs already places us in the regime of resource sharing, even when discussing a simple GA (without explicit sharing.).

4.2.1.2 Other Markov Models of Simple GAs

Two recent papers extend the Goldberg and Segrest model by allowing genome sizes (string lengths) greater than one (Davis & Principe, 1991; Nix & Vose, 1992). With string length ℓ, the number of possible binary strings is 2^ℓ, and the number of possible states (distributions of a population of size N over such a 2^ℓ gene space) is $\frac{(N+2^\ell-1)!}{N!(2^\ell-1)!}$ (Nix & Vose, 1992). This number grows polynomially in N and exponentially in ℓ, so that the transition matrix for a realistic GA implementation could not be generated, let alone manipulated and analyzed. However, Nix and Vose (1992) work with the matrix notation directly, rather than generating the actual probabilities. By assuming infinite population sizes and/or genome lengths, they describe asymptotic behaviors for a simple GA. Similarly, Davis and Principe were able to develop the outlines of a theoretical proof of convergence for a simple GA.

4.2.1.3 Other Markov Models of Niched GAs

Mahfoud (1991) adopts the extended models of Nix and Vose, with some simplifying assumptions, such as partitioning the 2^ℓ gene space into a much smaller number of equivalence classes of similar individuals. Mahfoud then analyzed the selection method known as Boltzmann tournament selection, which achieves some degree of niching. In 1993, Mahfoud extended his Markov analyses of niched GAs to fitness sharing, extending the work of Goldberg and Segrest (1987) by looking at multiple, non-overlapping niches (i.e., perfect sharing). Simultaneously, Horn (1993) also extended the Goldberg and Segrest model to fitness sharing but concentrated instead on the case of overlapping niches, and limiting himself to only the two-niche case. (The results of (Horn, 1993) are included in this chapter.) For both Horn and Mahfoud, the primary focus

[2]Although we are comparing a Markov chain's transition matrix to a phase diagram here, We would expect to the same curve in the phase diagram of the fixed population simple GA, were we to plot it. The bowed ridge of higher probability in the transition matrix makes such a prediction for the phase diagram quite intuitive.

was on niche maintenance (e.g., expected time to loss of a niche). Horn in particular calculated expected steady-states (i.e., equilibrium population distributions) and illustrated the existence of a "niching force" or "restorative pressure" in a niched GA. (Again, this is the work presented below.)

4.2.1.4 Limitations of Markov Modeling of GAs

The primary advantage of Markov modeling over other mother modeling methods is its completeness; the Markov models we have discussed account for all stochastic effects of all GA operators and track all GA dynamics. They are therefore *exact* models. As Vose has stated (Vose, 1995), there is really no distinction between the Markov chain and the GA it models.

Yet Markov analysis suffers from two major drawbacks: (1) intractability (regarding the *cost* of modeling), and (2) focus on asymptotic behavior. The first limitation we have mentioned above: the number of states in the chain grows polynomially in population size N, exponentially in chromosome length ℓ, and also polynomially in c, the number of different *classes* of individuals (or niches) being tracked (to use Mahfoud's (1995a,b) terminology). The size of the transition matrix is always S by S, or S^2, where S is the number of states. Calculation of many Markov properties, such as steady-state distributions, involve multiple matrix multiplications and inversions, making such exact modeling of GAs intractable. In addition, the transition matrix is specific to a particular problem (or function) being solved (optimized), so that the entire Markov analysis can be considered highly problem-dependent.

The second major limitation of the Markov approach is the emphasis placed on asymptotic behavior. Of course, Markov chains track transient behavior (i.e., at times $t = 0, 1, 2, 3, 4, 5,...$) exactly. But most types of *summarized* Markov analysis (e.g., those providing nice, neat scalar measures) are concerned with arbitrarily long-term behavior, as $t \rightarrow \infty$. For example, the long-term steady-state distribution calculation for a Markov chain model of a GA with mutation only (no selection or crossover) tells us that the GA will eventually find the global optimum (because it will eventually visit every point in the search space!). The same calculation for a GA with selection, crossover, but no mutation will show that the GA might not ever visit the global optimum. Such asymptotic analyses can be misleading, since they explicitly avoid any consideration of the actual time involved. Thus the result that mutation will cause the GA to visit every point in the search space (including the global optimum) is of no practical interest, since the amount of time required to do can be arbitrarily long (though finite). So many of the standard, textbook Markov analyses are of little or no use to GA modelers, but the temptation to apply them can be great once the chain is set up.

We try to avoid the limitations of Markov chain analysis in the rest of this chapter, discussing the drawbacks and strengths of particular analyses where appropriate. For example, we note that the Markov chain that is an *exact* model of the single-locus genome ($\ell = 1$) can be considered an approximate model of the general $\ell \geq 1$ case in which only two viable species (niches) are in the starting $t = 0$ population[3]. As another example, we force the asymptotic analysis of steady-state distributions to tell us about transient behavior in an absorbing Markov chain by first changing the absorbing Markov chain into an ergodic one. So the Markov chains can be used, with care, to give some meaningful, general, scalable results for GA dynamic behavior analysis.

4.2.2 A Markov Model of Perfect Sharing (Non-overlapping Niches)

To add in the effect of sharing, we must degrade each of the two fitnesses, f_A and f_B, by the existence of nearby individuals. We begin with the simple case of *perfect sharing*, such that the niches centered on each kind of individual do not overlap[4]. We choose the word "perfect" here for two reasons: (1) it reminds us of the term *perfect discrimination* which is used in the ecological literature to describe completely separate, independent species, and (2) both the GA dynamics and the analysis of the dynamics are greatly simplified by the lack of overlap, leading to some startlingly simple, clear results as will be shown again and again in later analysis. Perfect sharing is also remarkable because it is a special case of both fitness and resource sharing. That is, without overlap, fitness and resource sharing are exactly the same algorithm. To put it another way, the only time the two sharing methods differ is when there is overlap. To play with words, we might say that the "overlap" between the two methods is the case of *no* overlap!

For *perfect sharing* $\sigma_{sh} \leq 1$, so that **A**s do not degrade f_B and **B**s do not degrade f_A. And with $\sigma_{sh} \leq 1$, we are no longer concerned with α_{sh}, since for any setting of α_{sh}, the

[3]Mahfoud (1995a,b) calls such species/niches *equivalence classes*.

[4]The simple biological analog is the case of isolated niches filled with identical competitors evenly split a niche's resource among themselves.

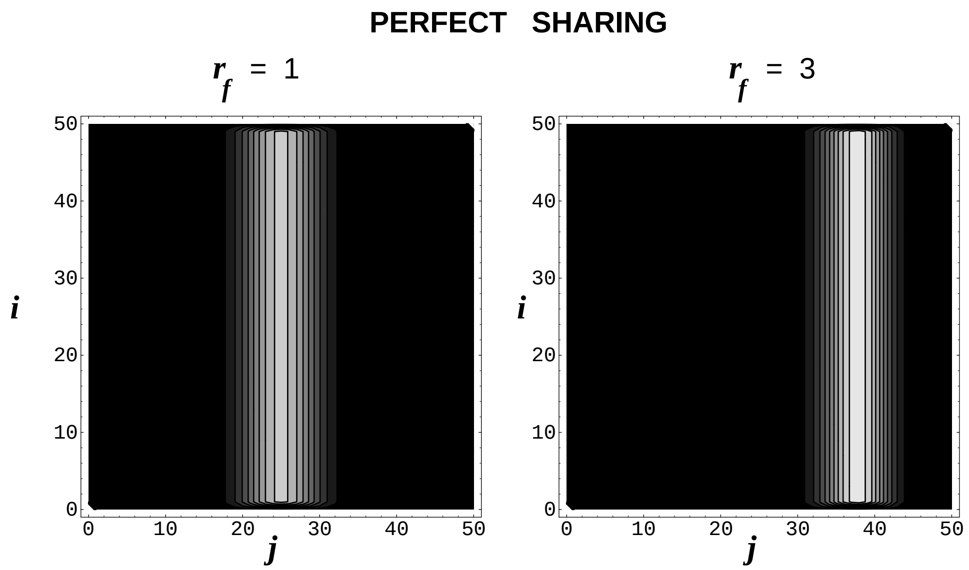

Figure 4.3: Transition matrices for perfect sharing ($\sigma_{sh} \leq 1$).

contribution of an individual to its own niche count is always 1. For *perfect resource sharing*, $f_{AB} = 0$ therefore $r_o = 0$.

For perfect sharing in general then, in the two niche case **A** and **B**, the shared fitnesses are simply $f_{A,sh} = \frac{f_A}{n_A}$, and $\frac{f_B}{n_B}$. If we substitute these shared fitness values $f_{A,sh}, f_{B,sh}$ for the objective fitness values f_A and f_B respectively in Equation 4.15, we obtain the transition probabilities[5]:

$$P_{trans}[i,j] = \binom{N}{j} * \left(\frac{r_f}{r_f+1}\right)^j * \left(\frac{1}{r_f+1}\right)^{N-j}. \tag{4.17}$$

Let us see how perfect sharing has affected our transition probability matrix. On the left of Figure 4.3 is the transition probability matrix for a GA with perfect sharing, population size $N = 50$, and fitness ratio $r_f = 1$. Interestingly, perfect fitness sharing exactly counters the benefits of quantity bestowed by proportionate selection. Each additional copy of a **B** in the population, for example, degrades f_0 but also increases the probability of selection under proportionate selection, resulting in the elimination of the current state i from the transition

[5]Note that Equation 4.15 cannot be used where $i = 0$ or $i = N$, since one of the quotients $\frac{f_A}{i}$ or $\frac{f_0}{N-i}$ does not exist there. But since these are absorbing states, we know their row of transition probabilities.

probability equation. So the row independence of the matrix for fitness sharing is not due to the removal of a complex force, but rather the addition of a new one (niching) that balances the old (selection/drift). For our simple model at least, perfect sharing combined with proportionate selection yields an ideal restorative force.

From Figure 4.3 left, we can say that in general, the effect of the niching pressure is to move the population toward the center states, where the numbers of **A**s and **B**s are balanced. For example, when there are more **A**s than **B**s we are likely to move towards states of lower numbers of **A**s. Thus we can interpret the niching force as being a stabilizing one, with a point of equilibrium far from the poles toward which selection and drift drive us. The apparent point of equilibrium is at the state $i = 10$, as we expect given that $r_f = 1$. What about for other r_f values? At the right of Figure 4.3 is the matrix for $r_f = 3$. Note how the peak of the probability ridge is where Equation 4.6 predicts, at $i = n_{A,eq} = 15$. Later, we will investigate this potential empirical confirmation of niched GA theory.

4.2.3 A Markov Model of Fitness Sharing (Overlapping Niches)

It is often the case that niches overlap. In fitness sharing, we easily might estimate σ_{sh} to be too large, sometimes on purpose (Goldberg, Deb, and Horn, 1992). In niched GAs and in life, while two species compete to fill a niche, the niches centered on them overlap. In this section, we examine the general case where $\sigma_{sh} > 1$.

The niche count for **A**s, m_A, is now $n_A + (N - n_A)(1 - \frac{1}{\sigma_{sh}})$ instead of just n_A, since we have to add in the share value of each copy of **B**: $1 - \frac{1}{\sigma_{sh}}$. Similarly, $m_B = (N - n_A) + n_A(1 - \frac{1}{\sigma_{sh}})$. Substituting the new degraded fitnesses f_A/m_A and f_B/m_B into the equations for p_A and p_B, dividing through by f_B to obtain r_f, and after some rearranging, we obtain the new transition probability equation[6]:

$$P_{trans}[i,j] = \binom{N}{j} \left(\frac{ir_f}{ir_f + (N-i)\frac{m_A}{m_B}} \right)^j \left(\frac{(N-i)}{ir_f\frac{m_B}{m_A} + (N-i)} \right)^{N-j}. \qquad (4.18)$$

[6]For now we assume that $\alpha_{sh} = 1$. For the single-locus model, varying α_{sh} has the same effect as varying σ_{sh}, since $(\frac{d_{AB}}{\sigma_{sh}})^{\alpha}_{sh} = \frac{1}{\sigma_{sh}^{\alpha_{sh}}}$ when $d_{AB} = 1$.

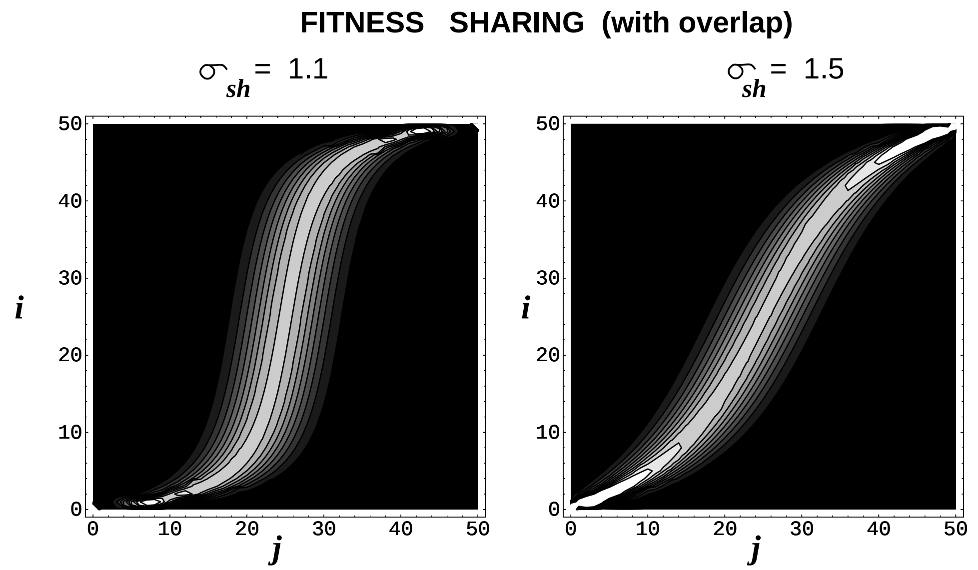

Figure 4.4: Transition matrices for fitness sharing with overlapping niches.

Compare our Markov chain for fitness sharing with that of perfect sharing (Equation 4.17). Note that when there is no niche overlap ($\frac{d}{\sigma_{sh}} = 1 \Rightarrow \frac{m_A}{m_B} = \frac{i}{N-i}, \frac{m_B}{m_A} = \frac{N-i}{i}$), the Markov chain (i.e., transitions) for fitness sharing reduces to that of perfect sharing. In Figure 4.4 we show the transition matrices resulting from Equation 4.18 for a population size $N = 50$, fitness ratio $r_f = 1$, and two different degrees of overlap. For very small overlaps ($\sigma_{sh} \approx 1$, such as $\sigma_{sh} = 1.1$ shown on the left of Figure 4.4), the niching pressure dominates and the matrix is very similar to that of perfect sharing. As the overlap increases (e.g., to $\sigma_{sh} = 1.5$ on the right of Figure 4.4), the effect of niching is diminished, most especially near the absorbing states, and genetic drift comes to dominate. For $\sigma_{sh} = 3$, our matrix (not shown) appears not significantly different from the case of pure genetic drift. This gradual transition from ideal niching to genetic drift agrees with intuition. When $\sigma_{sh} = 1$, Equation 4.18 reduces to Equation 4.17 for perfect sharing. As $\sigma_{sh} \to \infty$, the entire search space becomes a single niche, and we approach the case of genetic drift (or selection pressure, when $r_f \neq 1$).

If the effect of fitness sharing degrades gracefully with increasing error in σ_{sh} (overestimation), does the stability of our assumed steady-state also decay gracefully? To our list of questions about the nature of the supposed equilibrium, we can add this question of how the steady-state degenerates as we move away from the ideal niching situation. Such questions are addressed later in this chapter. For now, we expand our Markov model of fitness sharing to two other types of selection different from roulette wheel: stochastic remainder selection and binary tournament selection. These cases are of interest because their Markov models reveal similar restorative pressures, but some radically different behaviors.

4.2.3.1 Stochastic Remainder Selection

Stochastic remainder selection (SRS) (Goldberg, 1989; Goldberg & Deb, 1991) is a "less noisy" form of proportionate selection than is roulette wheel. Like roulette wheel, for each individual an expected number of offspring is calculated based on the individual's fitness relative to the average fitness. But under SRS, the integer portion of this expected number is automatically allocated, and only the fractional part is subjected to a stochastic "spin of the wheel". Thus if individual k has fitness 3 and the population's average fitness is 1.4, then the expected number of k's offspring would be $3/1.4 \approx 2.14286$. SRS would give k two offspring for sure, plus another "0.14286" in expectation.

Substituting SRS for RWS in our transition probabilities, with fitness sharing, leads to some interesting transition matrices. For example, Figure 4.5 shows a transition matrix for perfect sharing (no overlap), $N = 50$, and $r_f = 1$. Note the spikes of probability one. Such rows represent current populations in which the expected number of offspring for each copy of $\mathbf{A}$ is an integer. Thus the transition to the next generation is not stochastic at all, in such cases. Note the middle spike. Here one of the "singular rows" (i.e., rows with a single non-zero transition probability) has its spike right on the identity line (i.e., the next state is the same as the current state). Thus this spike, at the state with 25 copies of $\mathbf{A}$ and 25 copies of $\mathbf{B}$, is an absorbing state. It also happens to be the equilibrium point for this case. Thus fitness sharing can indeed have absorbing equilibrium states, depending on the "stochasticity" of the selection mechanism.

4.2.3.2 Naive Tournament Sharing

Another interesting phenomenon of fitness sharing is revealed by the use of binary tournament selection in place of RWS. As Oei, Goldberg, and Chang (1992) made clear, the simple (or *naive*) combination of fitness sharing and tournament selection induces undesirable, chaotic

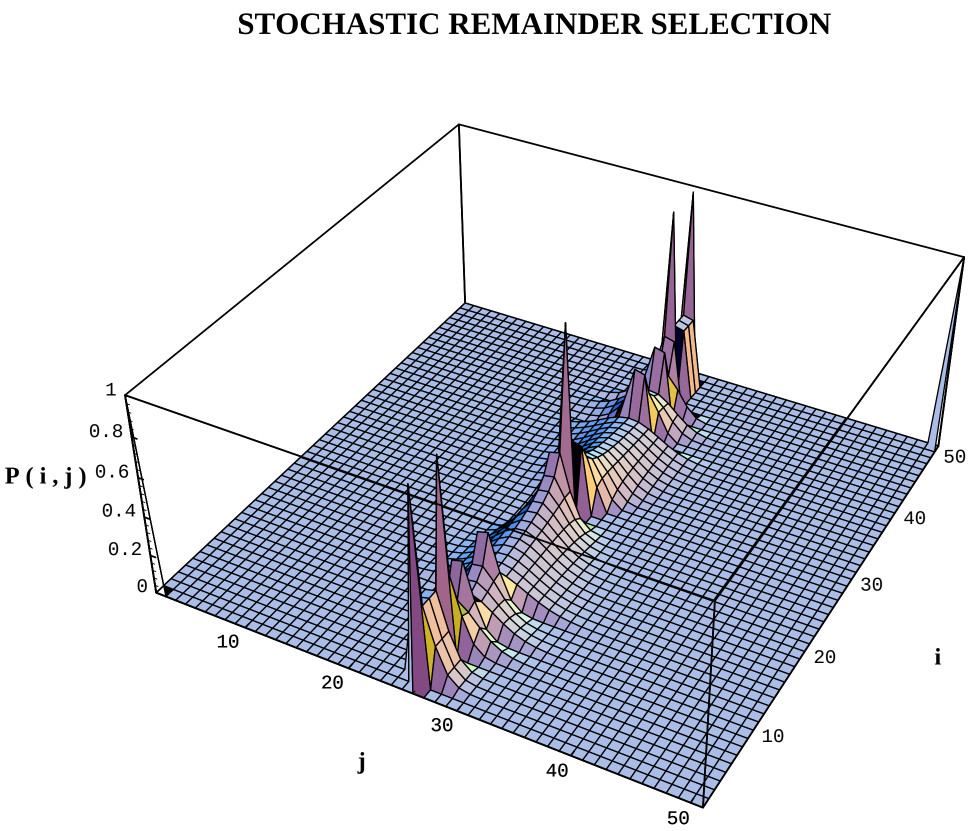

Figure 4.5: Under stochastic remainder selection, and other such deterministic selection methods, it is possible to have a third absorbing state, this one at sharing equilibrium.

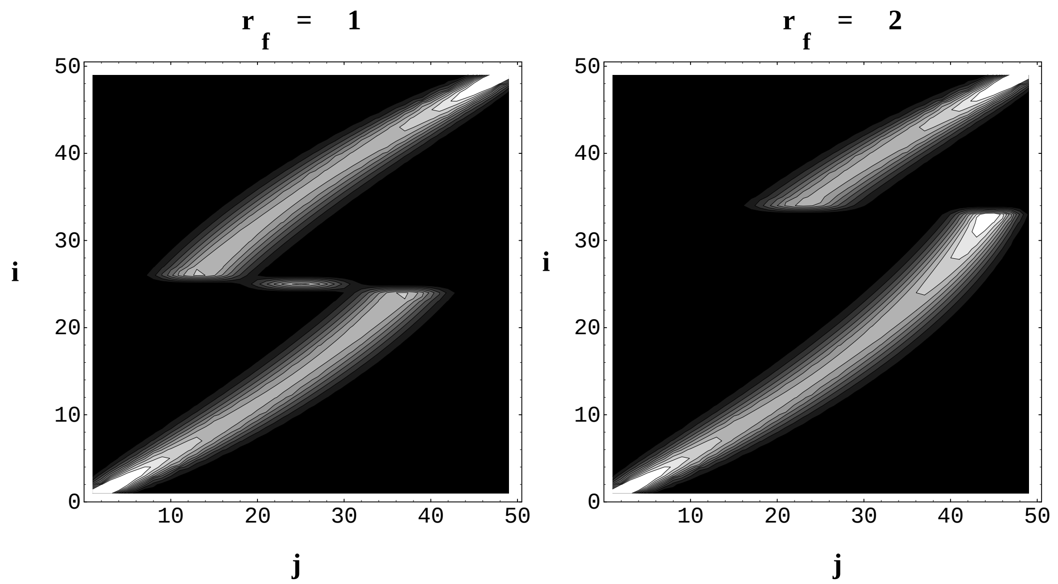

Figure 4.6: Simply conducting tournament selection using shared fitnesses leads to oscillatory, chaotic behavior. Here there is no overlap ($r_o = 0$). On the left, the fitnesses of the two niches are equal ($r_f = 1$), while on the right $r_f = 2$.

behavior in general. Here we have a chance to visualize this via our contour plots of Markov transition matrices.

We assume binary tournament selection (tournament size $s = 2$) using the shared fitnesses for determining the winner of each tournament. That is, we randomly (without replacement) choose two competitors. The competitor with higher shared fitness wins, and is therefore selected for reproduction. Thus the probability of selecting a copy of **A** (i.e., having it compete in and win a tournament) depends on whether its shared fitness is higher than **B**'s shared fitness. If $f_{A,sh} > f_{B,sh}$, then $p_A = 1 - (\frac{n_B}{N})^2$, which is simply its chance of being at least one of the competitors in a random tournament. (Recall that **A** is guaranteed to win any tournament in which it competes). But if $f_{A,sh} < f_{B,sh}$, then $p_A = (\frac{n_A}{N})^2$, which is the probability that it only competes against itself. Plugging in similar discontinuous functions for p_B into our transition probabilities, we define our Markov chain for "naive tournament sharing".

Figure 4.6 illustrates the transition matrices for two fitness ratios with population size $N = 50$. We note the discontinuity around the equilibrium. Apparently we can expect the

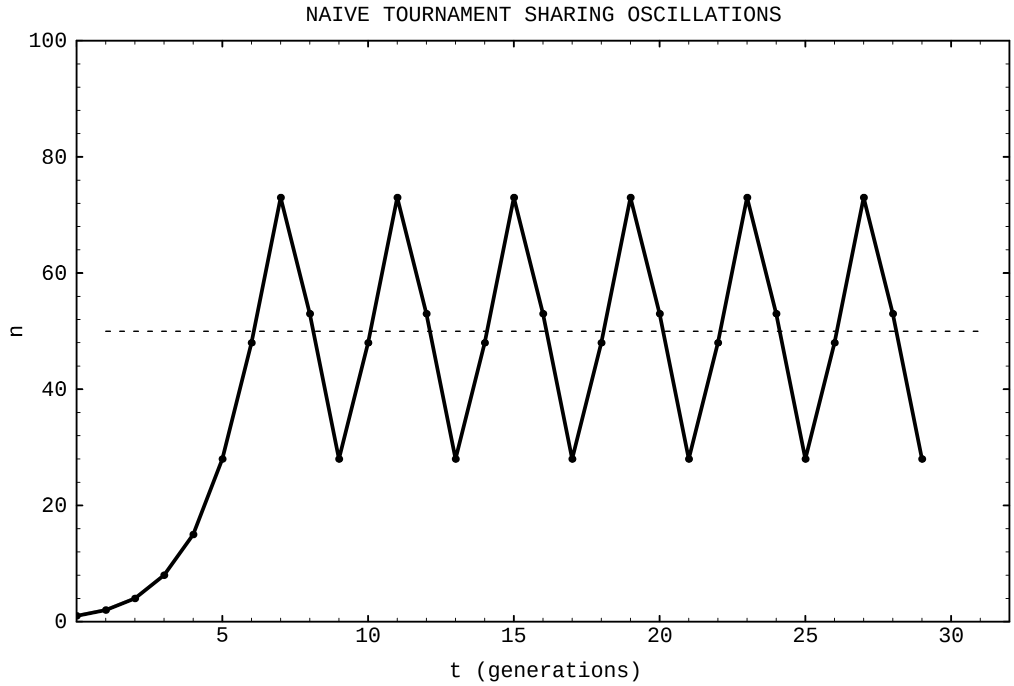

Figure 4.7: Lotka-Volterra predator-prey oscillations in species sizes?

GA to overshoot and undershoot the equilibrium, perhaps oscillating about it. In the case of equal fitness $r_f = 1$ we do see an equilibrium in the Markov process, but note how a small perturbation from the equilibrium point $N/2 = 25$ will lead to oscillatory behavior with no apparent pressure to return to equilibrium. So we confirm what Oei, Goldberg, and Chang (1992) predict, and we see that fitness sharing can induce an *unstable equilibrium* which we defined earlier. Although there is no "pressure" to leave equilibrium, the stochastic nature of tournament selection (random competitors) means we will be perturbed away from equilibrium. Note also that in general even unstable equilibria are not present, as in $r_f = 2$ in Figure 4.6, or in the case of an odd sized (odd N) population with $r_f = 1$.

In Figure 4.7 we trace the expected proportion of **A**s in a size $N = 100$ population under naive tournament sharing with $r_f = 1$, applying the Markov transition matrix each genera-

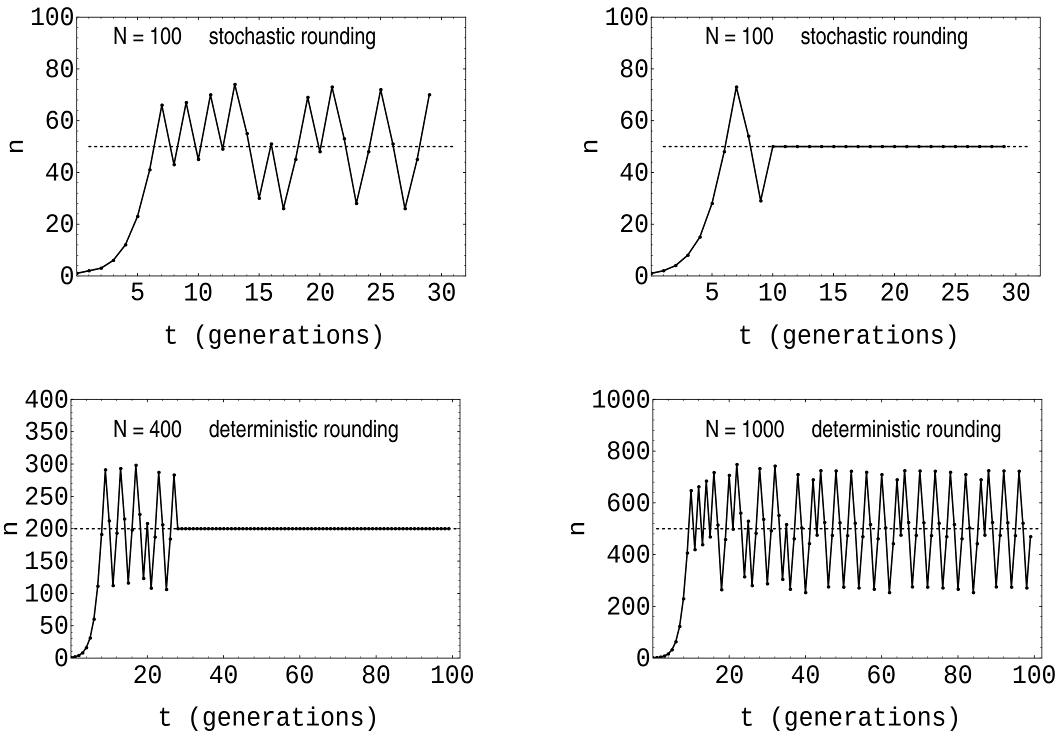

Figure 4.8: Various misleading artifacts of our modeling choices.

tion and then condensing the resulting probability distribution (over all possible states) into an expected state: the number of **A**s. We start with one copy of **A** at $t = 0$, the initial generation. We observe a rapid convergence toward equilibrium, followed by an overshooting, then an undershooting, and so on; an apparently periodic oscillation. This seems reminiscent of predator-prey oscillations generated by Lotka-Volterra growth equations. One population booms at the expense of the other and eventually of itself.

Although the oscillations in Figure 4.7 seem periodic, we note that this is an artifact of the model. To generate Figure 4.7 we consistently rounded our expected n_A. Thus we are averaging out the randomness of an individual run by averaging over many runs (in effect). If we were to add back in some randomness by "stochastically rounding" (i.e., sometimes rounding up, sometimes truncating), we would lose this periodicity, as the upper left plot in Figure 4.8 shows. Figure 4.8 illustrates some other potentially misleading artifacts of this kind of modeling. For example, the upper right plot shows that if our trace ever lands on the unstable equilibrium, it will stay there! This not realistic. Also, larger population sizes (lower left and lower right, Figure 4.8) can exhibit complex, semi-periodic behavior even with the "deterministic rounding". Finally, we note that Oei, Goldberg, and Chang (1992) predict chaotic behavior, and we can see indications of such behavior even with only two niches. In the case of $k > 2$ niches, we can imagine that the oscillations would be so coupled as to always result in chaotic behavior.

Clearly, capturing the actual behavior of the population under naive tournament sharing must be done carefully. The assumptions and simplifications of our models can mislead. However, the Markov model of Figure 4.6 is exact, and clearly indicates large oscillations about the niching equilibrium, whether chaotic or periodic.

On a last note, Oei, Goldberg, and Chang (1992) recommend a simple change, to use *continuously updated tournament sharing* to avoid such oscillations. This approach has been used successfully, but is not modeled by Markov chain here.

4.2.4 A Markov Model of Resource Sharing (Overlapping Niches)

We demonstrate the presence of a similar but unique niching force in resource sharing. We use the LCS as our instantiation of the abstract *resource sharing* paradigm.

4.2.4.1 Setting up the Model

We now make use of the Markov chain models described above to visualize niching under resource sharing. We use the shared fitnesses for rules **A** and **B**, from Equations 2.4 and 2.5 and substitute them for the objective fitnesses f_A and f_B in the derivation of Equation 4.16. Recall that n_A is the number of copies of rule **A** in our population of N rules, and let the rest of the population consist of n_B copies of rule **B**. Under the principle of resource sharing, which is the equal division of a resource among all individuals competing for it, we calculated the *expected* shared fitnesses of **A** and **B**:

$$f_{sh,A} = \frac{f_A - f_{AB}}{n_A} + \frac{f_{AB}}{N} \qquad\qquad f_{sh,B} = \frac{f_B - f_{AB}}{n_B} + \frac{f_{AB}}{N}, \tag{4.19}$$

where f_{AB} is the set of examples (i.e., fitness) covered by both **A** and **B**. Substituting the shared fitnesses for the objective fitnesses in the probability of selection $p_A = \frac{n_A f_A}{n_A f_A + n_B f_B}$, and rearranging, yields

$$p_A = \frac{f_A - f_{AB} + f_{AB}\frac{n_A}{n_A+n_B}}{f_A - f_{AB} + f_{AB}\frac{n_A}{n_A+n_B} + f_B - f_{AB} + f_{AB}\frac{n_B}{n_A+n_B}}. \tag{4.20}$$

Recalling that $n_B = N - n_A$, and dividing numerator and denominator by f_A, the above reduces to

$$p_A = \frac{1 - \frac{f_{AB}}{f_A} + \frac{f_{AB}}{f_A}\frac{n_A}{N}}{1 - \frac{f_{AB}}{f_A} + \frac{f_{AB}}{f_A}\frac{n_A}{N} + \frac{f_B}{f_A} - \frac{f_{AB}}{f_A} + \frac{f_{AB}}{f_A}\frac{N-n_A}{N}}. \tag{4.21}$$

Remembering our defined ratios $r'_f \equiv \frac{f_B}{f_A}$ for fitness, and $r_o \equiv f_{AB}/f_A$ for overlap, we find that

$$p_A = \frac{1 + r_o(\frac{n_A}{N} - 1)}{1 - r_o + r_o\frac{n_A}{N} + r'_f - r_o + r_o\frac{N-n_A}{N}}. \tag{4.22}$$

Finally, simplification yields

$$p_A = \frac{1 - r_o + r_o\frac{n_A}{N}}{1 - r_o + r'_f}. \tag{4.23}$$

(Recall that we earlier calculated the *proportion*, P_A, of **A**s in the population. We note here that in general $P_A \neq p_A$; that is, the *probability* p_A of choosing **A** from the current population using fitness proportionate selection is **not** the same as the *proportion* P_A of **A** in the current population. However, at equilibrium (i.e., any steady-state, such as uniform populations) this *is* true (i.e., $p_A = P_A$).) Similarly, for **B**,

$$p_B = \frac{r'_f - r_o\frac{n_A}{N}}{1 - r_o + r'_f}. \tag{4.24}$$

Before inserting these probabilities into our transition function $P_{trans}[i,j] = \binom{N}{j}(p_A)^j(p_B)^{N-j}$, we add in the time dependency notation, letting i be the number of copies of **A** in the current population at time t, and letting j be the number of copies in the next generation, at time $t+1$: $i \equiv n_{A,t}$ and $j \equiv n_{A,t+1}$. Now we can compute the complete transition matrix:

$$P_{trans}[i,j] = \binom{N}{j}\left(\frac{1 - r_o + r_o\frac{i}{N}}{1 - r_o + r'_f}\right)^j \left(\frac{r'_f - r_o\frac{i}{N}}{1 - r_o + r'_f}\right)^{N-j}. \tag{4.25}$$

Compare our Markov chain for resource sharing with that of fitness sharing (Equation 4.18) and that of perfect sharing (Equation 4.17). Note that when there is no niche overlap ($r_o = 0$), the Markov chain (i.e., transitions) for resource sharing reduces to that of perfect sharing (Equation 4.17).

4.2.4.2 Visualizing Restorative Pressure

Now we can begin to visualize the niching force in resource sharing. Let's again examine contour plots of the transition probability matrix. In Figure 4.9 we show several cases of overlapping coverage of rules **A** and **B**. Again, the population size is $N = 50$ and the fitness ratio of f_A to f_B is $r_f = 2$. In the margins we try to depict the corresponding coverages graphically. Note how the restorative pressure degrades with increasing r_o, from perfect sharing[7] at $r_o = 0$ to pure selection at $r_o = \frac{1}{r_f}$, which is the maximum overlap possible for a given r_f. So with complete overlap the restorative pressure, or niching force, disappears and only selection pressure remains. We observed the same phenomenon in fitness sharing. Note that at $r_f = 1$ and $r_o = \frac{1}{r_f} = 1$, we have the case of pure genetic drift.

4.3 Can Equilibrium be Maintained? Equilibrium versus Steady-State

So far we have established that equilibrium equations can be solved for both kinds of sharing, fitness and resource, and we have set up Markov models for each, by defining their state transitions. Furthermore, we have plotted the transition matrices to get a visual impression of a "restorative pressure" that apparently drives the population toward the niching equilibrium. Next we try to use Markov chain analysis to rigorously show that these equilibrium points are meaningful: that is, they can be maintained for long periods of time. We first try a straightforward application of standard asymptotic analysis of Markov processes. Traditional Markov chain asymptotic analysis finds the long-term "steady-state" of the model. Is the steady-state equivalent to the niching equilibrium? We will find that the asymptotic analysis is limited, in that it concentrates on the absorbing states and ignores the transient behavior of interest to us. We then modify the asymptotic analysis to focus on the transient behavior, in

[7] Remember that under resource sharing, the fitness division is often stochastic. We can therefore expect more noise (i.e., a greater spread) in the probability distributions for each row of the matrix if we were to model the stochastic allocation of credit under resource sharing. But this is not a qualitative difference in the Markov model.

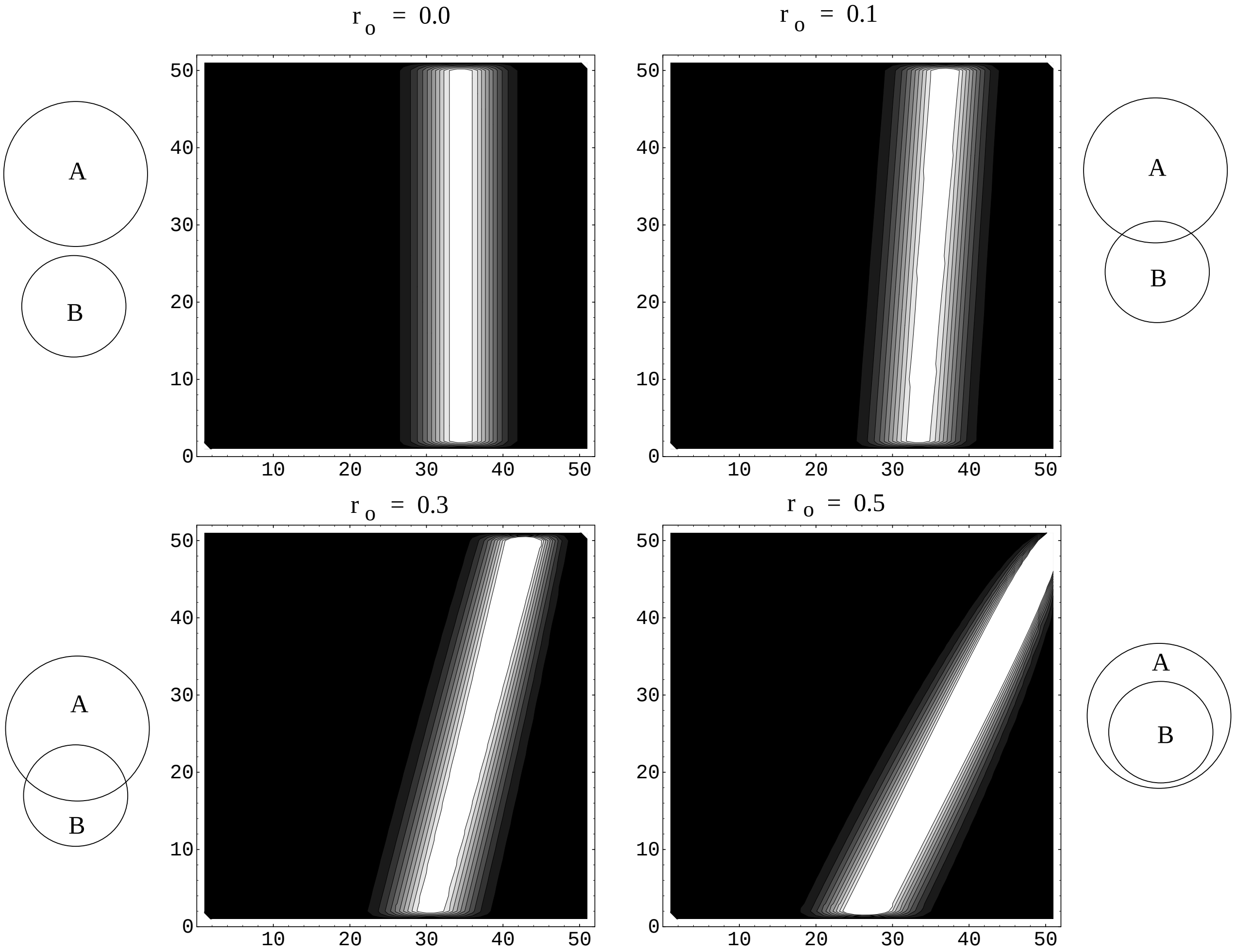

Figure 4.9: The niching pressure of resource sharing decreases with increasing overlap r_o. In all four cases $r'_f = \frac{1}{2}$.

which niching equilibrium *is* maintained. After verifying that sharing equilibrium can indeed be considered a steady-state in a Markov process, we can go on in the next section to derive useful closed-form results for niche (equilibrium) maintenance times.

As in previous sections, we begin our analysis with perfect sharing, the case of no niche overlap, which is common to both types of sharing. After deriving some general results for this case, we then specialize the analysis for the two different ways of handling niche overlap: fitness sharing and resource sharing.

4.3.1 Perfect Sharing

We try to make as much progress as possible in our analysis of perfect sharing, because such progress then applies to both fitness and resource sharing. In particular, in this section we examine the meaning and implications of the absorbing states.

4.3.1.1 Absorbing States and Steady-States

In all three cases of sharing (perfect, fitness, and resource), the Markov models reveal absorbing states for the uniform population distributions. Intuitively, since we assume a zero mutation rate, once our GA has converged to N copies of a single individual, there is no way to reintroduce diversity into the population, and any subsequent selection and crossover will not cause a change in the population makeup. But since niching pressures clearly (visually at least) drive the population *away* from these absorbing states, how much do, or should, the absorbing states affect the expected long-term behavior of the GA?

First, let's define *steady-state*. In Markov analysis, the steady-state of a Markov chain is actually a probability *distribution* over all possible states. Thus the Markov steady-state represents the expected portion of time spent in each state over any unit time interval beginning at some arbitrarily distant time in the future. To put it another way, the steady-state probability for each state s is the probability that we will find the Markov process in states s at some arbitrary time t in the future, as $t \to \infty$. We note that the steady-state distribution can of course be heavily focused on a particular state, even to the point of being entirely concentrated on a state s, so that its steady-state probability is one, and for all other states it is zero. In such singular cases, the steady-state is indeed a state of the Markov chain (rather than a distribution).

In standard Markov chain analysis, the only steady-states of an absorbing chain are absorbing states. Their existence in our models means that eventually the niching force must be overcome, despite its strength made obvious in Figure 4.3. The GA will be "trapped" into convergence.

4.3.1.2 Absorbing Markov Chain?

Yet in practice we do not wait around for the niched GA to converge to a uniform population. Often it is a matter of watching the population distribution for many generations and deciding that a run has "converged" to a noisy steady-state. Although the transition matrix tells us that this steady-state cannot last, we usually do not wait around long enough to see the GA wander that far from equilibrium.

So how do we reconcile our experience with our model? How can we talk about steady-states other than absorbing states when we have an absorbing chain? Two easy answers might come to mind from our earlier discussions:

- add mutation to force ergodicity, or
- use a less stochastic selection method to reduce drift.

The first easy answer involves any non-zero mutation rate $p_m > 0$. This would turn the Markov chain into an ergodic one, with no absorbing states, since there is a finite probability of leaving any state. In fact, there is a finite probability, however small, of reaching any state from any other state. But such a step seems unpromising; after all, we would be side-stepping the essential problem, which is in the model, not in the algorithm. However much a Markov chain analysis of sharing with $p_m > 0$ might prove the existence of a meaningful, long-term niching equilibrium, the fact would remain that sharing *without* mutation also maintains a long-term niching equilibrium. Changing our algorithm, however slightly, seems an unsatisfactory way to make up for a deficiency in our model. That said, one could argue that a niched GA with a very small mutation rate (all that is needed to establish ergodicity) could be an approximation, that is a *model*, of a niched GA with no mutation. And so an exact Markov model of the former would be an approximate model of the latter. Such an approach seems worthy of further investigation, but here we choose to fix our model, rather than our algorithm.

The second easy answer, substituting a less stochastic selection method, is suggested by our model of stochastic remainder selection in section 4.2.3.1. Recall that using stochastic remainder rather than roulette wheel can yield an absorbing state at niching equilibrium. This situation depends on carefully choosing population size and fitness ratio, but choosing even more deterministic selection methods can increase the chances of such an occurrence. Again we raise our objection above: why change the algorithm to fix a deficiency in the model[8]?

A third way of handling the uniform population absorbing states leaves the GA alone and instead changes the Markov chain: ignore the absorbing states and just analyze the transient states. In the well-known partitioning of states of an absorbing Markov chain,

$$P = \begin{pmatrix} Q & R \\ 0 & I \end{pmatrix}$$

we take the Q partition to be the entire matrix, ignoring $R, 0, I$, which only consist of two rows and two columns for the two absorbing states. If we normalize the Q matrix so that each row sums to 1, the resulting matrix, call it Q_{norm}, is an ergodic Markov chain that allows us to calculate steady-state probabilities for *all* (non-absorbing) states.

Before proceeding to analyze Q_{norm}, we must justify "chopping off" the absorbing state rows and columns. As an intuitive argument for ignoring the absorbing states, we look at the expected time to absorption. Absorption time is equivalent to the expected first passage time for the absorbing states. Expected absorption times for niched GAs should be much longer than those for the simple GA.

We calculate expected times to absorption from the Q matrix as in (Goldberg & Segrest, 1987). In short, we calculate the visitation matrix $V = (I - Q)^{-1}$, where "I" is the identity matrix and "-1" is the inverse operator for matrices, and assume a uniformly randomly generated initial population. Goldberg and Segrest calculated absorption times for a number of scenarios. For our comparison, we recall only their results for the case where $N \leq 20$ and $r_f = 1$. In a simple GA, with no niching and equal fitnesses ($r_f = 1$), absorption time is a linear function of population size. This is the case of genetic drift. We show this linear growth as the bottom-

[8]A second objection in this case is the fact that no matter which selection method is substituted, the uniform population states remain absorbing. Thus the asymptotic analyses would still show the GA spending a significant portion of total time at such states, far from niching equilibrium, and of little or no interest from a practical point of view.

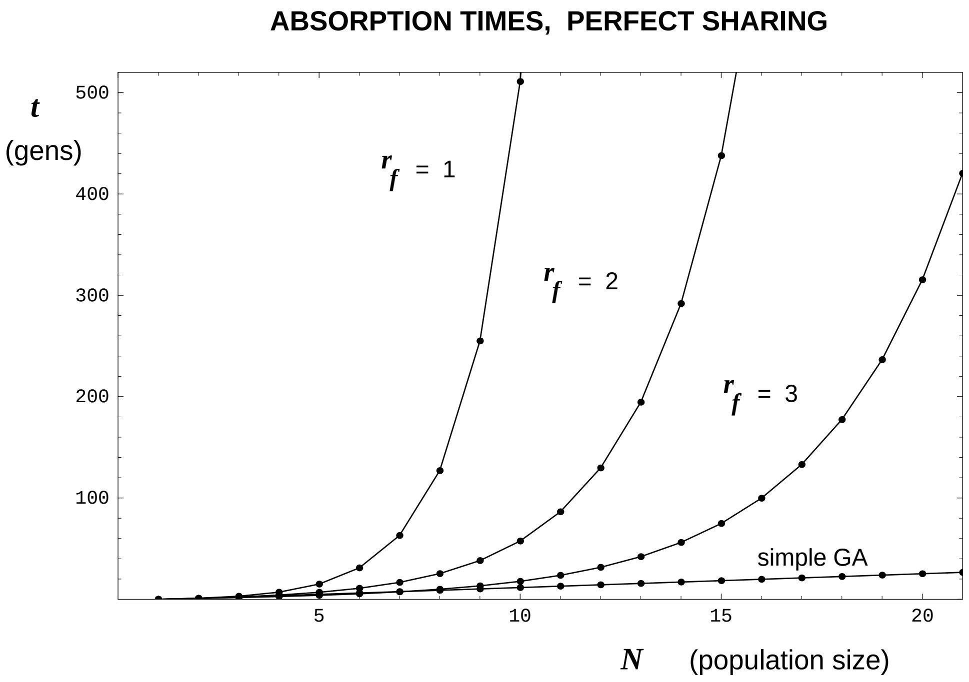

Figure 4.10: Expected times to absorption for perfect sharing.

most plot in Figure 4.10. We note that for $r_f \neq 1$ in the simple GA, we have the case of selective pressure driving the population toward convergence (i.e., toward one of the absorbing states). As Goldberg and Segrest illustrate, the expected absorption time for the simple GA under selective pressure grows sublinearly (specifically logarithmically) in population size N. We do not show such growth rates here in Figure 4.10, but rather concentrate on the superlinear growth rates of niching absorption times.

We calculate the expected absorption times t_{abs} for perfect sharing, also using $N = 1 \ldots 20$ and random initial populations. In Figure 4.10, these times are plotted for three different values of r_f. Note how even with $r_f = 3$, absorption time appears to grow exponentially with population size. In Figure 4.11 we plot the natural logarithm of expected times to absorption. The apparent linear growth in logarithmic times support our conjecture of exponential growth in t_{abs}.

Since the absorbing states have no effect on the behavior of the niched GA until they are entered, then as absorption time becomes arbitrarily large, the transient states appear increasingly like ergodic states to a Markov analysis, or so the intuition goes.

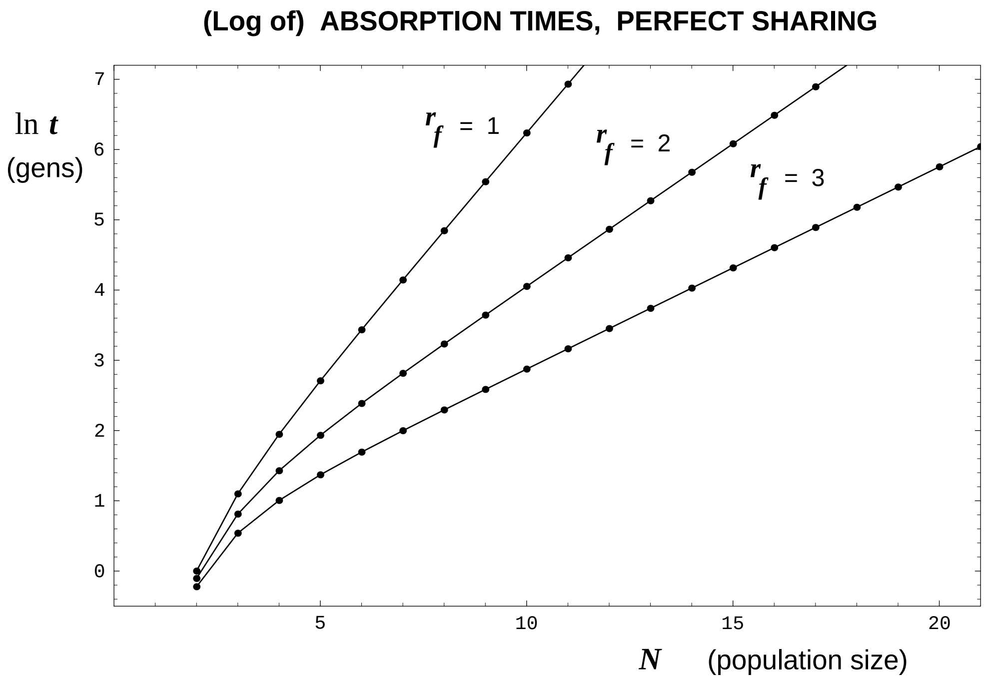

Figure 4.11: Absorption times for perfect sharing do indeed grow exponentially in population size N, even with varying fitness ratio r_f.

Fortunately, the problem of calculating steady state probabilities for "near-ergodic" absorbing Markov chains has received some attention in the applied probability literature. In (Darroch & Seneta, 1965), the authors examine several alternative approximations to the "effectively ultimate distribution". A thorough treatment of their work in relation to our matrix is beyond the scope of this dissertation. The interested reader is invited to compare two of their "best" candidate approximations, namely the *stationary conditional distribution*, $\vec{v}$, and the *quasi-stationary probability*, $w_j v_j$, to the steady-states $\vec{\Pi}$ we obtain for Q_{norm}. The stationary conditional distribution for Q is the left eigenvector, $\vec{v}$, of Q corresponding to the maximum modulus eigenvalue λ_{max}. The quasi-stationary probability is the product of the right and left eigenvectors, $\vec{w}$ and $\vec{v}$, of Q corresponding to λ_{max}.

We have compared the steady-state vector $\vec{\Pi}$ of Q_{norm} (calculated in the next section) to $\vec{v}$ and $w_j v_j$. For perfect sharing all three distributions are exactly the same[9], but in the case of niche overlap, these three distributions diverge. So to deepen our understanding of these

[9]The convergence of the three measures for perfect sharing follows from the row independence of Q for perfect sharing. $\vec{\Pi}$ for a row identical matrix is simply the row vector. The left eigenvector of λ_{max} is the same row vector, while the corresponding right eigenvector $\vec{w}$ is the all-ones vector, so that $w_j v_j = v_j, \forall j$.

"quasi-ergodic chains", and their accuracies, we consider the case of niche overlap, and thus proceed to the next section, fitness sharing, to discuss ergodic chains.

4.3.2 Fitness Sharing

The effect of niche overlap σ_{sh} on absorption times seems akin to that of fitness difference r_f. Figure 4.12 illustrates that increasing $\sigma_{sh} > 1$, while holding the fitness ratio constant at $r_f = 2$, has the effect of apparently decreasing the base in the exponential growth of absorption time with population size, just as increasing or decreasing r_f away from $r_f = 1$ had a similar effect in Figure 4.10. However, the logarithmic plot in Figure 4.12, bottom, indicates that growth in absorption times is still exponential in N, even with niche overlap.

Unlike r_f however, σ_{sh} seems to complicate the Markov process and hence our models of it. For example, varying r_f away from one does not change the fact that the transient state rows of the Markov transition table are identical. But changing σ_{sh} from one does cause the Markov chain to lose this significant property. One consequence of this is that the three stationary distribution vectors mentioned above all agree for all r_f with $\sigma_{sh} = 0$, but diverge as σ_{sh} increases beyond one. However, the divergence of the three distributions appears to grow very slowly with increasing σ_{sh}. Since our stationary distribution vector $\vec{\Pi}$ is in close agreement with the measures from Darroch and Seneta, and because it is much more easily calculated than are eigenvectors and values, we use it in our continuing study of niched GA steady-states.

4.3.2.1 Ergodic Markov Chain

We now have an irreducible Markov chain, Q_{norm}, with all ergodic states, 1 through $(N-1)$. Calculation of the steady-state probabilities is straightforward. We seek the vector of steady-state probabilities $\vec{\Pi} = \{\pi_1, \pi_2, \ldots, \pi_{N-1}\}$, where π_j is the steady-state probability for state j. To find $\vec{\Pi}$, we solve[10] ($\vec{\Pi}\, Q_{norm} = \vec{\Pi}$). We can analyze the vector $\vec{\Pi}$ to help us understand the behavior of our niched GA at steady-state. Figure 4.13 plots the steady-state distributions for a population size $N = 16$, fitness ratio $r_f = 1$, and four different values of σ_{sh}. We note that even with perfect sharing, our steady state for this small population size is fairly noisy. And as σ_{sh} increases, the steady-state distribution flattens. We could perform a number of statistical

[10]For a unique solution, remember that $\sum_{i=1}^{N-1} \pi_i = 1$.

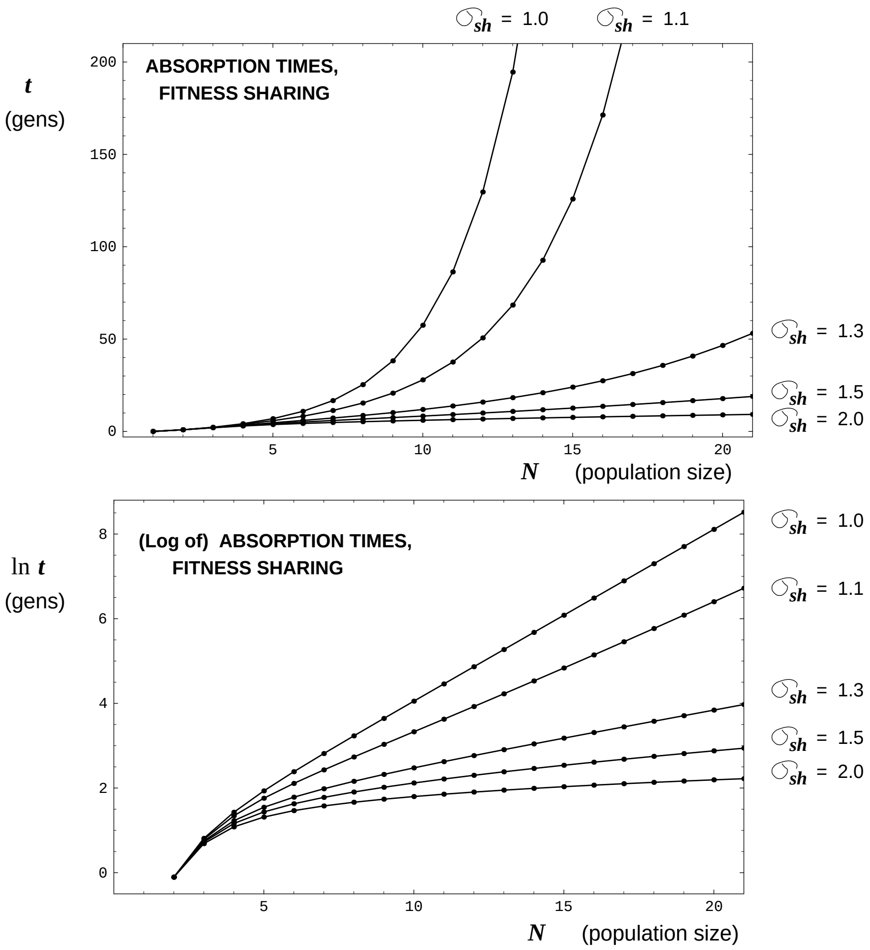

Figure 4.12: Expected times to absorption for fitness sharing show exponential growth with population size N, even for cases of overlap $\sigma_{sh} > 1$. For all plots here $r_f = 2$.

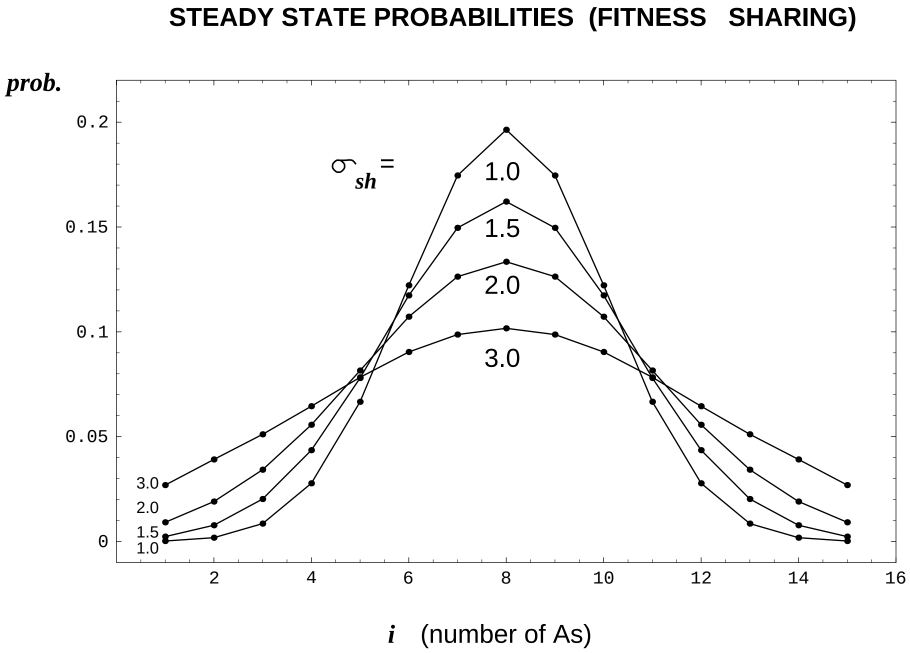

Figure 4.13: Steady state probability distributions, $\vec{\Pi}$, for fitness sharing with $N = 16$, $r_f = 1$, and varying degrees of overlap σ_{sh}.

analyses of the distribution, estimating the total expected amount of time spent within a certain range of states, or the expected recurrence time for returning to a particular level of **A**s in the population. In particular, these distributions can be approximated by a normal distribution whose variance, σ^2, would be an inverse measure of the stability and noisiness of the steady-state. Clearly, σ^2 is directly related to σ_{sh}.

For now, we use the steady-state distribution to calculate the *expected number of* **A***s*, $E[i]$, in the population at steady-state, and compare this to the equilibrium prediction $n_{A,eq}$ under perfect sharing assumptions (i.e., no overlap), from Equation 4.3. We can think of $E[i]$ as the number of **A**s we expect to see in the population at any distant point in the future. To calculate the expected number of **A**s, we multiply the steady-state probability for each state by the index for that state and then sum over all states: $E[i] = \sum_{i=1}^{N-1} i * \pi_i$.

Figure 4.14 compares $E[i]$ for different degrees of overlapping niches to $n_{A,eq}$ using a population size $N = 12$ and different values of fitness ratio r_f. The solid black line is the plot of the predicted equilibrium ($n_{A,eq}$) under perfect sharing (no overlap), from Equation 4.3. Note how all three expected value plots agree exactly with $n_{A,eq}$ only at $r_f = 1$. This agreement is due to the symmetry of every steady-state distribution when $f_A = f_B$. The small difference between

EXPECTED NUMBER OF A's

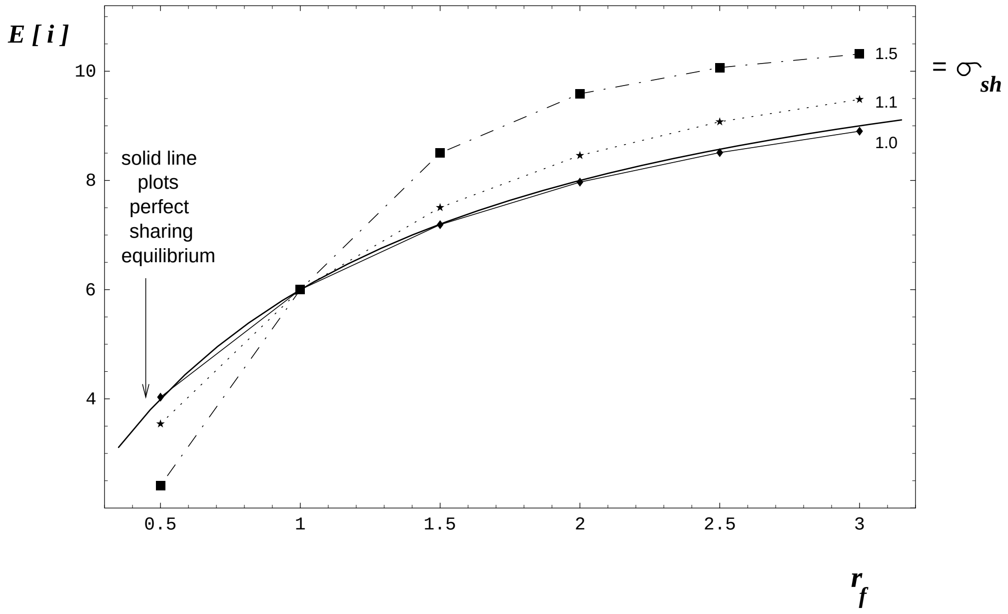

Figure 4.14: Expected number of **A**s at steady state.

$E[i]$ for perfect sharing ($\sigma_{sh} = 1.0$) and $n_{A,eq}$, when $r_f \neq 1$, must be due to the asymmetry of the error introduced in normalizing the Q matrix. The larger divergence from $n_{A,eq}$ for the overlapping niches, however, stems from the weakening of restorative pressure. The weaker niching force allows preferential selection to bias the steady-state toward the more highly fit solution. Note also that for any value of $\sigma_{sh} > 1$, increasing r_f while holding σ_{sh} constant also increases the difference between $E[i]$ and the $n_{A,eq}$. Although not shown, increasing population size N, holding r_f and σ_{sh} constant, decreases the divergence of any $E[i]$ from $n_{A,eq}$. Intuitively, degrading the steady-state by increasing r_f, σ_{sh} or decreasing N allows the asymmetric forces of selection pressure and drift to be felt.

In our current analysis of expected numbers of **A**, $E[i]$, we have been comparing our calculated $E[i]$ with the predicted equilibrium for perfect sharing (from Equation 4.3). But to further investigate the accuracy of our quasi-ergodic Markov approximation, and its steady-state $E[i]$ predictions, we should compare these predictions to the fitness sharing equilibrium which accounts for overlap (Equation 4.6):

$$n_{A,eq} = \frac{r_f \sigma_{sh} - \sigma_{sh} + 1}{1 + r_f} N.$$

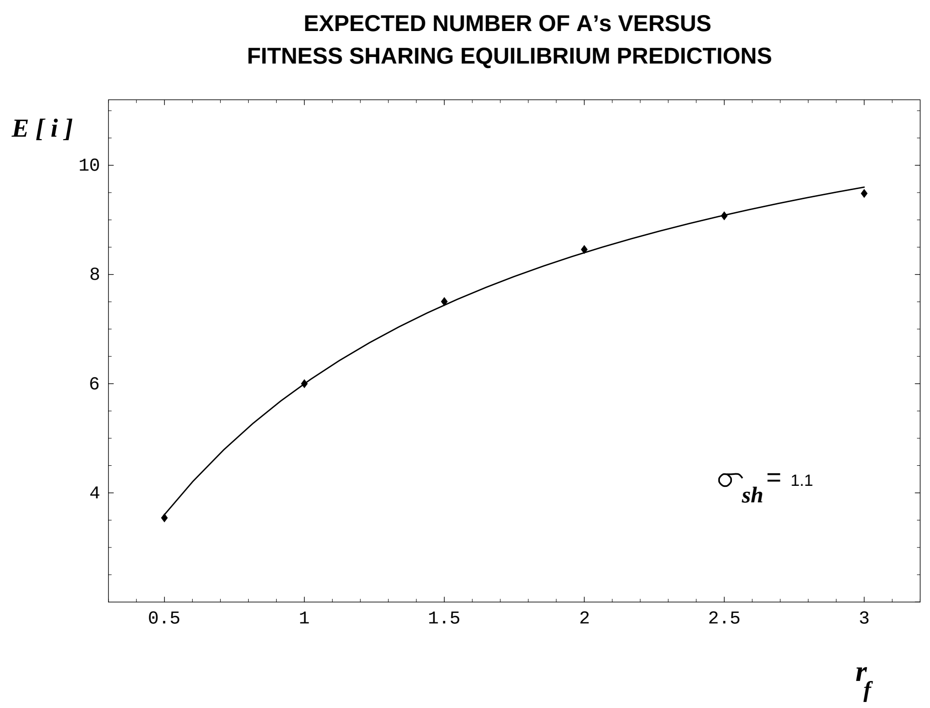

Figure 4.15: A comparison of our steady-state $E[i]$, calculated from the quasi-ergodic approximate Markov chain, agrees with the prediction from the simple fitness sharing equilibrium equation, at least for small overlap ($\sigma_{sh} = 1.1$ here).

Figure 4.15 shows the plot of $n_{A,eq}$ from the equation above as a solid line, while the points are the $E[i]$ from the steady-states of our quasi-ergodic approximation. Here we assume a population of $N = 12$ individuals, and a small overlap of $\sigma_{sh} = 1.1$. The differences appear small, but we recall that as overlap increases, the ergodicity assumption becomes less accurate, and we should expect divergence between the two predictions. As for the actual behavior, the variance in the actual $E[i]$ should grow as σ_{sh} increases and niching degrades.

4.3.3 Resource Sharing

To further establish that resource sharing generates a restorative pressure akin to that of fitness sharing, we also look at expected absorption times and steady-state distributions versus equilibrium, under resource sharing. Again we find that the expected absorption time appears to grow exponentially with population size N, just as with perfect sharing and fitness sharing, with the exponent in the growth decreasing as r_f moves away from one. Holding r_f constant, we find that the exponent of growth (in N) also decreases with increasing niche overlap, just as in fitness sharing.

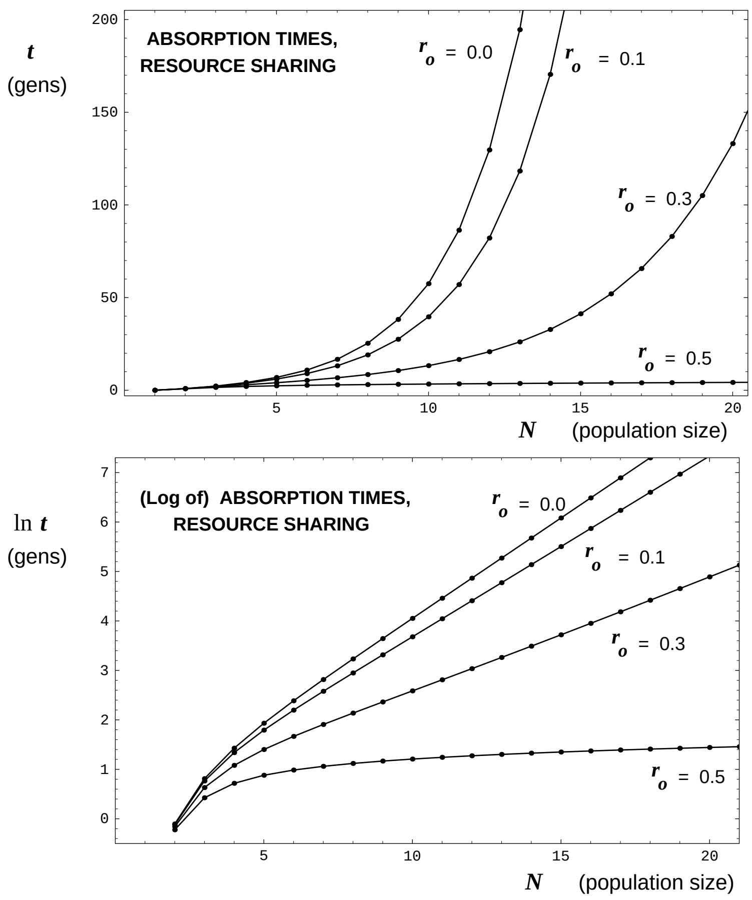

Figure 4.16: The expected time to niching failure under resource sharing appears to grow exponentially with population size N. The exponent of growth appears to decay with r_o. Here $r_f = 2$.

4.3.3.1 Time to Absorption

We vary $r_o > 0$ while holding r_f constant. In Figure 4.16 we show the growth in expected absorption (or niche loss) time as a function of N for $r_o = 0.0$ (perfect sharing), and $r_o = 0.1, 0.3, 0.5$, and $r_f = 2$. Recall that the expected time to niching failure under resource sharing is the absorption time of the Markov chain. This time appears to grow exponentially with population size N, a characteristic we propose as fundamental to niching. The exponent of growth appears to decay with r_o, representing a degradation of niching pressure. When $r_o = r'_f = \frac{1}{r_f}$, selection pressure dominates and growth in absorption time becomes linear for $r_f = 1$ (implying genetic drift) or logarithmic for $r_f \neq 1$ (implying selection pressure).

This growth (Figure 4.16) appears to be exponential. Indeed, for the case of perfect sharing ($r_o = 0$), we can find the exact exponent, in the next section. When $r_o \neq 0$, it might be impossible to calculate a closed-form expression for absorption time, thus making it difficult to prove exponential growth. However, in Figure 4.16 it certainly appears that all such growth is exponential, at least up until $r_o = \frac{1}{r_f}$, at which time the growth should become logarithmic (linear) in N, since this is the case of selection (drift). We suggest that exponential growth in niche failure time (i.e., absorption time), as N grows, is entailed by any restorative pressure, and thus could be used as an indicator of a "true nicher". If that is the case, then resource sharing clearly qualifies.

4.3.3.2 Steady-state Distributions and Equilibrium Points

Our results on niche maintenance times indicate that for realistically sized populations (i.e., $N \gg 20$) we can expect resource sharing to maintain a set of niches for a very long time (subject to the effects of other sources of noise). We next want to know what kind of distribution will be maintained during that time. Will the steady-state distribution center on the resource sharing equilibrium (Equation 4.10):

$$r_{eq,n} = \frac{r'_f - r_o}{1 - r_o}?$$

We now use the same *quasi-ergodic* approach used earlier with fitness sharing to approximate the absorbing chain by an ergodic chain, for the purpose of calculating the steady-state distribution. Again we simply "chop off" the rows and columns corresponding to the absorbing states. We then have to normalize the resulting (Q) submatrix so that all rows sum to one. The normed submatrix Q_{norm} can then be used to calculate the steady-state distribution of probabilities among the remaining $N - 1$ transient states.

We have performed the above calculation on the transition matrix for resource sharing (Equation 4.25) for a population size $N = 16$, fitness ratio $r_f = 1$, and various degrees of overlap $r_o \geq 0$. Figure 4.17 plots the distribution of probabilities for $r_o = 0.0$, 0.5, 0.7, and 0.8. This distribution is similar to that plotted in Figure 4.10 for fitness sharing. The highest probability is at the equilibrium point of $n_A = 8$. The distribution degrades (i.e., the variance increases) with increasing overlap. The apparent variance in the distribution at $r_o = 0$ (i.e., perfect sharing) is due entirely to the stochastic selection operator (roulette wheel selection).

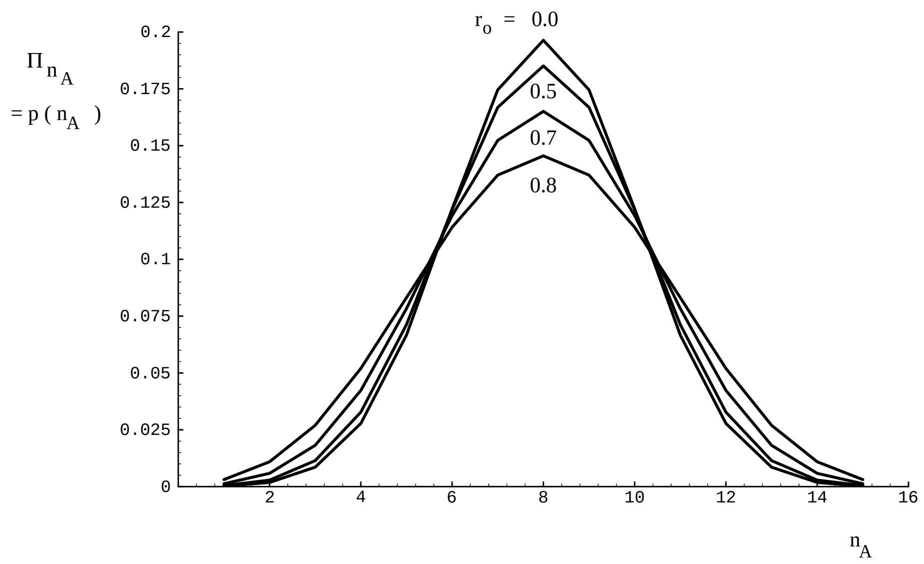

Figure 4.17: The steady-state probability distribution of the Markov chain model of resource sharing. Here we use population size $N = 16$, equally fit rules $r_f = 1$, and various degrees of overlap $r_o = 0.0, 0.5, 0.7, 0.8$. In all four cases, a distribution symmetric about the equilibrium point of $n_A = 8$ is maintained. This distribution appears to gradually degrade with overlap.

4.3.3.3 Niching Pressure at Equilibrium

The above results demonstrate the existence of a steady-state distribution centered on the equilibrium point predicted by the equilibrium condition (where all shared fitnesses are equal). We can use $r_{eq,n}$ above to look at the resource sharing function at and near the equilibrium point. With this $r_{eq,n}$ in hand, we can go back to our resource sharing function shown in Figure 3.9 as a function of r_o and r_n, and look at slices of the surface corresponding to $r_n = r_{eq,n}$ for various r_f. This gives us the sharing function that influences the population most of the time, or at least in expectation, since we expect the population distribution to stay near $r_{eq,n}$. Such a sharing function is single dimensional, a function of r_o (overlap) only, allowing us to compare the resource sharing to fitness sharing more directly.

In Figure 4.18, we plot $r_{eq,n}$ as a function of overlap r_o, for several different values of r_f. We see that the equilibrium point changes with overlap, disappearing when overlap is complete ($r_o = \frac{1}{r_f}$). For $r_f = 1$, there is no selection pressure and hence no preference. These curves lie in the (r_o, r_f) plane, and can be used to take slices of the resource sharing function in Figure 3.9.

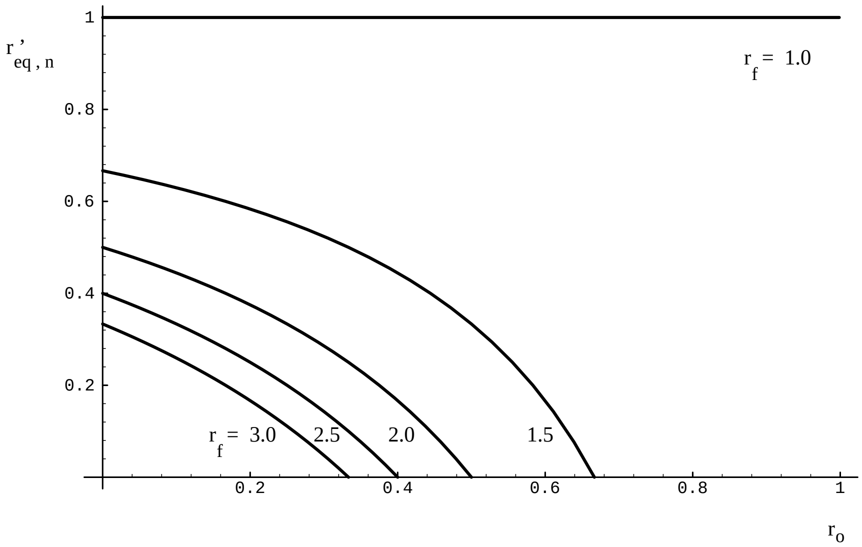

Figure 4.18: Equilibrium proportions change with varying overlap r_o and fitness ratios r_f.

We need only substitute $r_{eq,n}$ for r_n in Equation 3.12, and simplify:

$$\left[\frac{\partial D}{\partial n_B}\right]_{eq} = \frac{r_o}{(r_o - r'_f - 1)^2}. \tag{4.26}$$

We can then plot $\left[\frac{\partial D}{\partial n_B}\right]_{eq}$ as a function of r_o only: Figure 4.19 is a such a plot of slices (Equation 4.26) through the general sharing function $\frac{\partial D}{\partial n_B}$. That is, Figure 4.19 gives the one-dimensional resource sharing function that is arguably of greatest interest. If we assume that resource sharing will maintain a population distribution at or near $r_{eq,n}$ in Equation 4.10, then each copy of rule **B** will increase the denominator in the "divisional degradation" of rule **A**'s objective fitness by an amount that varies with the degree of overlap, as shown in Figure 4.19. The curves of Figure 4.19 look very much like those of fitness sharing in Figure 2.1, with $\alpha_{sh} < 1$. So we have further evidence that resource sharing behaves very much like fitness sharing. The differences (e.g., the way the resource "sharing function" varies as the population distribution moves away from equilibrium, while it is fixed in fitness sharing) are good foci for future analysis.

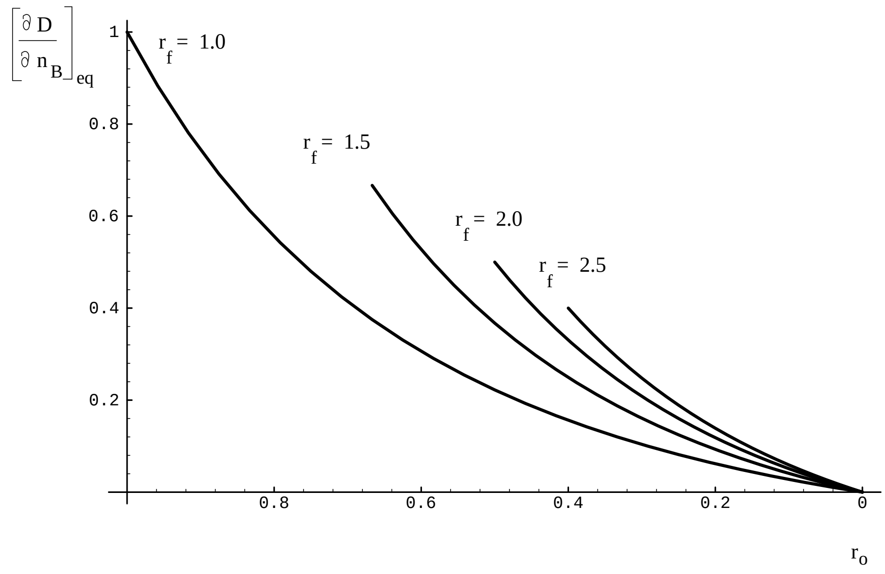

Figure 4.19: The resource sharing function along the curves of equilibrium.

4.4 Niche Maintenance Times

We conclude the analysis of this chapter by developing some useful measures of niche maintenance times. Specifically, we use expected absorption time as our measure of expected niche lifetime. To make absorption time practical, we seek closed-form expressions to approximate the exact but computationally expensive matrix calculations of Markov analysis.

To justify using absorption time t_{abs} as niche maintenance time[11], we rely on the results of the previous sections to convince ourselves that both fitness and resource sharing induce a restorative pressure that maintains a long-term steady-state distribution heavily concentrated on the niching equilibrium. That is, our previous analyses indicate that until a niche is lost (i.e., all members of **A** or **B**, or whatever niche, are eliminated from the population), niched GA selection will always try to return the population to the equilibrium point. We can therefore with some justification say that the expected time until niche loss (i.e., absorption) is the expected time that all of the current niches will be maintained[12].

[11] Absorption time t_{abs} is also called niche *extinction*, *failure* or *loss* time (e.g., Mahfoud, 1995a, 1995b).

[12] Niche maintenance time is also useful as an indicator of *quality* (or variance) of niche maintenance. The lower the expected niche maintenance time, the more fluctuations around niching equilibrium we can expect. After all, it is these stochastic fluctuations or variance around niching equilibrium that cause eventual niche loss.

Calculating the exact expected absorption time for an absorbing Markov chain is straightforward, although computationally expensive for large matrices (i.e., large population size N and/or many different niches). We therefore develop closed-form expressions, sometimes exact and sometimes approximate, below, for the usual three cases: perfect, fitness, and resource sharing. In all cases a random initial population, which has little chance of being in either absorbing state (for nontrivial population size N), is assumed.

4.4.1 Perfect Sharing: Exact Closed-Form Expression

In the case of perfect sharing, unlike the cases of niche overlap, we can develop an exact closed-form expression for absorption time t_{abs}. We can therefore confirm our suspicion that absorption time does indeed grow exponentially in population size N. (Although we suspect this is the case for non-zero niche overlap as well, especially given Figures 4.12 and 4.16, we cannot yet confirm this analytically.)

Since under perfect sharing the transition probabilities for all transient states are identical, it follows that prior to absorption we always have the same probability of absorption, namely $P_{absorb}(i) = P_{trans}[i,N] + P_{trans}[i,0]$, $\forall i : (0 < i < N)$. This is the probability that we will transit directly from the current state i (with i copies of **A**) to the absorbing state with N **A**s, or 0 **A**s. Now, $P_{trans}[i,N]$ is simply the probability of choosing N **A**s, which is the probability of choosing one **A**, raised to the N^{th} power: $P_{trans}[i,N] = (p_A)^N$. Similarly, $P_{trans}[i,0] = (p_B)^N$. The expected time to absorption, assuming that the chain is started in a transient state, is the inverse of the probability of absorption:

$$E[t_{abs}] = \frac{1}{(p_A)^N + (p_B)^N}. \tag{4.27}$$

Substituting $p_A = \frac{r_f}{r_f+1}$ and $p_B = \frac{r_f}{r_f+1}$ under perfect sharing (from Equation 4.17), yields

$$E[t_{abs}] = \frac{1}{(\frac{r_f}{r_f+1})^N + (\frac{r_f}{r_f+1})^N}. \tag{4.28}$$

After some rearranging:

$$E[t_{abs}] = \frac{(r_f+1)^N}{r_f^N + 1}. \tag{4.29}$$

The expected absorption time above grows exponentially in N for any $r_f > 0$. In particular, when $r_f = 1$, the growth is $O(2^N)$. We see that the expected absorption time indeed grows exponentially with population size N. The "effective base" in the growth decreases as r_f moves away from one.

4.4.2 Resource Sharing: Approximate Closed Form Expression

For $r_o > 0$ or $\sigma_{sh} > 1$ (i.e., niche overlap), it might be impossible to calculate a closed-form expression for the *exact* expected absorption time. But a closed-form expression for $E[t_{abs}]$ with overlap is desirable for our timing analysis, and we define an approximate one here. We use the approach above (for perfect sharing) by simply assuming that the population is *always* at the equilibrium point until the actual absorption event. That is, we assume only three possible states: equilibrium, and the two absorbing states (uniform populations). We do not try to justify this assumption here, nor do we try to bound the error, leaving this to future work. Rather, we merely observe that under the assumptions of very stable steady-states (e.g., large populations, low-variance selection), our approximation might be justified.

Justification aside, we go ahead and plug into the general Equation 4.27

$$E[t_{abs}] = \frac{1}{(p_A)^N + (p_B)^N},$$

using the probabilities of choosing **A** and **B** at equilibrium: $p_A \leftarrow p_{A,eq}$, and $p_B \leftarrow p_{B,eq}$ from Equation 4.12 above. Our assumption above translates here into the assumption that p_A is constant, at or near equilibrium, so that we can just use $p_{A,eq}$ and $p_{B,eq}$:

$$E[t_{abs}] = \frac{1}{(p_{A,eq})^N + (p_{B,eq})^N}. \tag{4.30}$$

If we assume we are at equilibrium, then the probability of selecting an **A** ($p_{A,eq}$) is the same as the proportion of **A**s at equilibrium ($P_{A,eq}$). This must be true since at equilibrium all (shared) fitnesses are equal ($f_{A,sh} = f_{B,sh}$), so

$$p_{A,eq} = \frac{n_A f_{A,sh}}{n_A f_{A,sh} + n_B f_{B,sh}} \Rightarrow \frac{n_A}{n_A + n_B} \Rightarrow \frac{n_A}{N} \Rightarrow P_{A,eq}.$$

We already have $P_{A,eq}$ from Equation 4.12:

$$P_{A,eq} = \frac{1 - r_o}{1 - 2r_o + r'_f} = p_{A,eq}.$$

Substituting $p_{A,eq}$ and $p_{B,eq} = 1 - p_{A,eq}$ into Equation 4.30 above yields

$$E[t_{abs}] = \frac{1}{(\frac{1-r_o}{1-2r_o+r'_f})^N + (1 - \frac{1-r_o}{1-2r_o+r'_f})^N}, \tag{4.31}$$

which simplifies just a bit to

$$E[t_{abs}] = \frac{(1 - 2r_o + r'_f)^N}{(1 - r_o)^N + (r'_f - r_o)^N}. \tag{4.32}$$

To test our closed-form approximation, we plot both the exact results (from the Markov chain model) and the approximations in Figure 4.20. In Figure 4.20, the solid dots are the plotted points for the exact (Markov) model, while the dashed lines are defined by Equation 4.31. In general, we see close agreement over the range plotted, except for high overlap (e.g., complete overlap when $r_o = \frac{1}{r_f} = 0.5$). As expected, our simple closed-form expression modeling absorption time loses accuracy for high overlap cases. Intuitively, with high overlap the variance of the equilibrium distribution will be high, and the population will not spend as much time near the equilibrium. Since the GA is spending less time at equilibrium, the use of the steady-state absorption probability as an estimate of the overall absorption probability introduces increasing error, as overlap increases.

4.4.3 Fitness Sharing: Approximate Closed Form Expression

For fitness sharing we replicate the above derivation for resource sharing's closed-form approximation of niche maintenance times. Starting with the general Equation 4.30

$$E[t_{abs}] = \frac{1}{(p_{A,eq})^N + (p_{B,eq})^N},$$

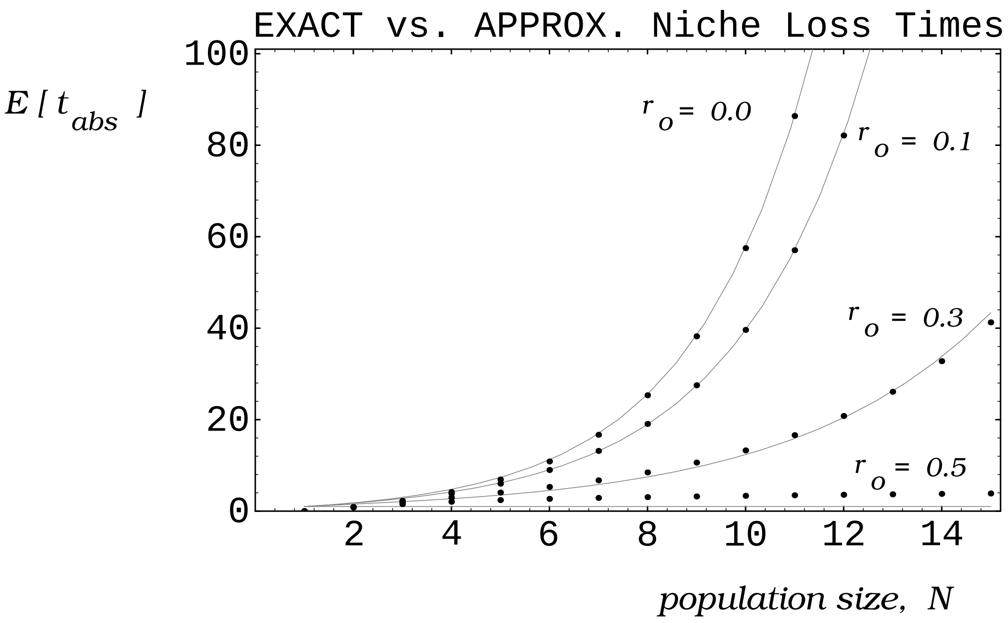

Figure 4.20: A comparison of *exact* expected niche loss times to the approximated times, as a function of population size. The exact results (from the Markov models) are shown as solid dots. The approximations, from our closed-form expression, are shown as dashed lines. The plots indicate general agreement for small niche overlap r_o. For all plots shown $r_f = 2$.

we substitute the probabilities of choosing **A** and **B** at *fitness sharing equilibrium*. Again, at equilibrium $p_{A,eq} = P_{A,eq}$, and we already have $P_{A,eq}$ from Equation 4.7:

$$P_{A,eq} = p_{A,eq} = \frac{r_f \sigma_{sh} - \sigma_{sh} + 1}{1 + r_f}.$$

We can calculate $p_{B,eq}$ from $p_{B,eq} = 1 - p_{A,eq}$:

$$p_{B,eq} = \frac{r_f - r_f \sigma_{sh} + \sigma_{sh}}{1 + r_f}.$$

Substituting $p_{A,eq}$ and $p_{B,eq}$ into Equation 4.30 above yields

$$E[t_{abs}] = \frac{1}{(\frac{r_f \sigma_{sh} - \sigma_{sh} + 1}{1 + r_f})^N + (\frac{r_f - r_f \sigma_{sh} + \sigma_{sh}}{1 + r_f})^N},$$

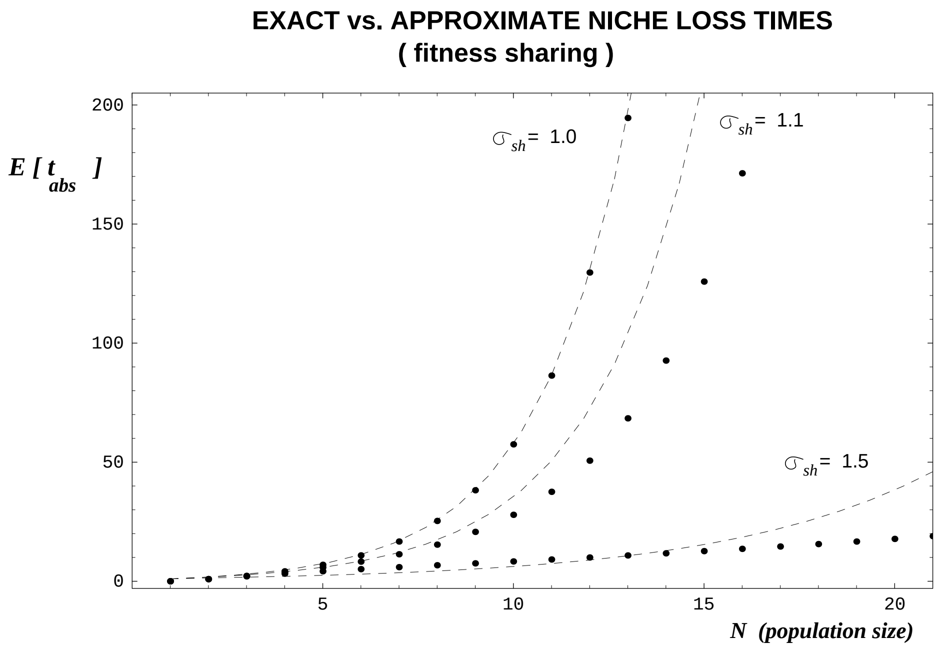

Figure 4.21: A comparison of *exact* expected niche loss times to the approximated times, as a function of population size. The exact results (from the Markov models) are shown as solid dots. The approximations, from our closed-form expressed model, are shown as solid lines. The plots indicate general agreement only for very small niche overlap (σ_{sh} near 1). For all plots shown $r_f = 2$.

which simplifies just a bit to

$$E[t_{abs}] = \frac{(1 + r_f)^N}{(r_f\sigma_{sh} - \sigma_{sh} + 1)^N + (r_f - r_f\sigma_{sh} + \sigma_{sh})^N}. \tag{4.33}$$

To test our closed-form approximation, we plot both the exact results (from the Markov chain model) and the approximations in Figure 4.21. In Figure 4.21, the solid dots are the plotted points for the exact (Markov) model, while the solid lines are defined by Equation 4.33. In general, we see close agreement over the range plotted only for fairly low degrees of overlap (e.g., $\sigma_{sh} \ll 1.1$). Again, our simple closed-form expression modeling absorption time loses accuracy for high overlap cases, as was true for resource sharing. But our approximate closed form expression for fitness sharing niche loss times appears to degrade much faster in accuracy with increased overlap.

4.4.4 Discussion

For both resource and fitness sharing, the closed form models are upper bounds on expected niche loss times. Thus they tend to err on the liberal side, predicting longer expected absorption times than what we can actually expect. This overly generous estimate can be seen in both Figure 4.21 and Figure 4.20. We know from our modeling assumptions that this type of error is inherent, since we assumed that the population distribution would always stay at equilibrium until a niche is lost. In reality, the population distribution will wander around equilibrium, and in such near-equilibrium states the chances of absorption are greater than they are at equilibrium. By ignoring the increased chances of absorption at near-equilibrium states, we cause our models to be upper bounds (on expectation). In future analysis, it might be desirable to obtain an accurate, but conservative bound on absorption times as well. To do so, we might choose a transition probability distribution different from the equilibrium, to plug into Equation 4.30. For our current analyses, we suggest that the above simple closed form models of absorption times, Equations 4.32 and 4.33, are suitable for our initial explorations of niching behavior and "meaning", especially in the case of resource sharing.

4.5 Summary

Through exploration and manipulation of the Markov model and associated asymptotic analysis, we have linked our previously calculated niching equilibria with Markov chain steady-state distributions. Our approximate analysis of an exact model (the Markov chain) supports our conjecture that the calculated niching equilibrium induced by sharing is indeed a stable one, under the dynamics of selection. Specifically, we have shown that niching equilibria exist and that they consist of diverse populations. Furthermore, the expected time until niche loss grows exponentially in population size N, even under increasing differences in niche fitness (r_f), and increasing competition (niche overlap r_o or σ_{sh}). This exponential growth means that it is easy to make N large enough to make niche maintenance times arbitrarily long, effectively infinite, for the practical intents and purposes of real GA implementations. Now that we have established confidence in sharing equilibrium (within the constraints of non-excessive overlap, sufficient population size, and non-disparate fitnesses), we can go on to derive some useful results. Our closed form approximations of niche loss times will prove very useful in Chapter 5.

Chapter 5

Convergence to Niching Equilibrium

Having found an upper bound on expected lifetimes of niches, and convinced ourselves that steady-state populations of rules can be maintained, we now turn to the question of how such steady-states are reached. Under normal selection (e.g., in a simple GA) it has been shown that convergence to the "equilibrium point" of a uniform population takes place very quickly. In the case of selective preference (i.e., $r_f \neq 1$) the convergence time grows logarithmically in population size N, while under genetic drift (i.e., $r_f = 1$), the expected convergence time grows linearly in N. Can we expect similarly quick convergence to "niching equilibrium"?

So called "niche convergence time" is also of interest because it provides a *lower bound* on niche maintenance time. We can compare this lower bound with our upper bound on niche maintenance times. When these two time bounds are close, we can expect poor niche maintenance. When they are far apart, we should expect long-term, steady-state niche maintenance. Under what conditions of niche overlap, fitness ratio, and population size can we expect to find the *phase transition* from poor to good niche maintenance; that is, the transition from competition to cooperation?

5.1 Background: Expected Proportions Analysis

To answer such questions, we use the simple, well-known method of *expected proportion equations* to model niche convergence. This method of tracking the expected next generation *population* only, rather than tracking the entire *distribution* over all possible populations, has been put to good use many times in the GA literature (e.g., Smith & Valenzuela-Rendón, 1989; Goldberg

& Deb, 1991; Neri & Saitta, 1995). The methodology has acquired several names. Vose (1993), noting that the accuracy of the proportion equations asymptotically approaches one with increasing population size, calls it the *infinite population model.* Recently, Neri and Saitta (1995) coined the term *virtual average population.* For an in-depth consideration of the accuracy of these models for finite populations, see Ben Goertzel's book (Goertzel, 1993, Appendix 1), in which he calls the trace given by the recurrent proportion equations the *local expected path*, as opposed to the "global" (exact) *expected path* given by the full Markov model. Also, Vose (1993, 1995) analyzes the differences and similarities between these two models (finite versus infinite population models).

To illustrate the expected proportions analysis, and to provide a basis for comparison with our niching results later, we review next some of the work done on convergence time in simple GAs (i.e., no niching).

5.2 Review: Simple GA Convergence

The expected proportion (or "recurrence relation" or "difference equation") analysis has been applied to a simple GA under proportionate selection by several authors in the literature, including Goldberg (1987; 1989a), Ankenbrandt (1991), and Goldberg and Deb (1991). Here we follow our own derivation, to arrive at the same kind of result: a closed-form expression for the expected proportion at generation t, with a logarithmic growth in convergence time with increasing population size N.

5.2.1 The Key Modeling Assumption

Under proportionate selection, the expected proportion of the next generation's population given to an individual is *equal* to the probability of selecting that individual for reproduction in the current generation's population. Thus if $P_{A,t}$ is the proportion of the current population, at time t, consisting of copies of rule $\mathbf{A}$, then $E[P_{A,t+1}] = p_A$, where p_A is calculated at time (generation) t. That is, p_A is the probability of selecting a single individual copy of rule $\mathbf{A}$ for reproduction from a particular population with $n_A = NP_{A,t}$ copies of rule $\mathbf{A}$ and

$n_B = N(1 - P_{A,t})$ copies of rule **B**. So in general

$$E[P_{A,t+1}] = p_A = \frac{n_A f_A}{n_A f_A + (1 - n_A) f_B}. \tag{5.1}$$

Dividing numerator and denominator by f_B, and substituting $r_f \equiv \frac{f_A}{f_B}$ yields

$$E[P_{A,t+1}] = \frac{n_A r_f}{n_A r_f + (N - n_A)}. \tag{5.2}$$

Then dividing numerator and denominator by population size N gives

$$E[P_{A,t+1}] = \frac{P_{A,t}\, r_f}{P_{A,t}(r_f - 1) + 1}. \tag{5.3}$$

Now we make the major assumption that is key to this type of analysis: we assume that $E[P_{A,t+1}] \approx P_{A,t+1}$. That is, we allow ourselves to plug the expected proportion for the next generation back into our Equation 5.3 to get the expected proportion for the $t+2$ generation, and so on. But $E[P_{A,t+1}] \approx P_{A,t+1}$ is a strict equality only for deterministic selection operators or for infinite populations (Vose, 1993). However, we claim to be modeling our finite population, stochastic selection GA here. So we will use $E[P_{A,t+1}]$ for $P_{A,t+1}$, and derive a simple linear recurrence relation on $P_{A,t}$:

$$P_{A,t+1} = \frac{P_{A,t}\, r_f}{P_{A,t}(r_f - 1) + 1}. \tag{5.4}$$

5.2.2 The Convergence Equation

To solve this recurrence relation, let us first divide numerator and denominator on the left by $P_{A,t}$, then invert both sides of the equation:

$$\frac{1}{P_{A,t+1}} = \frac{r_f - 1 + \frac{1}{P_{A,t}}}{r_f}. \tag{5.5}$$

Rearranging,

$$\frac{1}{P_{A,t+1}} = \frac{r_f - 1}{r_f} + \frac{1}{r_f}\left(\frac{1}{P_{A,t}}\right). \tag{5.6}$$

Now rather than solving for $P_{A,t}$, let's solve for the inverse, $P'_t \equiv \frac{1}{P_{A,t}}$:

$$P'_{t+1} = \frac{r_f - 1}{r_f} + \frac{1}{r_f} P'_t. \tag{5.7}$$

The above is straightforward to solve:

$$P'_t = \frac{r_f^t - 1 + P'_0}{r_f^t}. \tag{5.8}$$

Replacing P'_t and P'_0 and inverting both sides results in

$$P_{A,t} = \frac{r_f^t}{r_f^t - 1 + \frac{1}{P_{A,0}}}. \tag{5.9}$$

Multiplying top and bottom by $P_{A,0}$, we get

$$P_{A,t} = \frac{P_{A,0} r_f^t}{P_{A,0} r_f^t - P_{A,0} + 1}, \tag{5.10}$$

which is equivalent to Equation 7 of (Goldberg & Deb, 1991) with only $k = 2$ types of individuals (i.e., niches).

In Figure 5.1 we plot an example expected convergence using Equation 5.10 with a fitness ratio of $r_f = 2$ so that the population quickly converges to all **A**s. The initial proportion is $P_{A,0} = \frac{1}{100}$ (e.g., for a population size $N = 100$ and only a single **A** in the initial population).

5.2.3 Verifying the Convergence Equation

In the interests of time and space, we leave the verification of Equation 5.10 to future work, and rather discuss below two methods of verification.

Note that Equation 5.10 and hence Figure 5.1 rely on the assumption $E[P_{A,t+1}] \approx P_{A,t+1}$, so that they are *models* of the actual expected convergence. To compare the model and the actual, we could sample the actual by running a simple GA over many random initial populations, taking the average population distribution for each generation. Or we could simply apply the full Markov chain t times to a random initial distribution. This would result in a probability distribution over all possible states. We could then compute from this distribution the expected number (or proportion) of **A**s at that generation t. This value would be the EXACT (actual)

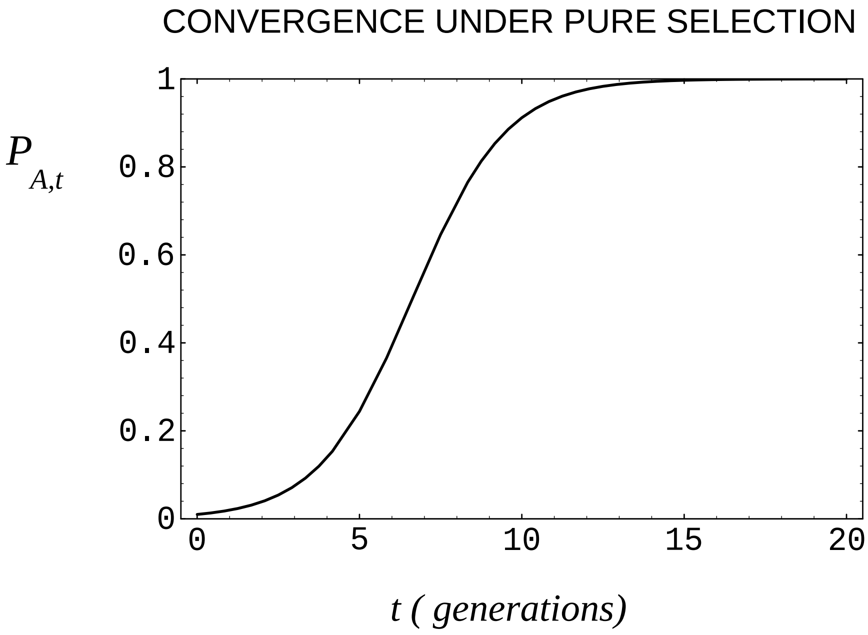

Figure 5.1: Expected proportion of **A**s in a simple GA under proportionate selection. Here $r_f = 2$, $N = 100$, and the initial proportion $P_{A,0} = \frac{1}{N}$.

expected number (proportion) of **A**s. That is, as we increase the number of independent runs (samples), we should approach the Markov values. Applying the Markov chain t times means t matrix multiplications, which can be fairly expensive computationally for larger N.

5.2.4 Convergence Times

We can solve Equation 5.10 for time (generations) t:

$$t = \frac{1}{\ln r_f}(\ln(\frac{P_{A,t}}{P_{A,0}}) + \ln(\frac{1 - P_{A,0}}{1 - P_{A,t}})). \tag{5.11}$$

We can see from this equation that the time to reach proportion $P_{A,t}$ increases rapidly for extreme proportions (i.e., $P_{A,t}$ near 1 and/or $P_{A,0}$ near 0), as our intuition suggests. Very small initial starting proportions imply slow initial growth, while proportions arbitrarily close to one imply arbitrarily large population sizes. In addition, increasing fitness difference speeds up convergence time, again in agreement with our intuition.

Equation 5.11 is written in terms of proportions, and so is general to all population sizes, including infinite N. To add in the effects of finite populations, let's assume population size

N and look for the time to go from a single copy of **A**, which is the worst case (i.e., longest time), to *almost* (one less than) full convergence[1]. In other words, we set $P_{A,0} = \frac{1}{N}$ and $P_{A,t_{conv}} = \frac{N-1}{N} = 1 - \frac{1}{N}$. Plugging these into Equation 5.11, and simplifying, we find that

$$t_{conv} = \frac{2}{\ln r_f} \ln(N-1). \tag{5.12}$$

Clearly then convergence time in the simple GA grows logarithmically in population size. But convergence time here in the "selective case" is "takeover" time (that is, absorption time), for a single species to take over the entire population. Will convergence time for niching, in which the evolving population arrives at some non-uniform population equilibrium, be different?

5.3 Perfect Sharing Convergence

We already know that the case of perfect sharing is special under proportionate selection. Sharing (with no overlap) allocates objective fitness to individuals in proportion to the numbers of individuals, while proportionate selection allocates numbers of individuals in proportion to their objective fitnesses. The result is an "instantaneous" (one generation) transition to the niching equilibrium, in expectation. Using Equation 5.1 above,

$$E[P_{A,t+1}] = p_A = \frac{n_A f_A}{n_A f_A + n_B f_B}, \tag{5.13}$$

but substituting the shared fitnesses $f_{A,sh} = \frac{f_A}{n_A}$ and $f_{B,sh} = \frac{f_B}{n_B}$ for the objective fitnesses f_A and f_B,

$$E[P_{A,t+1}] = \frac{n_A \frac{f_A}{n_A}}{n_A \frac{f_A}{n_A} + n_B \frac{f_B}{n_B}}, \tag{5.14}$$

results in

$$E[P_{A,t+1}] = \frac{f_A}{f_A + f_B} = \frac{r_f}{r_f + 1} = P_{A,eq}.$$

Thus we expect to get to equilibrium in one time step, no matter in what state $P_{A,0}$ we begin[2]. So $t_{conv} = 1$ for perfect sharing.

[1]Note that according to Equation 5.10, the infinite population asymptotically approaches uniform convergence, therefore our finite model will asymptotically approach N. So we consider convergence to be $N-1$ to allow us to compute a finite convergence time t.

[2]Unless $n_A = 0$ or $n_B = 0$ in which case we are starting in an absorbing state.

5.4 Resource Sharing Convergence

Next we examine convergence time for resource sharing, saving the case of fitness sharing for later, because, as it turns out, resource sharing is easier to analyze than fitness sharing.

5.4.1 The Convergence Equation

We retrieve our formula for the probability of selecting an **A** from our Markov model for resource sharing in the last chapter, specifically Equation 4.23:

$$p_A = \frac{1 - r_o + r_o \frac{n_A}{N}}{1 - r_o + \frac{1}{r_f}}. \tag{5.15}$$

Substituting p_A into Equation 5.1 above,

$$E[P_{A,t+1}] = p_A = \frac{1 - r_o + r_o \frac{n_A}{N}}{1 - r_o + \frac{1}{r_f}}, \tag{5.16}$$

and noting that $P_{A,t} = \frac{n_A}{N}$ at time t, we find that

$$E[P_{A,t+1}] = \frac{1 - r_o + r_o \frac{n_A}{N}}{1 - r_o + \frac{1}{r_f}} = \frac{1 - r_o + r_o P_{A,t}}{1 - r_o + \frac{1}{r_f}}.$$

Rearranging, we find that

$$E[P_{A,t+1}] = \frac{1 - r_o}{1 - r_o + \frac{1}{r_f}} + \frac{r_o}{1 - r_o + \frac{1}{r_f}} P_{A,t}.$$

Now we make our major assumption that $E[P_{A,t+1}] \approx P_{A,t+1}$ and the above becomes a simple linear recurrence relation on $P_{A,t}$. Defining $P_{A,0}$ to be the initial population, we solve the difference equation and rearrange to get:

$$P_{A,t} = \frac{1 - r_o}{1 - 2r_o + \frac{1}{r_f}} + \left(P_{A,0} - \frac{1 - r_o}{1 - 2r_o + \frac{1}{r_f}} \right) \left(\frac{r_o}{\frac{1}{r_f} - r_o + 1} \right)^t. \tag{5.17}$$

Remembering that at equilibrium (steady-state) $P_{A,eq} = \frac{1-r_o}{1-2r_o+\frac{1}{r_f}}$ (Equation 4.12), and introducing $\beta \equiv \frac{r_o}{\frac{1}{r_f}-r_o+1}$, we simplify to

$$P_{A,t} = P_{A,eq} - (P_{A,eq} - P_{A,0})\beta^t$$

or

$$P_{A,t} = P_{A,eq}(1-\beta^t) + P_{A,0}\beta^t. \tag{5.18}$$

We note that $\beta \leq 1$. This must be so as the numerator of β, $r_o \equiv \frac{f_{AB}}{f_A}$, must always be ≤ 1, while the denominator must always be ≥ 1, since $f_{AB} \leq f_B \Rightarrow \frac{f_{AB}}{f_A} \leq \frac{f_B}{f_A} \Rightarrow r_o \leq \frac{1}{r_f}$, and so $1 \leq \frac{1}{r_f} - r_o + 1$. In general, $\beta < 1$ and Equation 5.18 illustrates the exponential decay of the initial state's effect on the current population, and the corresponding exponential growth of the "long term" steady state's influence.

The thick line plotted in Figure 5.2 illustrates a typical convergence graph. With fitness ratio $r_f = 2$ and overlap ratio $r_o = \frac{1}{3}$, the equilibrium proportion $P_{A,eq} = \frac{1-r_o}{1-2r_o+\frac{1}{r_f}} = \frac{4}{5} = 0.8$. In Figure 5.2 we see very fast convergence to this equilibrium.

To see the effect of varying the degree of overlap, Figure 5.2 shows convergence curves for a number of different r_o values. The lowest curve corresponds to $r_o = \frac{1}{10}$, very little overlap, in which we expect to see almost instantaneous (i.e., one generation) convergence to steady-state ($P_{A,eq}$). The other two curves show how increasing overlap increases the convergence time but also changes the equilibrium point. At maximum overlap, $r_o = \frac{1}{r_f} = \frac{1}{2}$, we see convergence to a population of all **A**s.

In the degenerate case of $f_A = f_B = f_{AB}$, the two species/rules **A** and **B** are identical, completely overlapping, and Equation 5.18 reduces to a constant, $P_{A,0}$, indicating no selective pressure away from the initial population distribution. This is the case of *genetic drift*, in which equations of expectation (e.g., Equation 5.18) are least useful, failing to describe the variance of the selection operator. In the case of genetic drift, this variance dominates the dynamics, and leads to eventual loss of one or the other species.

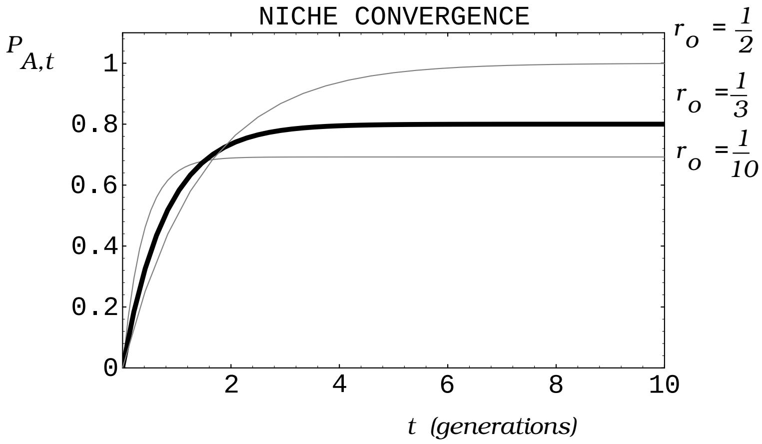

Figure 5.2: Expected convergence, under resource sharing, with varying overlap: $r_o = 1/10$, $r_o = 1/3$, and $r_o = 1/2$ (complete overlap). Fitness ($r_f = 2$) is the same for all plots.

5.4.2 Convergence Times

Here we derive a useful expression for two-niche convergence times by solving Equation 5.18 for time t (in generations). Rearranging Equation 5.18, we move t to one side:

$$\beta^t = \frac{P_{A,t} - P_{A,eq}}{P_{A,0} - P_{A,eq}}.$$

Taking the logarithm of both sides and solving for t, yields:

$$t = \frac{\ln(\frac{P_{A,eq} - P_{A,t}}{P_{A,eq} - P_{A,0}})}{\ln \beta}. \tag{5.19}$$

In general $P_{A,t}$ approaches $P_{A,eq}$ asymptotically with t. More practically, we introduce the discrete nature of a finite population of size N, by asking for the time it takes to go from one copy of **A** ($P_{A,0} = \frac{1}{N}$) to within at most one individual ($\frac{1}{N}$) of the equilibrium population ($P_{A,t} = P_{A,eq} - \frac{1}{N}$):

$$t = \frac{\ln(\frac{\frac{1}{N}}{P_{A,eq} - \frac{1}{N}})}{\ln \beta}.$$

Simplifying yields:

$$t_{conv} = \frac{-\ln(P_{A,eq}N - 1)}{\ln \beta}. \tag{5.20}$$

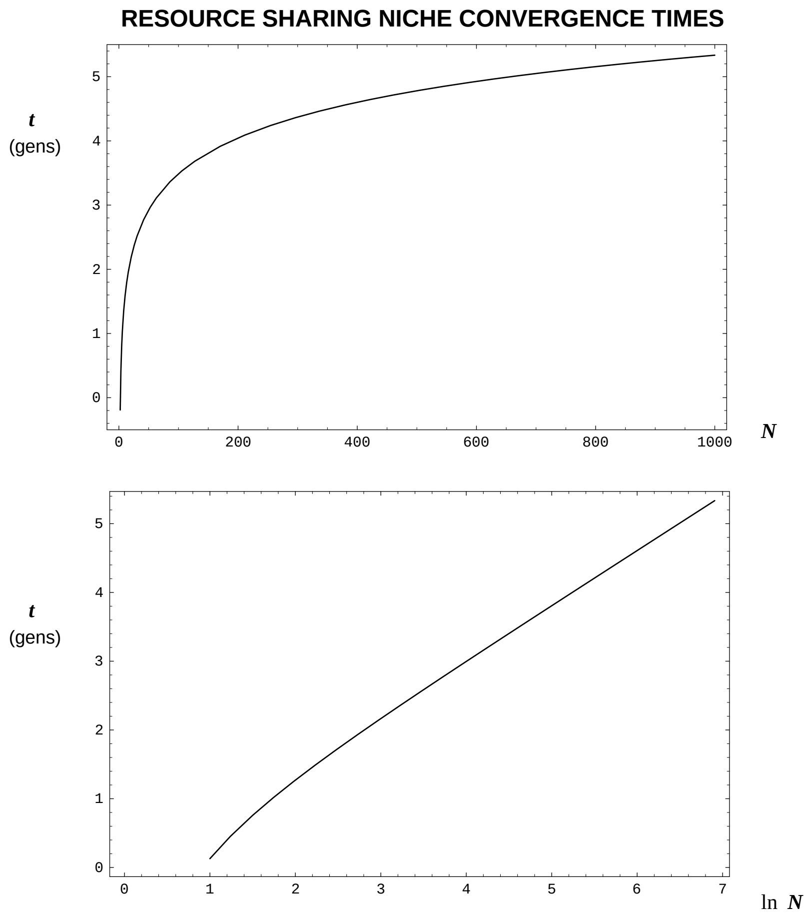

Figure 5.3: Expected niche convergence time for resource sharing grows logarithmically in population size N.

We can see immediately that expected convergence time grows logarithmically in population size N, as we suspected. Figure 5.3 illustrates a typical expected growth in convergence time with increasing N. Here $r_f = 2$ and $r_o = \frac{1}{3}$ (remember that β is a function of r_f and r_o).

5.5 Fitness Sharing Convergence

The fitness sharing convergence results are more difficult to obtain.

5.5.1 The Convergence Equation

First we try the usual approach from above. We start with the probability of selecting an **A** under fitness sharing,

$$p_A = \frac{n_A f_{A,sh}}{n_A f_{A,sh} + (n_B) f_{B,sh})}. \tag{5.21}$$

We recall that **A**'s shared fitness $f_{A,sh}$ is its objective fitness f_A divided by its niche count, which is the sum of the sharing function values over all N population members. But since we only have two types of individuals, **A** and **B**, we only have two possible contributions to the niche count: 1 for each **A** and $(1 - \frac{d(A,B)}{\sigma_{sh}})$ for each **B**:

$$f_{A,sh} = \frac{f_A}{n_A + n_B(1 - \frac{d(A,B)}{\sigma_{sh}})}.$$

Earlier we normalized $d(A, B)$ to one, so

$$f_{A,sh} = \frac{f_A}{n_A + n_B(1 - \frac{1}{\sigma_{sh}})}.$$

Furthermore, let's define $\mathcal{S} \equiv (1 - \frac{1}{\sigma_{sh}})$ (that is, the sharing function Sh). Now we can rewrite the probability of selecting an **A** under fitness sharing:

$$p_A = \frac{n_A \frac{f_A}{n_A + n_B \mathcal{S}}}{n_A \frac{f_A}{n_A + (N - n_A)\mathcal{S}} + n_B \frac{f_B}{n_B + n_A \mathcal{S}}}. \tag{5.22}$$

Dividing numerator and denominator by N to get proportions and by f_B to get the fitness ratio, then substituting p_A into Equation 5.1 above,

$$E[P_{A,t+1}] = p_A = \frac{\frac{P_{A,t} r_f}{P_{A,t} + (1 - P_{A,t})\mathcal{S}}}{\frac{P_{A,t} r_f}{P_{A,t} + (1 - P_{A,t})\mathcal{S}} + \frac{1 - P_{A,t}}{1 - P_{A,t} + P_{A,t}\mathcal{S}}}. \tag{5.23}$$

This is a rather messy equation that can't be simplified much nor can it be solved (as a recurrence relation) directly for $P_{A,t}$. However, we can solve for something just as useful, the ratio of **A**s to **B**s: $\frac{n_A}{n_B}$. Note that the ratio of numbers is the same as the ratio of proportions: $\frac{P_{A,t}}{P_{B,t}} = \frac{n_A}{n_B}$. We'll name the proportion ratio $P_{r,t} \equiv \frac{P_{A,t}}{P_{B,t}}$ for short. Now since $P_{r,t+1} = \frac{P_{A,t+1}}{P_{B,t+1}} = \frac{P_{A,t+1}}{1 - P_{A,t+1}}$, we can just divide both sides of Equation 5.23 above by $(1 - P_{A,t+1})$. On the left hand side this gives us:

$$\frac{E[P_{A,t+1}]}{1-P_{A,t+1}} = \frac{P_{A,t+1}}{1-P_{A,t+1}} = P_{r,t+1} = E[P_{r,t+1}].$$

But before writing out the entire right hand side, we notice that the right hand side of Equation 5.23 is of the form $\frac{a}{a+b}$, so when we divide it by $1-\frac{a}{a+b}$, we'll have:

$$\frac{\frac{a}{a+b}}{1-\frac{a}{a+b}} = (\frac{a}{a+b})(\frac{a+b}{a+b-a}) = \frac{a}{b}.$$

Remembering what a and b stand for, and putting our left hand side $E[P_{r,t+1}]$ back in,

$$E[P_{r,t+1}] = \frac{\frac{P_{A,t}\, r_f}{P_{A,t}+(1-P_{A,t})\mathcal{S}}}{\frac{1-P_{A,t}}{1-P_{A,t}+P_{A,t}\mathcal{S}}}. \tag{5.24}$$

Making our key assumption that $E[P_{r,t+1}] \approx P_{r,t}$ and rearranging,

$$P_{r,t+1} = \frac{P_{A,t}\, r_f}{P_{A,t}+(1-P_{A,t})\mathcal{S}} \frac{1-P_{A,t}}{1-P_{A,t}+P_{A,t}\mathcal{S}}. \tag{5.25}$$

Dividing through by $1-P_{A,t}$, and remembering our defined ratio of proportions $P_{r,t} \equiv \frac{P_{A,t}}{1-P_{A,t}}$, gives

$$P_{r,t+1} = r_f \frac{P_{r,t}(1+P_{r,t}\,\mathcal{S})}{P_{r,t}+\mathcal{S}}. \tag{5.26}$$

This can be rewritten a bit more succinctly by dividing numerator and denominator by $P_{r,t}$:

$$P_{r,t+1} = r_f \frac{1+P_{r,t}\,\mathcal{S}}{1+\frac{\mathcal{S}}{P_{r,t}}}. \tag{5.27}$$

Equation 5.27 appears simple in form, but unfortunately it is difficult to solve exactly. To obtain a closed-form, we use the continuous approximation method. That is, we set up a differential equation approximating the difference equation 5.27 and then try to integrate the differential equation. Subtracting $P_{r,t}$ from both sides of Equation 5.27 above, we obtain

$$P_{r,t+1} - P_{r,t} = r_f \frac{1+P_{r,t}\,\mathcal{S}}{1+\frac{\mathcal{S}}{P_{r,t}}} - P_{r,t},$$

which we can then approximate by the differential equation

$$\frac{dP_{r,t}}{dt} = r_f \frac{1 + P_{r,t}\mathcal{S}}{1 + \frac{\mathcal{S}}{P_{r,t}}} - P_{r,t},$$

which upon integration yields

$$t + \frac{\mathcal{S}\ln P_{r,t}}{\mathcal{S} - r_f} + \frac{(\mathcal{S}^2 r_f - r_f)\ln(\mathcal{S} - r_f + P_{r,t} - \mathcal{S} r_f P_{r,t})}{\mathcal{S} - r_f - \mathcal{S}^2 r_f + \mathcal{S} r_f^2} = C. \tag{5.28}$$

We can solve for the constant C if we substitute the initial proportion ratio $P_{r,0}$ for the first generation ($t = 0$):

$$C = 0 + \frac{\mathcal{S}\ln P_{r,t}}{\mathcal{S} - r_f} + \frac{(\mathcal{S}^2 r_f - r_f)\ln(\mathcal{S} - r_f + P_{r,t} - \mathcal{S} r_f P_{r,t})}{\mathcal{S} - r_f - \mathcal{S}^2 r_f + \mathcal{S} r_f^2}.$$

Plugging this C back into Equation 5.28, we can solve for t as a function of proportion ratio "desired", $P_{r,t}$, (as well as of the initial proportion ratio $P_{r,0}$):

$$t[P_{r,t}] = \frac{\ln(\frac{P_{r,0}}{P_{r,t}})}{1 - \frac{r_f}{\mathcal{S}}} + \frac{(\mathcal{S} - \frac{1}{\mathcal{S}})\ln(\frac{\mathcal{S} - r_f + P_{r,0} - \mathcal{S} r_f P_{r,0}}{\mathcal{S} - r_f + P_{r,t} - \mathcal{S} r_f P_{r,t}})}{1 + r_f - \mathcal{S} - \frac{1}{\mathcal{S}}}. \tag{5.29}$$

So now we have in Equation 5.29 a closed-form equation relating time (generations) to proportion ratio ($P_{r,t} = \frac{P_A}{P_B}$). Unfortunately, we cannot solve this equation generally for $P_{r,t}$ in terms of t, but we can always solve it numerically for any particular values of t, r_f, and $P_{r,0}$. For now we compare our continuous approximation, Equation 5.29, to our more accurate discrete model, Equation 5.27, in Figure 5.4. We assume equal fitness $r_f = 1$, and an overlap ($\sigma_{sh} = 2$ so that $\mathcal{S} = 1 - \frac{d}{\sigma_{sh}} = 0.5$). Note how the population asymptotically approaches the niching equilibrium proportion ratio ($\frac{n_A}{n_B} = P_{r,eq} = 1$). Note also how the curve of this plot seems the same in form to convergence to uniform populations in the simple GA (Figure 5.1), while the plot for convergence to niching equilibrium under resource sharing (Figure 5.2) has a qualitatively different shape.

5.5.2 Convergence Times

We derive a useful expression for two-niche convergence times for fitness sharing. Equation 5.29 is already solved for time t (in generations). But since it is an equation strictly in terms of

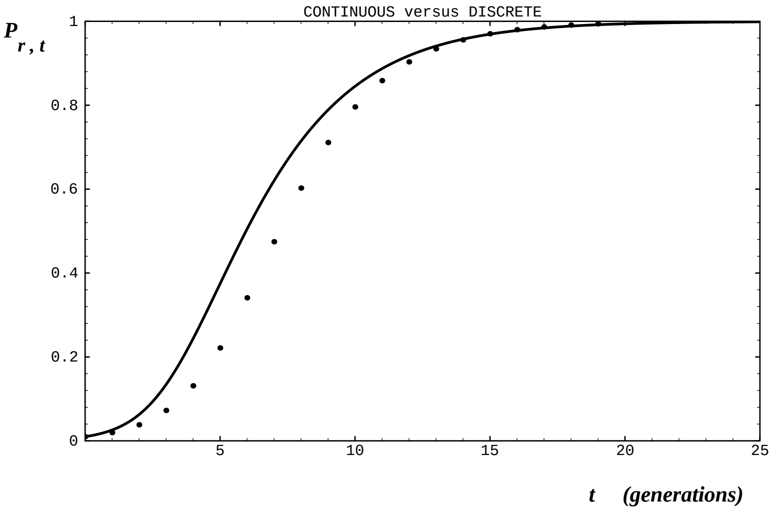

Figure 5.4: Expected convergence under fitness sharing. $P_{r,t}$ is the ratio of n_A to n_B at time t. The predictions of the discrete model (difference equation) are the solid dots, while the solid line is the prediction of the continuous model. Here fitness $r_f = 1$, so that equilibrium is a population of half **A**s and half **B**s, or $P_{r,eq} = 1$. We see the population proportion ratio asymptotically approaches this. Note: we assumed an overlap of $\sigma_{sh} = 2$ so that $\mathcal{S} = 1 - \frac{d}{\sigma_{sh}} = 0.5$. Also, we assume an initial starting proportion $P_{r,0} = 0.01$, which might be, for example, a single copy of **A** in a population of $N = 100$.

proportions, it models asymptotic convergence. By using proportions, it implicitly models infinite (or rather arbitrarily large) population sizes. We now introduce into Equation 5.29 the discrete nature of a finite population size N by defining the two proportion ratios $P_{r,0}$ (the initial proportions) and $P_{r,t}$ (the proportions at time t). We'll model the bounding case of growing from a single copy of species **A** to within one copy of the niching equilibrium.

We define the initial proportion as follows. With $n_{A,0} = 1$, the initial proportion of **A**s is $P_{A,0} = \frac{1}{N}$, and $P_{B,t} = \frac{N-1}{N}$. It follows that the initial proportion *ratio* $P_{r,0} = \frac{P_{A,0}}{P_{B,0}} = \frac{1}{N} / \frac{N-1}{N} = \frac{1}{N-1}$.

At equilibrium, we recall Equation 4.7:

$$P_{A,eq} = \frac{r_f \sigma_{sh} - \sigma_{sh} + 1}{1 + r_f}. \tag{5.30}$$

First we substitute our sharing function $\mathcal{S} = 1 - \frac{1}{\sigma_{sh}}$ for σ_{sh} using $\sigma_{sh} = \frac{1}{1-\mathcal{S}}$:

$$P_{A,eq} = \frac{\frac{r_f - 1}{1-\mathcal{S}} + 1}{r_f + 1}.$$

When we are just one copy of **A** less than the equilibrium:

$$P'_{r,eq} = \frac{P_{A,eq} - \frac{1}{N}}{P_{B,eq} + \frac{1}{N}} = \frac{P_{A,eq} - \frac{1}{N}}{1 - P_{A,eq} + \frac{1}{N}},$$

where $P'_{r,eq}$ is our symbol for *near* (within one of) equilibrium. Now we can substitute in $P_{A,eq}$ from Equation 5.30 and simplify:

$$P'_{r,eq} = \frac{N\frac{r_f - 1}{1-\mathcal{S}} + N - r_f - 1}{(N+1)(r_f+1) + N\frac{1-r_f}{1-\mathcal{S}} - N}. \tag{5.31}$$

So we want to know the expected time to go from $P_{r,0} = \frac{1}{N-1}$ to $P_{r,t} = P'_{r,eq}$ in Equation 5.31 above. Plugging these two proportion ratios into $P_{r,0}$ and $P_{r,t}$ into Equation 5.29 above, and solving and simplifying, yields:

$$\begin{aligned} t_{conv} \quad = \quad & \frac{(\mathcal{S}^2-1)}{r_f\mathcal{S}-\mathcal{S}^2+\mathcal{S}-1} \ln\Big(\frac{(\frac{1}{N-1}-r_f+\mathcal{S}+\frac{r_f\mathcal{S}}{1-N})(Nr_f\mathcal{S}+r_f\mathcal{S}+\mathcal{S}-r_f-N-1)}{(r_f+1)^2(\mathcal{S}-1)^2}\Big) \\ & + \frac{\mathcal{S}}{\mathcal{S}-r_f} \ln\Big(\frac{N-Nr_f\mathcal{S}-r_f\mathcal{S}-\mathcal{S}+r_f+1}{(N-1)(r_f\mathcal{S}-N\mathcal{S}+\mathcal{S}+Nr_f-r_f-1)}\Big). \end{aligned} \tag{5.32}$$

Although not a very simple formula, this expression for expected niche convergence time is clearly logarithmic in population size N, just as we have predicted and just as we have already shown for resource sharing. Figure 5.5 illustrates a typical expected growth in convergence time with increasing N from 1 to 1000. Here we plot Equation 5.32 with $r_f = 1$ and $\mathcal{S} = \frac{1}{2} \Rightarrow \sigma_{sh} = \frac{1}{2}$. The growth in time does appear to be logarithmic.

5.6 A Control Map For Niching: Cooperation versus Competition

We have shown in this chapter that convergence to niching equilibrium is fast, behaving much like convergence in the simple GA under selection pressure alone. Both convergence times, for the simple GA and for sharing, grow logarithmically in N, population size. This is in sharp contrast to the very long niche maintenance times calculated in chapter 4, which grow very rapidly (exponentially) in N. But with both niche maintenance and niche convergence, performance degrades with increasing niche overlap, objective fitness difference, and decreasing population size. As the general niching conditions become less favorable, the difference between

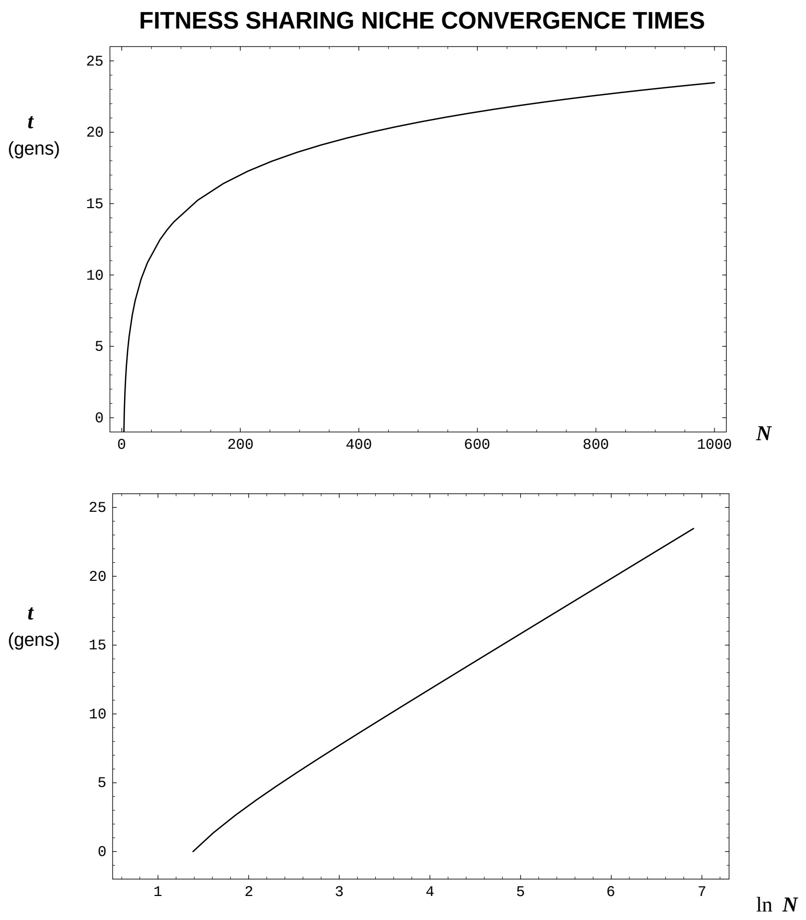

Figure 5.5: Expected convergence times under fitness sharing grow logarithmically in N.

niche convergence time and expected niche loss (maintenance) time narrows. Again we face a blurry area between cooperating species (i.e., long niche maintenance times) and competing species (e.g., very short maintenance times).

In this section we use the results of chapters 4 and 5 to further define the elusive but critical boundary between competition and cooperation. We develop a preliminary version of a "control map" to predict the bounds of niching success and failure, comparing theoretical boundaries with empirical results, and discuss the implications and applications of such control maps.

5.6.1 Comparing Niche Maintenance Times to Niche Convergence Times

In this subsection we make an initial attempt to bring the above two timing results together. In Figure 5.6 we illustrate a simple way of using these two results together. The upper curve plots the expected niche extinction times (using the approximations) as a function of population size. The lower curve is the expected niche convergence time. Here again the fitness ratio $r_f = 2$, but now the overlap ratio is relatively high, $r_o = 0.45$. This overlap means that $r_f * r_o = 90\%$ of **B**'s covered examples are also covered by **A**. This relatively high ratio of overlap was chosen to illustrate clearly the predicted niching failure at low population sizes (e.g., $N < 20$). At high enough population sizes (e.g., $30 < N$), the difference between the niche convergence and niche extinction times is quite high.

Characterizing the critical phase transition from poor niche maintenance to robust niche maintenance as population size increases, is tricky. The niche convergence model (lower curve) breaks down for small population sizes, while the niche extinction time approximation (upper curve) loses accuracy with increasing overlap. Yet our initial models above give us clear indication that sharing works well *within bounds*. And apparently we have some hope of defining those boundaries. In the following subsections we attempt to find these boundaries.

5.6.2 Equating Niche Maintenance and Niche Convergence Times

In the previous chapter we found it difficult to use the single measure of niche maintenance time to distinguish "good" niching (i.e., cooperation) from "bad" (i.e., competition). Perhaps with niche convergence times in hand we can better go after that boundary. In Figure 5.6 we saw how small N could bring t_{abs} and t_{conv} together, while large enough N could mean fast

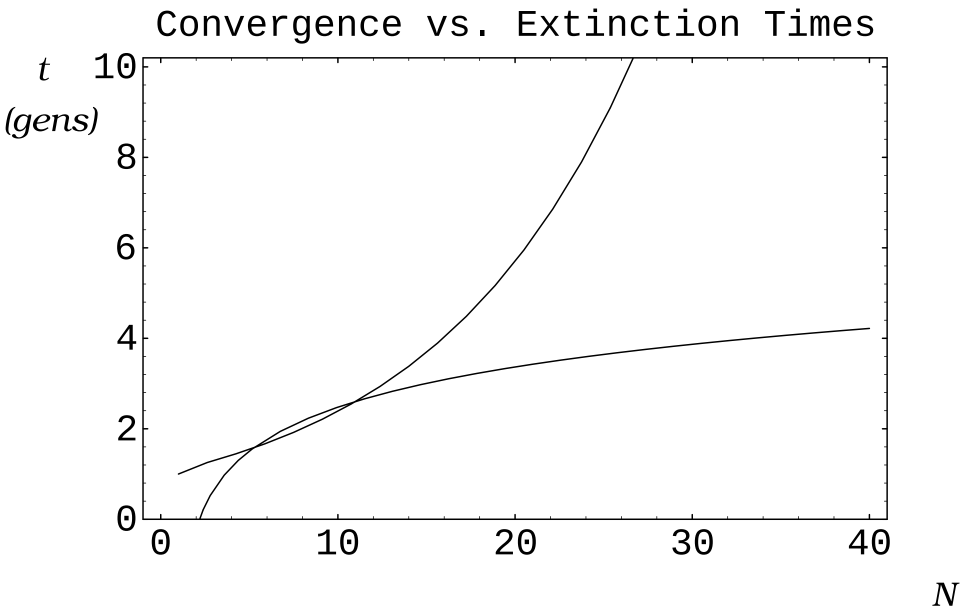

Figure 5.6: Expected niche extinction times (upper curve) versus expected niche convergence times (lower curve). Here fitness ratio $r_f = 2$ with very high overlap $r_o = 0.45$ (near maximum).

t_{conv} and long t_{abs}. Perhaps if we equate t_{abs} and t_{conv}, we can find a meaningful boundary between cooperative pairs of niches and competitive pairs.

To put it in more concrete terms, note that t_{conv} and t_{abs} are functions of overlap, fitness ratio, and population size. (We assume resource sharing for now and use its notation.) Thus $t_{conv}(r_o, r_f, N)$ and $t_{abs}(r_o, r_f, N)$ denote functions. The region defined by $t_{conv} = t_{abs}$ is a surface through the three dimensional space $< r_o, r_f, N >$. To aid visualization, we fix N and get a two-dimensional space with $t_{conv} = t_{abs}$ now defining a line. On one side of that line, $t_{conv} > t_{abs}$, and niching fails, while on the other side $t_{conv} \ll t_{abs}$ and niching succeeds.

With all of the noise in the stochastic operators of a GA, expecting a crisp line is unrealistic. But we could easily define a boundary *region* as the space between two crisp bounds. One bound might be defined as $c_{succ} t_{conv} = t_{abs}$, where $c_{succ} > 1$ is some constant representing how much longer than t_{conv} we expect t_{abs} to be in order to predict niching success. Thus increasing c_{succ} raises our requirement for cooperation. Niche maintenance times for successfully cooperating species with overlapping niches would have to be at least c_{succ} times their t_{conv}. Similarly we might define another bound as $c_{fail} t_{conv} = t_{abs}$ (where c_{fail} might be < 1). Thus for two niches to be considered truly competing, such that we can count on a quick resolution to their conflict

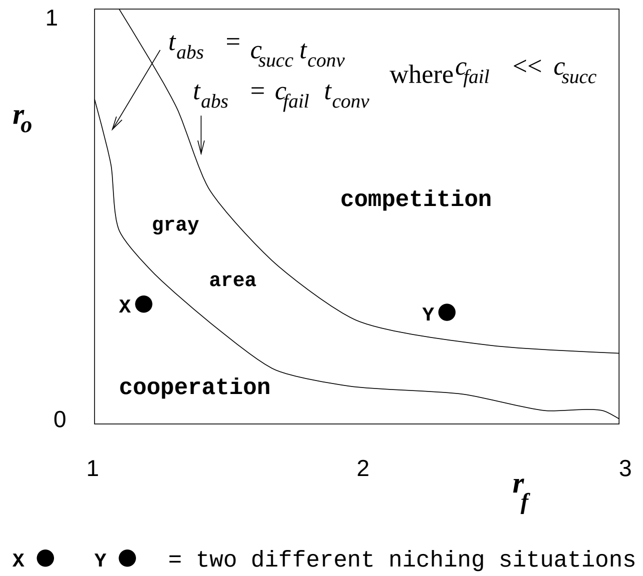

Figure 5.7: Speculative *cooperative-competitive* boundary for resource sharing given population size N, found by setting $c\, t_{conv} = t_{abs}$.

and a single emergent winner, their expected niche loss time must be no more than c_{fail} times longer than their expected convergence time. Between the two upper and lower boundary lines would be a gray area of niching scenarios, in which we could not predict with much confidence if both niches would be maintained, or if a single clear winner would emerge. But above the boundary region we could say with some confidence (and perhaps a quantifiable amount) that such niche pairs are cooperative and would be maintained, while below the boundary region we could say with confidence that niches in such pairs are competitive, and that only one of each pair will win and be represented in the population.

Figure 5.7 gives a hypothetical example. Here each possible niching situation is a coordinate in the space $< r_o, r_f >$, since we have fixed N to some value. We would like to able to divide the space into cooperative and competitive regions. We could then adjust GA and niching parameters, such as the exponent of fitness scaling, or some tolerance of niche overlap (e.g., σ_{sh} or α_{sh}), or population size N, etc., to get the cooperative-competitive boundary that we want. We could then say, "yes, pair X should be considered cooperative; I'd want them both. But pair Y should be treated as a competitive relationship; I want only the better species." (See Figure 5.7.)

5.6.3 An Example: A Control Map for Resource Sharing

In this subsection we develop theoretical bounds for niching under resource sharing[3], by solving the equation $c\, t_{conv} = t_{abs}$, and then using large and small values of c for success and failure bounds respectively. Retrieving t_{abs} from chapter 4, Equation 4.32, and t_{conv} from earlier in this chapter, Equation 5.20:

$$c\, t_{conv} = t_{abs},$$

$$c\,\frac{-\ln(P_{A,eq}N-1)}{\ln\beta} = \frac{(1-2r_o+\frac{1}{r_f})^N}{(1-r_o)^N+(\frac{1}{r_f}-r_o)^N},$$

$$c\,\frac{-\ln(\frac{1-r_o}{1-2r_o+\frac{1}{r_f}}N-1)}{\ln(\frac{r_o}{\frac{1}{r_f}-r_o+1})} = \frac{(1-2r_o+\frac{1}{r_f})^N}{(1-r_o)^N+(\frac{1}{r_f}-r_o)^N}. \tag{5.33}$$

Before proceeding to solve the above equation (for r_o as a function of r_f and N for example), we pause to simplify the notation. Recall that we defined the ratio of overlap r_o as $r_o \equiv \frac{f_{AB}}{f_A}$; that is, the ratio of the amount of overlap, f_{AB}, to the fitness f_A of the better niche **A**. But since **A** is the better niche, and hence $f_A \geq f_B$, then r_o must always vary between 0 and $\frac{1}{r_f}$, since f_{AB} reaches its maximum at $f_{AB} = f_B$, as we noted earlier. Thus the upper limit ratio r_o varies with the ratio of fitness r_f. For the sake of our analysis, however, it is desirable to have the ratio of overlap vary from 0 to 1 always, no matter what r_f is. This will allow for easier algebraic manipulation and simplified visualization. We therefore introduce another ratio of overlap: $r_{ov} \equiv \frac{f_{AB}}{f_B}$. Note that r_{ov} varies from 0 to 1. Also note that $r_{ov} = \frac{f_{AB}}{f_B} = \frac{f_{AB}}{f_A}\frac{f_A}{f_B} = r_o r_f$. Thus $r_o = \frac{r_{ov}}{r_f}$, and we can simply substitute $\frac{r_{ov}}{r_f}$ for r_o in Equation 5.33 above:

$$c\,\frac{-\ln(\frac{r_f-r_{ov}}{r_f-2r_{ov}+1}N-1)}{\ln(\frac{r_{ov}}{1-r_{ov}+r_f})} = \frac{(r_f-2r_{ov}+1)^N}{(r_f-r_{ov})^N+(1-r_{ov})^N}. \tag{5.34}$$

Solving Equation 5.34 for a closed-form expression is difficult. Even after fixing population size N, we would still find it difficult to solve the above for r_{ov} as a function of r_f, or for r_f as a function of r_{ov}. With one side of the equation being logarithmic and the other being polynomial, it is not straightforward to solve such equations for closed-form expressions. We could turn to transcendental functions or approximations for closed-form solutions. For now,

[3]Note that the case of resource sharing is difficult enough, algebraicly. Recall that fitness sharing in general gives us even more complicated expressions for t_{conv}, and t_{abs}.

we use numerical solutions obtained via a computer software package for mathematics (namely, Mathematica[4] and its "FindRoot" function). That is, given a specific population size N and fitness ratio r_f, we numerically solve for r_{ov} (using Mathematica's "FindRoot" and giving it a starting point of $r_{ov} = 0.9999$).

Numerically solving Equation 5.34 for r_{ov}, over a series of r_f values and a fixed N, we can interpolate a bound (for a given c). For example, if we assume a population size $N = 50$, a constant $c = 10$, and then vary r_f from 1 to 4, we can plot r_{ov} as a function of r_f (by sampling r_f at intervals of 0.1 and then interpolating), to yield the upper plot in Figure 5.8. Similarly, changing c to 1000, we get the lower plot in Figure 5.8. Notice that as c increased, the boundary decreased. Intuitively, the greater the difference c we are asking for between convergence and loss times, the less overlap and fitness difference we can tolerate. Thus the area above the $c = 10$ plot should represent failed niching, while the area below the $c = 1000$ line represents successful niching (surviving pairs of niches)[5]. The area in between the two bounds is the "gray area" in which we cannot reliably predict success or failure of niching.

Thus Figure 5.8 is the theoretical control map for pairs of niches, covering the range of possible scenarios (fitness versus overlap) and predicting success or failure. Next we perform the experiments to see how well theory corresponds to reality.

5.6.4 Empirical Results

Rather than perform actual runs of a GA with resource sharing among two overlapped niches, we simply use the Markov chain model from Chapter 4, section 2. Since this model is exact, it gives us the expected results for an arbitrarily large number of runs of a real GA. In particular, we use the Markov chain for resource sharing to give us the *expected niching performance* for two niches with $N = 50$, fitness ratios varying from 1 to 4, and overlap varying from none ($r_{ov} = 0$) to complete ($r_{ov} = 1$). We define *expected niching performance* as the probability that both niches are still represented in the population at some arbitrary point in the future. We can use the Markov chain directly to calculate the probability that the population is in a given distribution (i.e., that the chain is in a given state) after any number of generations. We can therefore ask, given a particular initial distribution at generation 0, what is the probability that

[4]From Wolfram Research, Inc., Champaign, IL, USA.

[5]The values of c, namely 10 and 1000, are chosen arbitrarily. This limitation is discussed later in this section.

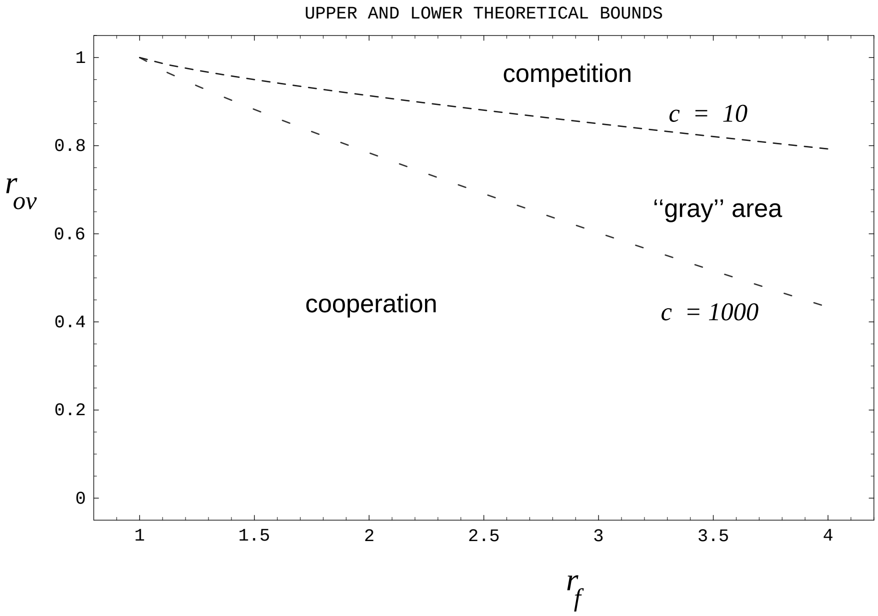

Figure 5.8: Theoretical *cooperative-competitive* boundary for resource sharing given population size $N = 50$, and by arbitrarily choosing $c = 10$ for the niching failure boundary (the lower bound on competition) and $c = 1000$ for the niching success boundary (upper bound on cooperation).

the process has not been absorbed into one of the states corresponding to a uniform population (of all one niche), after some number t generations. (Recall that we are assuming no mutation, so whenever all the representatives of a niche are lost from the population, the niche is lost forever.) This probability, which is one minus the probability of being absorbed by generation t, is the probability of niching success (that is, both niches have survived). This probability is equivalent to the expected number of GA runs, using different random number seeds, in which both niches are present at generation t.

For our "empirical results" we fix population size N to 50 individuals, and choose $t = 200$ generations as a somewhat arbitrary but long period for niches to survive. We choose as an initial population distribution the expected distribution corresponding to a random initialization[6]. Computing the probability of both niches surviving 200 generations requires the multiplication of the initial population distribution vector with the Markov transition matrix (from section 4.2)

[6]In other words, since there are only two niches **A** and **B**, and hence only two possible individuals, the distribution is a binomial one, over the spectrum from "all **B**s" to "all **A**s".

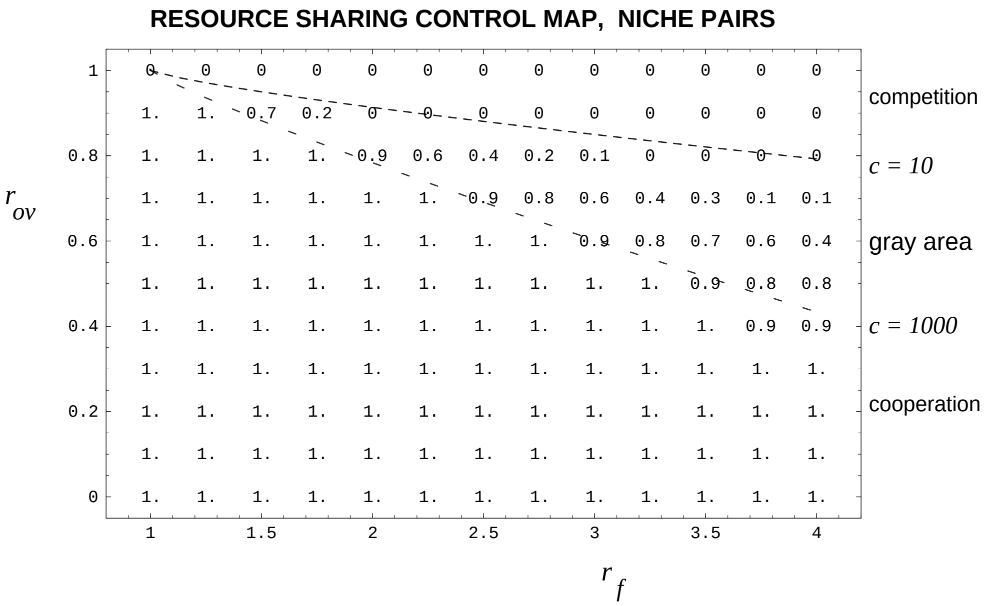

Figure 5.9: Analytical results superimposed on empirical results: the numbers plotted are the expected survival rate for the niche *pairs* (i.e., both niches survive), after $t = 200$ generations. These niching success probabilities are obtained to infinite precision via the Markov chain, but are here shown rounded to the nearest tenth.

200 times (once for each generation), and then adding the probabilities for the two absorbing states (corresponding to all **A**s and all **B**s) and subtracting this sum from one[7].

In Figure 5.9 we show these probabilities, rounded to the nearest tenth[8], for various degrees of niche overlap and fitness ratios. We can call these probabilities *niche performance*, or "niche survival rate". Also shown in Figure 5.9 are the theory bounds from Figure 5.8.

5.6.5 Discussion

The results in Figure 5.9 reveal some important issues and hide some others. These are issues that bear analysis, including the seemingly close agreement between the model's predictions and the actual results, the almost linear nature of the boundaries, and the question: "what is all of this good for?". These and other points are discussed below, in terms of limitations, contributions, and future work.

[7]Or, equivalently, adding up the probabilities for all of the transient states.
[8]I.e., one significant digit.

5.6.5.1 Limitation on Predictive Ability

The analytical boundaries and the actual expected performance data seem to agree quite well, but such agreement must be qualified. We have seen that the analytical bounds are sensitive to the setting of the constant c, as they should be, but we have no theory to guide us in choosing the appropriate c values for the success and failure boundaries. Instead, we chose c values of 10 and 1000 to approximately fit the data "after the fact". Now it is true that the data do bear out our theoretical expectations that the general equation $c\, t_{conv} = t_{abs}$ could be used to find the boundaries of niching success and failure. And it is also true that our prediction that the constant for the competition bound should be much smaller (by several orders of magnitude) than the constant for the cooperation bound (i.e., $c_{fail} \ll c_{succ}$). But it is clear that without some theory for setting c, we cannot fully use our models to *predict* niching success/failure. However, we believe that future work can produce guidance for setting c_{succ} and c_{fail} based on analysis of the relative effects of noise from nondeterministic sources (e.g., selection, crossover, sharing operators).

Future work should also more closely examine the effect of t on the boundaries; that is, how the niche failure and success boundaries change over the generations. Note that our theoretical model does not include the number of generations t as a parameter, but of course our experimental results must. Our models therefore implicitly "predict" that the boundaries do not change (much) over a large range of t. Although Figure 5.9 shows only the results for $t = 200$, a number of other ending generations were looked at, including $t = 50$, 100, 150, 250, and 300. All had similar niching success rates as shown in Figure 5.9, but with slightly decreasing probabilities as t increased (as expected). Thus we are claiming here that the effect of t on niching success/failure is not nearly as significant as the effects of N, r_f, and r_{ov}, but more analysis and experiment is needed to support this conjecture.

5.6.5.2 Contribution: A Quasi-linear Relationship

Figure 5.9 indicates that both analysis and experiment agree on the nearly linear relationship of r_{ov} to r_f along the contours of constant probability of niching success. But Equation 5.34, the source of our theoretical bounds, is highly non-linear. A closer analysis (below) of Equation 5.34

suggests a dominant linear term. First, we repeat Equation 5.34:

$$c\,\frac{-\ln(\frac{r_f - r_{ov}}{r_f - 2r_{ov}+1}N - 1)}{\ln(\frac{r_{ov}}{1-r_{ov}+r_f})} = \frac{(r_f - 2r_{ov} + 1)^N}{(r_f - r_{ov})^N + (1 - r_{ov})^N}.$$

We note that for $r_f \approx 1$, the right hand side essentially reduces to

$$\frac{(r_f - 2r_{ov} + 1)^N}{(r_f - r_{ov})^N + (r_f - r_{ov})^N} \Rightarrow \frac{(r_f - 2r_{ov} + 1)^N}{2(r_f - r_{ov})^N} \Rightarrow \frac{1}{2}(\frac{r_f - 2r_{ov} + 1}{r_f - r_{ov}})^N.$$

Writing the full equation, and rearranging slightly, yields

$$\sqrt[N]{2c\,\frac{-\ln(\frac{r_f - r_{ov}}{r_f - 2r_{ov}+1}N - 1)}{\ln(\frac{r_{ov}}{1-r_{ov}+r_f})}} = \frac{r_f - 2r_{ov} + 1}{r_f - r_{ov}}.$$

Calling the left hand side K, with $K \equiv \sqrt[N]{2c\,t_{conv}}$, we rewrite the above as

$$K = \frac{r_f - 2r_{ov} + 1}{r_f - r_{ov}},$$

and solve for r_{ov}:

$$r_{ov} = \left(\frac{1-K}{2-K}\right) r_f + \frac{1}{2-K},$$

which is linear in r_f if K is constant. Just how much does K vary? From the definition of K we see that for large population sizes N we are dealing with a very small root of $2c\,t_{conv}$. If t_{conv} does not vary much with r_f, K can be considered more or less constant[9].

In future work we might investigate the dominant terms in Equation 5.34 and look for an approximation that yields a compact, closed-form expression relating r_{ov} to r_f linearly, as a function of c and N.

5.6.5.3 Contribution: What We Have Now Versus What We Had Before

A natural question to ask after our comparison of empirical and theoretical niching bounds is: what were our previous theoretical bounds and how do they compare? Prior to the current analysis, the only quantified bounds to be found in the literature are the simple ones that come

[9]Note that t_{conv} is logarithmic in r_f.

from the equilibrium equations directly. For example, we recall the equilibrium condition for resource sharing, Equation 4.12:

$$P_{A,eq} = \frac{n_A}{N} = \frac{1 - r_o}{1 - 2r_o + \frac{1}{r_f}},$$

where n_A is the number of copies of species **A**.

Certainly, we can expect difficulty maintaining both niches if the expected number of copies of **B** at equilibrium is less than one: $n_B < 1$. (Recall that we have all along assumed that **B** is the lesser fit niche: $f_B \leq f_A$.) So we look for the boundary $n_B = 1$. Substituting $n_B = N - n_A = 1 \Rightarrow n_A = N - 1$ into the equation above, and also substituting $r_o = \frac{r_{ov}}{r_f}$ to put the bound in terms of our new overlap ratio, and rearranging, we get:

$$r_{ov} = \frac{N - r_f - 1}{N - 2}. \tag{5.35}$$

This bound is then our "old bound on competition", and is shown as the upper solid line in Figure 5.10. Using only our equilibrium condition, we could at least safely say that for r_f, r_{ov} situations above this line, we can expect niching failure (and hence pure competition), since < 1 copies of the lesser niche are expected to survive at equilibrium.

As for an upper bound on cooperation, below which we can be sure both niches will be maintained, we really did not have a rigorous boundary. Without models of niching performance under overlap, it was often assumed that niching would fail if any overlap existed[10]. While much empirical evidence of niching's robustness existed (e.g., Deb, 1987; Goldberg, Deb, & Horn, 1992), and even some theoretical work indicated graceful degradation of niching performance with increasing niche overlap (e.g., Horn, 1993), no measure of sharing's tolerance of overlap has been proposed. Thus we indicate the "old bound on cooperation" in Figure 5.10 as the flat line just at or above $r_{ov} = 0$.

Between the two solid lines indicating the old bounds on competition and cooperation is the large "gray area" indicating situations of fitness and overlap in which we had no theoretical guidance for predicting niching success or failure. Figure 5.10 also shows our new bounds

[10]Hence some of the criticisms of sharing as "brittle", as well as the frequent assumptions of "perfect discrimination" (e.g., Mahfoud, 1995).

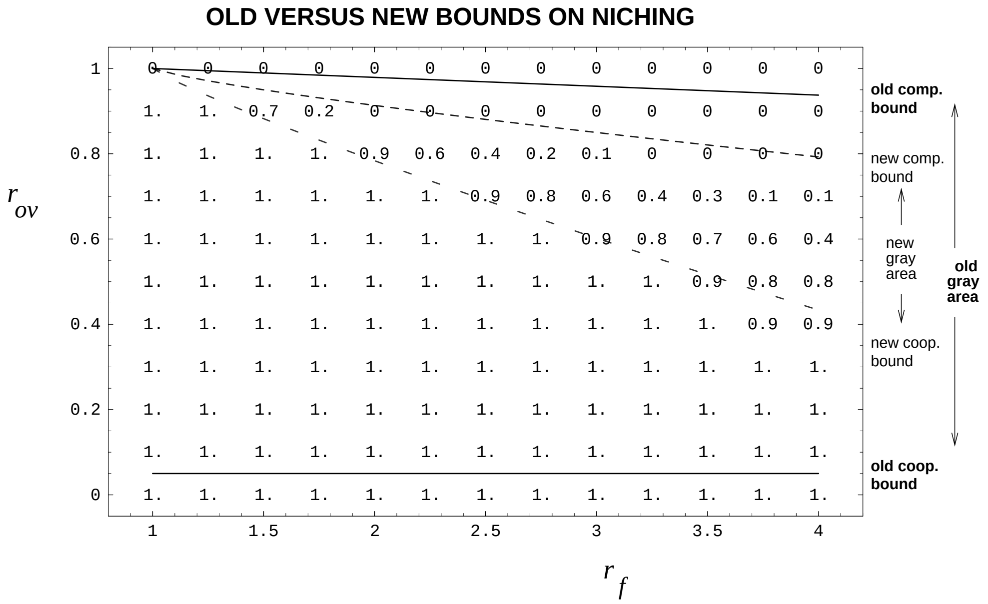

Figure 5.10: A comparison of our previous theoretical predictions (solid lines) of pair-wise niching success and failure versus our new bounds (dashed lines), both superimposed on the actual results. The old bounds come from the equilibrium equations in Chapter 4.

developed in this thesis, for comparison. The newer bounds seem to more tightly bound the *actual* gray area of niching.

5.6.5.4 Limitation: Prediction Without Control

Although we have named Figure 5.9 a control map, it is not as directly useful as the mixing control maps of (Thierens, 1995), for example. Our model is certainly predictive, showing us the regions of cooperation, competition, and the gray area in between. But to have more direct *control* over which niching situations constitute cooperation and which competition, we need to replace the parameters r_f and r_{ov} with more easily manipulated parameters, such as the probability of crossover p_c and the selection pressure s used by Thierens. That is, we want the user of the niched GA to be able to set parameters at the beginning of a GA run, so as to place the boundary between cooperation and competition to ensure the survival of desirable pairs of niches while also ensuring the resolution (convergence) of niche pairs that the user deems competitive.

Such control parameters might not be hard to find. Candidates for fitness sharing control parameters include niche radius σ_{sh}, sharing function exponent α_{sh}, power sharing exponent

β_{sh} (Goldberg, Deb, & Horn, 1992), and population size N. For example, as β_{sh} increases, the importance of objective (unshared) fitness increases, and the range of "tolerable" r_f decreases. Similarly, as α_{sh} and/or σ_{sh} increase, the range of tolerable niche difference (i.e., overlap) decreases. So on the x-axis, r_f might be replaced by β_{sh}, while on the y-axis r_{ov} might be replaced by some function of σ_{sh} and α_{sh}, to yield a more practical control map. This map would allow us to tune our niched GA parameters so that the niche pairs we desire do indeed survive, while the overlapped, conflicting niche pairs we do not want are quickly resolved in favor of the better niche.

For resource sharing, the candidates for control parameters are not as obvious (other than increasing N to support more overlap and fitness differences). It appears promising however that analogs of fitness sharing's parameters can be found. Metaphors from nature as well as review of common successful techniques in classifier systems and immune system models inspire such optimism. For example, it is not necessary to strictly divide up a resource completely. In nature as well as in some recent LCS work, a resource might become "more available" with increasing demand. Or it might be the case that an individual might not be able to use more than one fraction of a particular resource anyway; thus additional users of the resource do not so drastically reduce the original user's share. Such effects would allow the niched GA to support increased r_{ov}. Similarly, if we were to simulate more destructive aspects of competition, such as the severe depletion of a resource with increasing usage, we could increase the importance of r_f and lower the capacity of the system for handling overlap. Identification of direct control parameters for resource sharing might involve algorithmic details particular to specific implementations, such as antibody-antigen binding strengths as a "niche radius" in immune system models, or the bidding algorithms (auctions) in the LCS. This is clearly a subject for future research.

In general, control over the bounds of cooperation and competition is a worthy goal. For example, it is desirable to reduce the "gray area" between the bounds, since this region constitutes the set of niching situations which are unpredictable. It appears that a simple way of reducing the gray area is to increase population size N. Larger N should raise the two bounds plotted in Figure 5.9, but will likely raise the lower plot (that is, the upper bound on cooperation) at a greater rate than the upper plot. Confirming such an effect would be a short-term extension of the current work.

5.6.5.5 Other limitations

Additional limitations of this control map are its restriction to resource sharing and its assumption of a single pair of niches. As for the former limitation, the analysis in this section can and should be duplicated for fitness sharing, in future work. As for the latter limitation, the control map should easily extend to the case of multiple pairs of (possibly overlapping) niches in a single population, with little change. But the extension to three or more tuples of mutually overlapping niches (i.e., triples, quadruples, etc.) is not necessarily straightforward. We discuss the extensions of this and all of our niching models to $k \geq 3$ niches in Chapter 7, section 2.

5.7 Summary

In this chapter we have shown that convergence to niching equilibrium is fast, as we suspected. It is immediate for perfect sharing, and grows slowly, logarithmically, in population size N for overlapped niches. This fast convergence time means that not only is niching equilibrium reached quickly initially, but that perturbations from equilibrium are quickly compensated for. This fact supports our assumption in Chapter 4 that the GA will maintain the population at or very near equilibrium most of the time. We converted our models of niching convergence times to practical, concise closed-form expressions. Contrasting the fast (logarithmic) convergence times with the long (exponential) niche loss times of Chapter 4, we show that niching equilibrium dominates the evolutionary time scale. But we note that increasing fitness difference r_f and increasing overlap r_o or σ_{sh} slow niche convergence time while also speeding up niche loss. We conjecture that when these two characteristic niching times are similar, perhaps differing by an order of magnitude only, niching fails. Applying this conjecture, and solving for the combinations of fitness and overlap that give us these niche failure conditions, we can plot the boundary of niching failure and success. (The boundary turns out to be fairly linear for resource sharing, and indicates that niche loss time dominates the calculation of the cooperation-competition boundary.) This boundary gives us a control map predicting which combinations of niche fitness and niche overlap will lead to successful niching (i.e., cooperation) and which will lead to failed niching (i.e., successful competition). Such a control map is the first step in being able to tune our niched GAs to promote exactly the kinds of cooperation and competition we deem appropriate for the problem at hand.

Chapter 6

The Meaning of Niching Equilibrium

We have spent two chapters defining niching equilibrium, analyzing its stability, and predicting the time spent reaching it and staying at it. We have assumed that equilibrium is important and useful, justifying that assumption by intuition but also with empirical evidence from our own experience and from the literature. Here we ask some harder questions about the nature of equilibrium, trying to characterize what it represents. Does it represent a solution to some problem, with the solution representation distributed over the population? Is the equilibrium distribution *optimal* by any figure of merit? In Chapter 8 we will explore the *utility* of equilibrium by applying niching to several problem types. But here in this chapter we concentrate on the intrinsic meaning, if any, of this strong attractor of GA population dynamics. In particular, we are intrigued by four potential qualities of niching equilibrium:

1. *niching as computation* - equilibrium represents a solution to a complex set of non-linear equations,

2. *niching as a search mechanism* - equilibrium promotes schemata in proportion to their average fitness,

3. *niching as population optimization* - equilibrium is the optimal population distribution according to a simple function of population distribution,

4. *population as solution* - equilibrium can encode/represent a "cooperative solution" to a hard problem.

All of these possible characteristics of niching equilibrium deserve further work but here we present some preliminary analysis of the first three conjectures. Although these early results are both promising and intriguing, we note that the discussions in this chapter are much more speculative in nature than those of Chapters 4 and 5.

6.1 Convergence as Computation?

In this section we examine the complexity of the equilibrium equations (Equation 4.1): $f_{A,sh} = f_{B,sh}$, for all pairs of niches **A** and **B**. If the solution of the system of equilibrium equations is the equilibrium point, then the convergence of the GA to niching equilibrium can be thought of as the computation necessary to solve the equations.

Up to this point we have examined only the two-niche case, in which the equilibrium was easy to calculate. That is, the system of equations to be solved consisted of setting the shared fitnesses of the two niches equal to each other, plus the equation constraining the sum of the subpopulations to total N, the population size. For all three cases (perfect sharing, fitness sharing, and resources sharing) these equations are easy to solve in the two-niche case. Indeed, we have assumed that figuring out the equilibrium is straightforward, while the hard part was predicting the time spent at equilibrium and the convergence rate for getting to equilibrium. But for $k > 2$ niches, even calculating equilibrium can become complicated, at least for resource sharing.

In the next few sections we show that perfect sharing and fitness sharing present no real difficulties to equilibrium calculation. In particular, fitness sharing's equilibrium equations result in a system of linear equations. In the case of resource sharing however, the equilibrium equations become quite complex for $k > 2$ niches, with polynomials of degree $2k-3$ to be solved. Solving systems of highly non-linear equations is difficult in general. If the GA with resource sharing is quickly finding solutions to such equations, it could be argued that we have isolated a natural algorithm performing some truly complex computation: a nice example of *evolutionary computation*. Perhaps this computation is general enough to be useful in solving otherwise intractable systems of equations. Perhaps the niched GA will reveal a simple algorithm for solving a class of nonlinear equation systems. But at this point we are merely speculating.

6.1.1 Calculation of Perfect Sharing Equilibrium In One Equation

Solving the system of equations to calculate niching equilibrium for perfect sharing is simple. As we increase the number of niches k, we simply increase the number of equations but we do NOT increase the complexity of the equations. For example, the system of equations for the two-niche case is simply (Equation 4.1): $f_{A,sh} = f_{B,sh} \Rightarrow$

$$\{\frac{f_A}{n_A} = \frac{f_B}{n_B}, n_A + n_B = N\},$$

and for the three-niche case:

$$\{\frac{f_A}{n_A} = \frac{f_B}{n_B}, \frac{f_A}{n_A} = \frac{f_C}{n_C}, n_A + n_B + n_C = N\}.$$

And in general for higher k:

$$\{\frac{f_A}{n_A} = \frac{f_B}{n_B}, \ldots, \frac{f_A}{n_A} = \frac{f_K}{n_K}, n_A + \ldots + n_K = N\}.$$

To calculate a particular niche count n_j for a particular niche j, we note that for any niche i,

$$\frac{f_i}{n_i} = \frac{f_j}{n_j},$$

at equilibrium. We can rearrange this as:

$$f_i = \frac{f_j}{n_j} n_i.$$

Now suppose we index the k niches from one to k. We can let i vary from one to k as we sum the above equation over all k niches (but j continues to designate the same niche):

$$\sum_{i=1}^{k} f_i = \sum_{i=1}^{k} \frac{f_j}{n_j} n_i.$$

We can bring $\frac{f_j}{n_j}$ out of the sum on the right, and also divide both sides by k:

$$\frac{\sum_{i=1}^{k} f_i}{k} = \frac{f_j}{n_j} \frac{\sum_{i=1}^{k} n_i}{k}.$$

Note that the left hand side is simple the average niche fitness, call it $\bar{f}$, while on the right, $\frac{\sum_{i=1}^{k} n_i}{k}$ is simply the entire population N:

$$\bar{f} = \frac{f_j}{n_j} N,$$

which can be rearranged to yield:

$$n_j = \frac{N f_j}{k\bar{f}}.$$

Thus the complexity of solving for perfect sharing equilibrium is minimal, requiring the solution of a single simple equation involving the sum of k terms (the niche fitnesses f_i) to get $\bar{f}$. Such simplicity is expected given that equilibrium is reached in a single generation under perfect sharing. Thus the GA under selection with niching must be "solving" the equilibrium equations very quickly, and so they must be simple to solve.

6.1.2 Calculation of Fitness Sharing Equilibrium Involves k Linear Equations

The system of equations for fitness sharing equilibrium is a bit more involved than that of perfect sharing, as we add in the terms for niche overlap. Recall the two-niche case: (Equation 4.1): $f_{A,sh} = f_{B,sh} \Rightarrow$

$$\{\frac{f_A}{n_A + n_B Sh_{AB}} = \frac{f_B}{n_B + n_A Sh_{AB}}, n_A + n_B = N\},$$

which for the three-niche case becomes:

$$\begin{aligned}
\{\frac{f_A}{n_A + n_B Sh_{AB} + n_C Sh_{AC}} &= \frac{f_B}{n_B + n_A Sh_{AB} + n_C Sh_{BC}}, \\
\frac{f_B}{n_B + n_A Sh_{AB} + n_C Sh_{BC}} &= \frac{f_C}{n_C + n_A Sh_{AC} + n_B Sh_{BC}}, \\
&n_A + n_B + n_C = N\}.
\end{aligned}$$

And in general for higher k:

$$\begin{aligned}
\{\frac{f_A}{n_A + \ldots + n_K Sh_{AK}} &= \frac{f_B}{n_B + \ldots + n_K Sh_{BK}}, \\
\vdots \qquad & \qquad \vdots \\
\frac{f_A}{n_A + \ldots + n_K Sh_{AK}} &= \frac{f_K}{n_K + \ldots + n_A Sh_{AK}},
\end{aligned}$$

$$n_A + \ldots + n_K = N\}.$$

Thus no matter how many niches and how much overlap, fitness sharing equilibrium conditions will always generate a system of linear equations. (Cross multiplying any of the first $k-1$ equations above, so that the left-hand-side numerator is multiplied by the right-hand-side denominator, etc., results in polynomials on either side of the equals sign all of degree one. In other words, the equations can be rearranged to form linear combinations of the k variables $n_A, n_B, n_C, \ldots, n_K$.) A system of linear equations (k equations in k unknowns, for the k niches) has a single unique solution that can be quickly found in time polynomial in the number of unknowns and equations k. Thus the complexity of the fitness sharing equilibrium equations is linear, while the cost is polynomial, in k.

6.1.3 Calculation of Resource Sharing Equilibrium Means Polynomial Equations

Resource sharing on the other hand presents some very interesting complications. As soon as we generalize to three niches, we are solving cubic polynomials in three variables to find equilibrium. For $k > 3$ niches, the degree of polynomials to be solved increases linearly in k, so that soon we are solving apparently very hard systems of equations to find out where selection and niching will naturally drive the GA quite rapidly! This raises the question of whether selection with niching is performing some useful, general, and intensive computation. We begin with a review of the two-niche resource sharing case.

6.1.3.1 Two-Niche Review

First we review the two-niche case, with overlap, illustrated again in Figure 6.1, top. We recall the equilibrium equation for resource sharing (Equation 4.1): $f_{A,sh} = f_{B,sh} \Rightarrow$

$$\frac{f_A - f_{AB}}{n_A} + \frac{f_{AB}}{n_A + n_B} = \frac{f_B - f_{AB}}{n_B} + \frac{f_{AB}}{n_A + n_B}. \tag{6.1}$$

The term $\frac{f_{AB}}{n_A+n_B}$ cancels, and we are conveniently left with a linear equation in two unknowns. The other equation, $n_A + n_B = N$, is also linear.

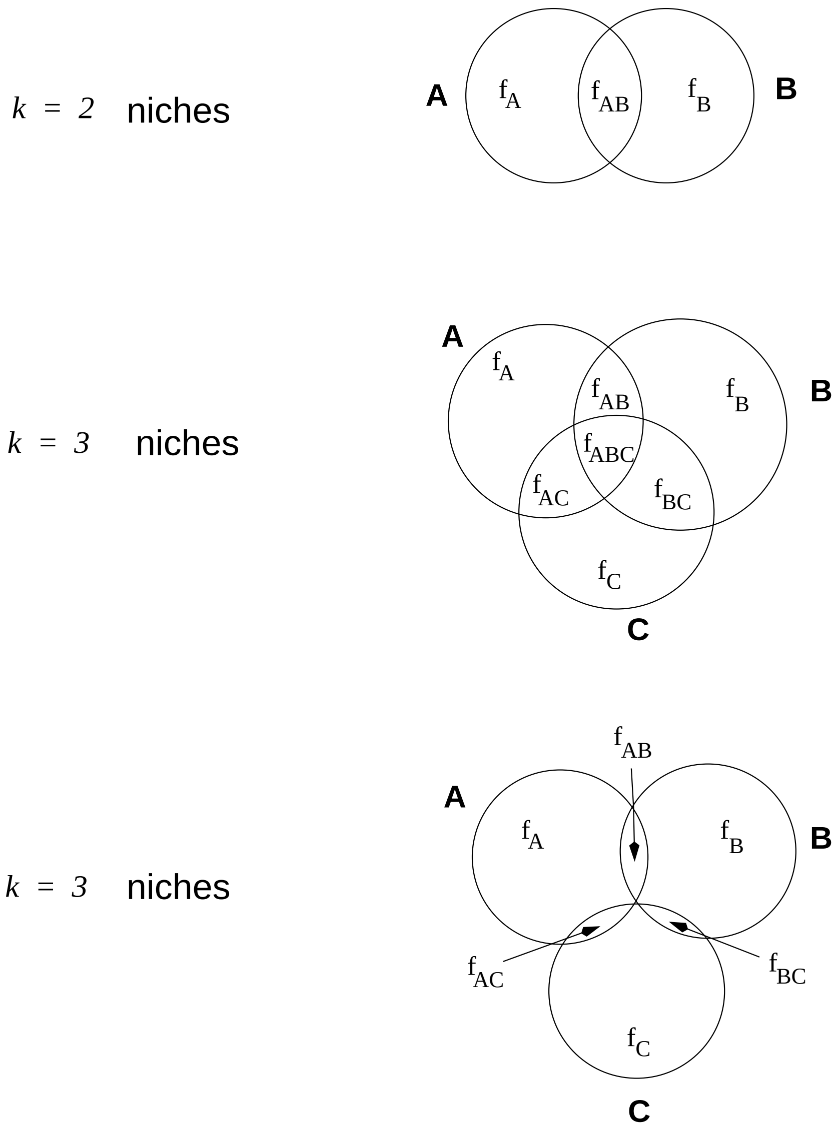

Figure 6.1: Two and three-niche cases of resource sharing, with overlap. For the purpose of solving, and/or measuring the complexity of, the equilibrium equations we can ignore the common overlap area f_{ABC} in the $k = 3$ niches case (middle of figure).

6.1.3.2 Three-Niche Complications

Next we examine the $k = 3$ niche case, shown in the middle of Figure 6.1. Real complications arise if we have overlap between all three pairs, that is $f_{AB}, f_{BC}, f_{AC} > 0$. This is the most general case. Note that we can ignore any overlap common to all three, which we denote as f_{ABC}. For the purpose of calculating the equilibrium, this term will cancel out as common to both sides of any equilibrium condition equation, for example: $f_{A,sh} = f_{B,sh} \Rightarrow$

$$\begin{aligned}
&\frac{f_A - f_{AB} - f_{AC} - f_{ABC}}{n_A} + \frac{f_{AB}}{n_A + n_B} + \frac{f_{AC}}{n_A + n_C} + \frac{f_{ABC}}{n_A + n_B + n_C} \\
= \; &\frac{f_B - f_{AB} - f_{BC} - f_{ABC}}{n_B} + \frac{f_{AB}}{n_A + n_B} + \frac{f_{BC}}{n_B + n_C} + \frac{f_{ABC}}{n_A + n_B + n_C}.
\end{aligned}$$

The term $\frac{f_{ABC}}{n_A+n_B+n_C}$ appears on both sides of the equation and so cancels. We choose to ignore the three-way overlap f_{ABC} for the following calculations. (Alternatively, we can imagine the situation as one in which the only overlaps occur between pairs of niches, as in Figure 6.1, bottom.)

For the three-niche case we have a system of three independent equations in three unknowns: $\{f_{A,sh} = f_{B,sh}\ ,\ f_{A,sh} = f_{C,sh}\ ,\ n_A + n_B + n_C = N\}$. (Note that the fourth equation, $f_{B,sh} = f_{C,sh}$, is redundant.) We can divide through by N to put everything in terms of proportions, $\{f_{A,sh} = f_{B,sh}\ ,\ f_{A,sh} = f_{C,sh}\ ,\ P_A + P_B + P_C = 1\}$:

$$\begin{aligned}
\{&\frac{f_A - f_{AB} - f_{AC}}{P_A} + \frac{f_{AB}}{P_A + P_B} + \frac{f_{AC}}{P_A + P_C} = \frac{f_B - f_{AB} - f_{BC}}{P_B} + \frac{f_{AB}}{P_A + P_B} + \frac{f_{BC}}{P_B + P_C}, \\
&\frac{f_A - f_{AB} - f_{AC}}{P_A} + \frac{f_{AB}}{P_A + P_B} + \frac{f_{AC}}{P_A + P_C} = \frac{f_C - f_{AC} - f_{BC}}{P_C} + \frac{f_{AC}}{P_A + P_C} + \frac{f_{BC}}{P_B + P_C}, \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad P_A + P_B + P_C = 1\}.
\end{aligned}$$

Again, common terms cancel in the first two equations (terms $\frac{f_{AB}}{P_A+P_B}$ and $\frac{f_{AC}}{P_A+P_C}$ respectively), yet the equations are still non-linear. Combining fractions and cross multiplying, we see that the equations are cubic in the proportions (i.e., P_A, P_B, P_C), which are the variables for which we are solving. For example, the first equation above can be rearranged as:

$$\begin{aligned}
&f_A P_B (P_A + P_C)(P_B + P_C) &+\ & f_{AC} P_A P_B (P_B + P_C) \\
& &-\ & f_{AB} P_B (P_A + P_C)(P_B + P_C) \ - \ f_{AC} P_B (P_A + P_C)(P_B + P_C) \\
= \; &f_B P_A (P_A + P_C)(P_B + P_C) &+\ & f_{BC} P_B P_A (P_A + P_C) \\
& &-\ & f_{AB} P_A (P_A + P_C)(P_B + P_C) \ - \ f_{AC} P_A (P_A + P_C)(P_B + P_C)
\end{aligned}$$

which is clearly a third-degree polynomial.

Solving a system of third-degree equations is much more complicated algebraicly than is solving a system of linear equations. In fact, the symbolic solution for P_A for example would take up an entire page. Rather than show the symbolic solutions (in terms of $f_A, f_B, f_C, f_{AB}, f_{BC}, f_{AC}$), we show the approximate solutions for a particular instance of niching, with $f_A = 5, f_B = 3, f_C = 2, f_{AB} = 2.1, f_{BC} = 0.7,\ f_{AC} = 0.1$, and no three way overlap ($f_{ABC} = 0$). The program Mathematica (Wolfram, 1990) finds four solutions:

$$
\begin{aligned}
(1)\quad \{P_A &= -0.6551664478\ldots, \\
P_B &= 0.9795951985\ldots, \\
P_C &= 0.6755712493\ldots\} \\[1em]
(2)\quad \{P_A &= 0.8336143802\ldots, \\
P_B &= -0.2601220806\ldots, \\
P_C &= 0.4265077003\ldots\} \\[1em]
(3)\quad \{P_A &= 0.8783159065\ldots, \\
P_B &= 0.9639271269\ldots, \\
P_C &= -0.8422430335\ldots\} \\[1em]
(4)\quad \{P_A &= 0.6652633707\ldots, \\
P_B &= 0.08978565417\ldots, \\
P_C &= 0.2449509751\ldots\}
\end{aligned}
\tag{6.2}
$$

The first three solutions all have negative components, so clearly the last solution is the correct unique solution.

The solution (4) above took a great deal more calculation than would a two-niche situation or even a three-niche situation with only two overlaps (i.e., with one of f_{AB}, f_{BC}, f_{AC} set to zero). So the questions arise: does the GA actually *find* this equilibrium (i.e., converge to it), and if so, how easily?

To answer these questions, we trace the expected proportions from generation to generation of a single possible run of a GA with the three-niche situation given above (that is, $f_A = 5, f_B =$

$3, f_C = 2, f_{AB} = 2.1, f_{BC} = 0.7$, $f_{AC} = 0.1$, and $f_{ABC} = 0$). This trace was obtained using the recurrence relations from the analysis of resource sharing convergence in section 1.4 (e.g., Equation 5.1) generalized to the three-niche case. For example, the expected proportions for niche **A** go as follows:

$$E[P_{A,t+1}] = \frac{P_{A,t} f'_{A,t}}{P_{A,t} f'_{A,t} + P_{B,t} f'_{B,t} + P_{C,t} f'_{C,t}},$$

for proportionate selection (e.g., RWS), where $P_{A,t}$ is the proportion of **A**s at time t, $E[P_{A,t+1}]$ is the expected proportion of **A**s at time $t+1$, and the denominator of the left hand side above is the average fitness of the population at time t. The shared fitnesses, denoted f' above, are defined as before, so that for $f'_{A,t}$:

$$f'_{A,t} = \frac{f_A - f_{AB} - f_{AC}}{P_{A,t}} + \frac{f_{AB}}{P_{A,t} + P_{B,t}} + \frac{f_{AC}}{P_{A,t} + P_{C,t}},$$

and similarly for $f'_{B,t}$ and $f'_{C,t}$.

In Figure 6.2 we plot a trace of these expected proportions. We begin the trace with an initial population distribution at time $t = 1$ of

$$P_{A,1} = 0.66526337, \qquad P_{B,1} = 0.3, \qquad P_{C,1} = 0.03473663,$$

so that **A** is started at its expected equilibrium proportion, while **B** and **C** are started away from their equilibrium levels calculated in solution (4) in Equation 6.2 above. As Figure 6.2 shows, the population distribution approaches the calculated equilibrium as expected, but takes several generations to do so. After many generations the distribution becomes arbitrarily close to the predicted equilibrium proportions. For example, at generation $t = 10,000$:

$$P_{A,10000} = 0.6652633707\ldots, \quad P_{B,10000} = 0.089785654\ldots, \quad P_{C,10000} = 0.244950975\ldots,$$

which agrees with solution (4) above in all of the significant digits shown.

Of particular interest is the fact that $P_{A,t}$ initially drops from its initial equilibrium level after the first generation, and then takes several generations to recover. Recall that this is expected behavior, and thus the drop is not due to the stochastic nature of selection, or niche

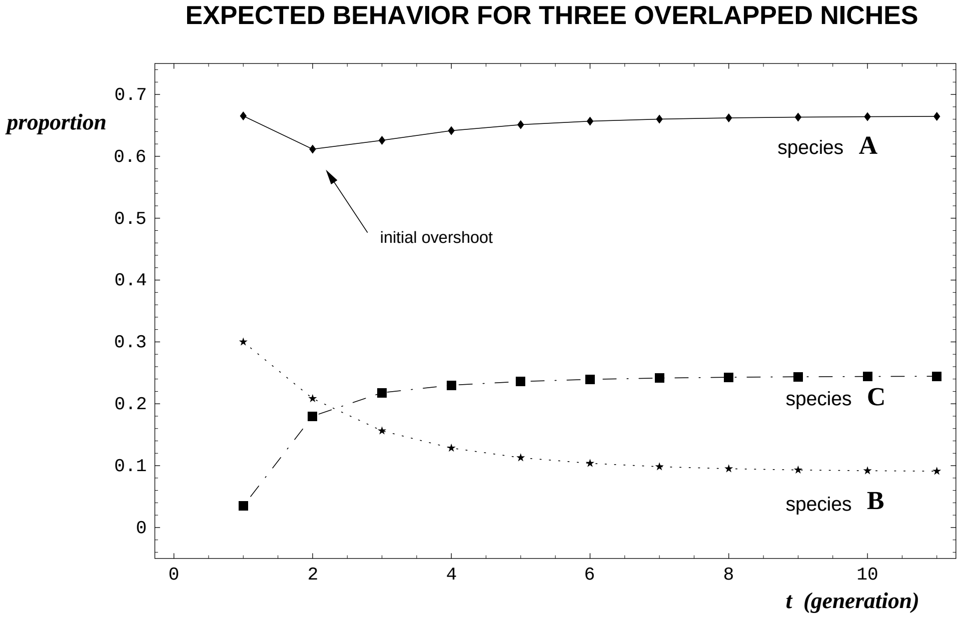

Figure 6.2: A trace of the expected proportions for a niched GA run with three overlapping niches shows a more complicated path to equilibrium than what we've seen for overlapped pairs of niches. Note how **A** is at first driven *away* from its equilibrium level by selection.

count sampling, for example. Rather, this dip points out the difficulty GA selection has in finding the equilibrium.

More complicated "errors" in GA niching convergence can occur: Figure 6.3 shows what is expected to happen under a slightly different situation than the one above. Here we reduce f_{AC} from 0.1 to 0.001, leaving the other objective fitnesses and overlaps the same as above. We also start at a different initial population distribution $\{P_{A,1}, P_{B,1}, P_{C,1}\} = \{0.9, 0.0999, 0.0001\}$. As Figure 6.3 shows, both **A** and **C** overshoot their equilibrium levels at time $t = 2$, errors from which they gradually recover.

Recall that under perfect sharing, convergence to equilibrium occurs immediately, in a single generation. Under overlap, selection can take some time to reach equilibrium. And in the three-niche, mutually overlapping situation, we can actually have movement away from equilibrium levels and overshooting of the equilibrium levels.

Based on our empirical data presented so far, we conjecture that the dips and overshoots found here for three mutually overlapping niches do *not* occur in "simpler" niching situations. More specifically, we know that without overlap convergence is immediate (one generation).

We conjecture that for two overlapping niches, and for three overlapping niches that are *not* mutually overlapped (i.e., with at least one of f_{AB}, f_{BC}, f_{AC} equal to zero), niching convergence is *monotonic*, with no dips or overshoots. That is, without "circular overlaps" initial species proportions should gradually but monotonically increase or decrease to their equilibrium proportions (depending on whether their initial levels were too low or too high, respectively). Thus we are suggesting that *non-monotonic* convergence characterizes a more complicated niching situation, such as the "circular overlap" of three-way mutually overlapping niches. This conjecture is a subject for future work.

The apparent difficulty the niched GA experiences with three-way mutually overlapping niches seems justifiable. Consider the very complicated and contradictory relationships of cooperation and competition occurring simultaneously. Referring back to our discussion of layers of competition and cooperation from Chapter 3, we defined "level 2" cooperation (Figure 3.2) as a pair of niches/species "ganging up" on a third by both overlapping the third. In this sense, we can see that species **A** and **B** can be seen as cooperating against **C** in Figure 6.1 (middle and bottom), and yet we can also view the pair **B** and **C** as cooperating against **A**, and then of course **A** and **C** cooperate against **B**. If we had only two overlaps, (e.g., say that $f_{AC} = 0$), we would have clear and consistent relationships of cooperation and competition.

6.1.3.3 General k-niche Case

In the general k-niche case we assume that all pairs of niches contain some non-zero overlap: $f_{IJ} > 0$, $\forall$ niches I and J. Therefore the shared fitnesses for each niche/species will contain k terms: $f_{I,sh} = \frac{f_I}{n_I} + \frac{f_{IA}}{n_I+n_A} + \frac{f_{IB}}{n_I+n_B} + \ldots + \frac{f_{IK}}{n_I+n_K}$. When set equal to the shared fitness of some other niche, say J (thus $f_{I,sh} = f_{J,sh}$), then one term will cancel, namely $\frac{f_{IJ}}{n_I+n_J}$ and we will be left with $k-1$ terms on each side. Combining these fractions will mean multiplying together the $k-1$ denominators to create a common denominator of degree $k-1$:

$$n_I(n_I + n_A)(n_I + n_B)\ldots(n_I + n_K),$$

while the numerator will be a polynomial of degree $k-2$:

$$f_I(n_I + n_A)(n_I + n_B)\ldots(n_I + n_K) + f_{IA}n_I(n_I + n_B)(n_I + n_C)\ldots + \ldots.$$

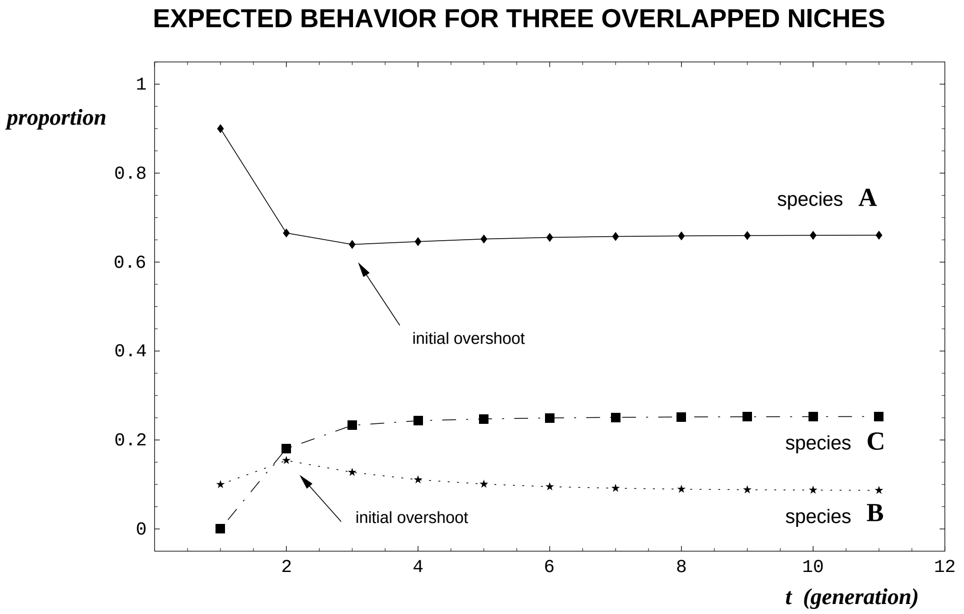

Figure 6.3: In another three-way mutually overlapped niching situation, we see two of the three species overshooting their eventual equilibrium levels.

With two fractions on either side of the equation, we cross multiply to produce a polynomial on each side of the equation equal to the left-hand-side denominator times the right-hand-side numerator, and vice-versa. Multiplying a $k-1$ degree polynomial by a $k-2$ degree polynomial produces a $(k-1)+(k-2) = (2k-3)$ degree polynomial. This result agrees with our $k=2$ and $k=3$ niche cases above, in which we obtained linear and cubic polynomials respectively.

Thus the degree of the equilibrium equations increases rapidly (linearly) with the number of niches k. Furthermore we note that this is a lower bound, because we ignored overlaps among more than two niches (i.e., overlaps of "order" three or more, where "order" could be defined as the number of niches sharing the overlap). In the three-niche case we analyzed above, there were no niche overlaps among more than two niches at a time, except for the order-three overlap f_{ABC}. But the order-k overlap in any k-niche case always cancels from every equilibrium equation, as it is a common term in the shared fitness of every one of the k niches. Overlaps of order $k-1$ and below, however, do not cancel from every equilibrium equation, and so can add to the total degree of the polynomials resulting from such equations. Thus $2k-3$ is only a lower bound on the degree of the equilibrium equations. So we can say that the calculation of resource sharing equilibrium involves a set of highly nonlinear equations, each of which is polynomial in the number of niches k, in general. The cost of solving the equations is at the very least polynomial in k, and may well be more expensive depending on the complexity of the algorithm that is used to solve them.

Finding solutions to k^{th}-degree polynomials in one variable is in general computationally expensive for $k > 3$ and difficult for $k > 5$. For a *system* of k k^{th}-degree polynomials in k unknowns, it is often not possible to find a solution. The fact that niching finds the solution relatively quickly (at least we assume it will for the k-niche case) raises the intriguing possibility that the simple process of GA resource sharing could be used to solve difficult systems of equations of a certain form. Perhaps the fitnesses and overlaps (e.g., f_A, f_{AB}) could be set according to the coefficients of a given set of polynomial equations to be solved. Or it could be the case that the type of non-linear equations we have examined contain enough "structure" that they are fairly easily and quickly solved by analytical means, perhaps using an iterative solution method like the GA's generational approach to equilibrium. Abstracting such an iterative algorithm directly from the niched GA might then make GA niching a relatively poor solution method (since it is probabilistic and hence not deterministic). Still, the possibility of an efficacious method to solve a general class of difficult systems of non-linear equations is deserving of further study as a new form of *evolutionary computation*.

6.2 Niching and GA Search

Both intuition and experience tell us that fitness sharing is a boon to GA search. In this section we review those intuitions and practical experience, and add some preliminary analysis to explain and predict sharing's effects on search. In particular we examine how schemata, the basic building blocks for GA search, are maintained at niching equilibrium (at least for perfect sharing).

6.2.1 Background and Motivation

In their application of a *niched Pareto GA* to three problems, Horn and Nafpliotis (1993)[1] found evidence that sharing helped the GA find more solutions than the GA could without sharing.

[1]This work is published in summarized form as (Horn, Nafpliotis, & Goldberg, 1994) and is reproduced (in part) in Chapter 8 of this dissertation.

Their algorithm uses sharing within the equivalence class defined by the Pareto optimality criterion, i.e., the Pareto optimal set. *Pareto domination tournaments* are used to drive the population toward the *Pareto optimal frontier*, while sharing among the members of the current *non-dominated* (or Pareto optimal) set spreads this subpopulation out over the nondominated tradeoff surface. Without sharing, the population converges to only a small portion of the entire Pareto surface, without ever having generated Pareto optimal solutions from other regions. Thus the simple addition of sharing causes generates many alternative solutions, which are all equivalent (i.e., globally optimal) by the (Pareto) selection criterion[2]. The fact that other GA researchers applying combinations of sharing and Pareto selection to other problems have also reported success at locating more Pareto optimal solutions with sharing than without (Cieniawski, 1993; Fonseca & Fleming, 1993; Srinivas & Deb, 1995), adds further evidence that sharing benefits search, at least for Pareto selection.

Intuitively, fitness sharing *should* help GA search because it attacks the problem of *premature convergence*. A single solution can only take over a segment of the population in proportion to its fitness relative to the rest of the population. By maintaining high quality diversity, sharing should allow more time for crossover to recombine building blocks correctly. More specifically, Thierens and Goldberg (1993) and then Thierens (1995) have shown that *mixing times* can be much longer than *convergence times* (Goldberg & Segrest, 1987; Goldberg & Deb, 1991) under selection (or genetic drift) alone. This timing mismatch often leads to premature convergence. But Horn (1993), Horn, Goldberg, and Deb (1994), Mahfoud (1994a, 1994b), Thierens (1995), and this dissertation show that sharing can maintain diverse niches, at equilibrium, almost indefinitely. Furthermore, we have shown here that equilibrium is reached quickly, sometimes in one generation[3].

Thierens (1995) suggests that any robust niching method, in particular sharing, should help avoid the timing mismatch between selective convergence and mixing, although he states that niching will not significantly shorten the expected mixing times, which can grow exponentially in the number or size of building blocks[4]. It is not immediately clear however, just how sharing

[2]Note that all other GA parameters, such as p_c and N, are held constant; only sharing is adjusted.

[3]Under "perfect sharing" conditions and proportionate selection.

[4]Thierens (1995) does not provide any analytical support of his conjectures on niching, but his experiments show improved search with niching (his *elitist recombination*) over a GA without niching, on certain test problems.

maintains highly fit *schemata* at equilibrium. The analysis below addresses schema maintenance under niche maintenance.

6.2.2 Analysis: Equilibrium and the Schema Theorem

The following results apply to any subset of the population, including subsets defined by schemata. Let $\mathcal{S}$ and $\mathcal{S}'$ be two distinct (possibly overlapping) subsets of the current population $\mathcal{P}$. Let i be an index over all of the niches (i.e., distinct individuals in $\mathcal{P}$) containing members of $\mathcal{S}$, and let j be an index over all of the niches containing members of $\mathcal{S}'$. Let m_i denote the niche count of the i^{th} niche of $\mathcal{S}$ and m'_j be the j^{th} niche count for $\mathcal{S}'$. (We are not concerned here with the numbers of such niches, which might be different for $\mathcal{S}$ and $\mathcal{S}'$.) We further define f_i to be the fitness of niche i in $\mathcal{S}$, and f_j to be the objective (i.e., unshared) fitness of niche j in $\mathcal{S}'$. We will assume for now perfect sharing (no overlap) so that m_i equals the *number* of individuals in niche i of $\mathcal{S}$, and m'_j equals the number of individuals in niche j of $\mathcal{S}'$. We further define n and n' to be the number of members of $\mathcal{S}$ and $\mathcal{S}'$ respectively, in the current population. We note that $n = \sum_{\forall i} m_i$ and $n' = \sum_{\forall j} m'_j$.

At equilibrium, $\frac{f_i}{m_i} = \frac{f'_j}{m'_j} \Rightarrow m_i f'_j = m'_j f_i$, $\forall i, j$. Therefore we can write

$$\sum_{\forall\, i,j} m_i f'_j = \sum_{\forall\, i,j} m'_j f_i. \tag{6.3}$$

Factoring, we derive

$$\sum_{\forall i} m_i \sum_{\forall j} f'_j = \sum_{\forall j} m'_j \sum_{\forall i} f_i. \tag{6.4}$$

Rearranging, we find

$$\frac{\sum_{\forall i} m_i}{\sum_{\forall j} m'_j} = \frac{\sum_{\forall i} f_i}{\sum_{\forall j} f'_j}. \tag{6.5}$$

The ratio on the left of Equation 6.5 is the ratio of the size of $\mathcal{S}$ to the size of $\mathcal{S}'$ (i.e., $\frac{\sum_{\forall i} m_i}{\sum_{\forall j} m'_j} = \frac{n}{n'}$), so that

$$\frac{n}{n'} = \frac{\sum_{\forall i} f_i}{\sum_{\forall j} f'_j} = \frac{f}{f'}, \tag{6.6}$$

where f and f' are the summed fitnesses of the *niches*, not the individuals, of $\mathcal{S}$ and $\mathcal{S}'$ respectively. Thus the relative representation (i.e., numbers of individuals) of the two sets at steady state is directly proportional to the ratio of their summed fitnesses.

If $\mathcal{S}$ and $\mathcal{S}'$ represent schemata hyperplanes (i.e., the members of $\mathcal{S}$ in population $\mathcal{P}$ all have in common the schema associated with $\mathcal{S}$), then Equation 6.6 states that the two (possibly competing) schemata will be given relative representation according to their total (summed) niche fitness. In particular, if we let $\mathcal{S}'$ represent the schema $* * * \ldots *$ (i.e., the entire search space), then Equation 6.6 says that the sharing equilibrium should give each schema $\mathcal{S}$ a number of representatives proportional to its niche fitness with respect to all niches in the population. That is, schemata with above average niche fitness will receive greater than average numbers of representatives in the population. Like the schema theorem for a simple GA (Holland, 1975; Goldberg, 1989a), the above "niched schema theorem" states that above average fitness schemata are favored (in terms of population representation). Unlike the schema theorem, the niched schema theorem describes a dynamically stable situation (i.e., sharing equilibrium). The schema theorem, on the other hand, is a difference inequality, describing the rate of growth of schemata representations during the relatively fast period of selection-driven convergence to a (near) uniform population.

If we add in the effect of low-point crossover (e.g., one or two point crossover), we should find that the sharing equilibrium favors lower order, higher fit schemata over higher order, lower fit schemata, as predicted by the schema theorem for the simple GA. And just as some researchers have successfully used *static schema average fitnesses*[5] to construct easy and difficult problems for a simple GA, we should be able to use static estimates of summed niche fitness for schemata to predict the performance of GAs with fitness sharing.

6.2.3 Empirical Confirmation

The fitness sharing "theory" developed above is not sufficient to predict specific GA performance, and hence cannot yet be adequately tested. However, it does suggest that a GA with sharing should perform at least as well as a simple GA in finding a global optimum, since both favor short, fit schemata (i.e., building blocks). And the comparison of pure selection's $O(c \log N)$ convergence times to sharing's $O(c^N)$ niche maintenance times, in the context of crossover's minimal mixing times (Thierens & Goldberg, 1993), points to an expected improvement in search.

[5]That is, fitnesses averaged over a large, uniformly random population, or over the entire search space.

This performance improvement should come with the introduction of sharing even without increasing population size. This contrasts with Goldberg's original conservative population sizing guidelines for niched search (Goldberg, 1989b). Goldberg suggests using a population size of $N = kn$, where k is the number of niches expected/desired, and n is the population size sufficient for a simple GA (without niching) to find one of the k optima[6]. This early and conservative guideline is assumed by (Beasley, Bull, & Martin, 1993) in their cost comparison of sequential niching and fitness sharing, leading to some results favoring sequential niching over the "parallel niching method" (Mahfoud, 1995a,b) of fitness sharing. Mahfoud (1995a,b) however, suggests that population sizing for class (niche) maintenance dominates population sizing for "class formation" (a term he uses to mean niche search). That is, Mahfoud conjectures that on many problems, a population size sufficient for the maintenance of the desired niches, will be more than large enough for the search required to find the niches. He supports his conjecture empirically, with results from several test problems and using several niching methods, including sharing.

Below we present some preliminary experiments of our own, indicating that the addition of fitness sharing alone *can* improve GA search.

6.2.3.1 F1: The massively multimodal problem

Returning to Goldberg, Deb, and Horn's (1992) work on niching for massively multimodal problems, we find several runs indicating superior search via fitness sharing. For example, Table 6.1 summarizes two nearly identical runs, one with a simple GA and the other with sharing. In both runs, the probability of crossover $p_c = 0.9$, there is no mutation ($p_m = 0$), and power sharing is used, with a fitness exponent of $\beta = 15$. Thus the sharing run is exactly the same as the successful run described in (Goldberg, Deb, & Horn, 1992) and summarized in the next chapter, with the exception of the smaller population size $N = 300$. The simple GA run is the same as the sharing run, except that sharing is turned off and an additional 200 randomly generated individuals are added to the population[7], for a total of $N = 500$.

As Table 6.1 shows, the sharing run finds all 32 global optima, and even maintains most of them, at least as long as the hundredth generation. The simple GA, on the other hand,

[6] Where n might be estimated theoretically (e.g., Goldberg, Deb, & Clark, 1992) or empirically.

[7] Since the same random seed is used for both runs, the first 300 individuals in both populations are the same.

Massively Multimodal F1		
	simple GA	w/Sharing
Pop size N	500	300
Exp. of fitness	15	15
σ_{sh}	(na)	5
max. # of diff. globals found (of 32)	20	32
generation in which last global found	24	43
# number of globals in generation 100	1	30

Table 6.1: Sharing finds more global optima on F1.

only finds 20 globals over the course of its run. As expected, by the hundredth generation, the simple GA has converged completely to a single solution, and with no mutation, there is no chance of any further changes. Note that even with the smaller population size, sharing takes longer than the simple GA to finish its search (that is, to find the last global it will find), doing so in the 43^{rd} generation as opposed to the 24^{th} generation for the simple GA. The additional time taken for the extra searching is evidence for the idea that premature convergence is what precludes the simple GA from finding more globals. With a population size of 500, we expect the simple GA's initial random population to have $500 * 2/2^6 = 15.625$ copies of each building block necessary to build all 32 global optima. Even with a population size two fifths smaller, sharing is able to maintain more building blocks longer; long enough to put together all 32 global optima.

Thus if we were really hoping to find a *particular* global optimum of the 32 possible, we would be more likely to find it via sharing. However, if we were only interested in finding *any* global optimum (i.e., we only cared about maximizing fitness), then the results on F1 do not show any distinction between the two algorithms. Next, we turn to an open problem in civil engineering for evidence of such a distinction.

6.2.3.2 F2: the New York City tunnels problem

Function F2 is an open problem in water distribution pipeline design, known as the *New York City tunnels problem* (or simply "the NY problem"). We demonstrate how sharing can improve the GA's probability of locating the best optima, but we leave the discussion of this problem and the results to Chapter 8: Applications of Niching (and the utility of sharing equilibrium).

6.3 Optimization (of a Population)

Since GA search is key to GA function optimization, the above results indicating that niching helps GA search are good news for GAs as function optimizers (GAFOs) as well. But in addition, sharing can be seen as a "population optimizer" in the same way that a simple GA can be viewed as function optimizer. We show that sharing can optimize by describing a simple scalar function of the population distribution, whose maximum value occurs at the niching equilibrium distribution. Thus niching equilibrium is in some sense an optimal population.

6.3.1 Background

The idea of evolution, whether artificial or natural, as an optimizer of organisms is controversial enough (De Jong, 1992; 1993), but the proposition that evolution (at least selection) somehow optimizes a *population* of organisms might be considered radical. Yet if we could model evolution as a population optimizer, we might be able to predict population space trajectories, and basins of attractions for locally or globally optimal populations. Here we explore the possibility that at least for niching we can indeed model selection as an optimizer of some scalar function of the current population.

6.3.2 Population Optimization in the Simple GA

First, let us consider the base case of the simple GA, in which there are no interactions among individuals, other than pure competition. Thus, the fitness function is "context-free", depending only on the individual being evaluated. A simple GA can be viewed as a population optimizer where the population function is simply the sum or average fitness:

$$F(P) = \sum_{i \in \mathcal{P}} f_i. \tag{6.7}$$

This function is optimized when $\mathcal{P}$ consists of N copies of the best individual. Selective pressure in a simple GA drives the population toward such a uniform, maximal population.

6.3.3 Population Optimization in a Niched GA

Selective pressure in a shared GA however, cannot be modeled as an optimizer of $F(\mathcal{P})$ above. Let k be the number of distinct classes (niches) in a population $\mathcal{P}$ of size N. Let m_i be the number of individuals (copies) in class i, $1 \leq i \leq k$, so that $N = \sum_{i=1}^{k} m_i$. Furthermore, let f_i be the objective fitness of class i. Then the shared fitness f_i' of each individual in class i is $f_i' = \frac{f_i}{m_i}$, under the assumption of perfect sharing (i.e., no overlap). Then the sum of the population's fitness is the sum of the shared fitnesses:

$$F(\mathcal{P}) = \sum_{i=1}^{k} m_i f_i' = m_A f_A' + m_B f_B' + \ldots, \tag{6.8}$$

which is simply

$$F(\mathcal{P}) = \sum_{i=1}^{k} m_i f_i' = \sum_{i=1}^{k} m_i \frac{f_i}{m_i} = \sum_{i=1}^{k} f_i. \tag{6.9}$$

Thus the sum of population fitnesses is independent of the distribution of individuals among the k classes/niches, as long as each niche has at least one individual. So the function $F(\mathcal{P})$ does not serve us very well as a model of *niching pressure*, since it does not predict the shared GA's tendency to reach and maintain sharing equilibrium.

Interestingly, the *product* of population fitnesses does appear to serve well as a function for population optimization. We call the product of fitnesses F' and define it:

$$F'(\mathcal{P}) = \prod_{i \in \mathcal{P}} f_i' = \prod_{i \in \mathcal{P}} \frac{f_i}{m_i}. \tag{6.10}$$

We assume a population with k distinct niches present, with no overlap among the niches (i.e., perfect sharing). So there are m_i copies of solution i for niche i, with $1 \leq i \leq k$. Then we can write f' as:

$$F'(\mathcal{P}) = \prod_{i=1}^{k} (\frac{f_i}{m_i})^{m_i} = (\frac{f_A}{m_A})^{m_A} (\frac{f_B}{m_B})^{m_B} \ldots. \tag{6.11}$$

To show that proportionate fitness sharing (i.e., selection plus niching) optimizes F', we demonstrate that F' is optimal when the niched GA is at equilibrium. That is, when $f_i/m_i = f_j/m_j \ \forall i, j \in \mathcal{P}$, then $F'(\mathcal{P}) = F'_{max}(\mathcal{P})$. Again, we examine only the selection and niching operators, so there is no introduction of new material into the population. We assume a population that is *not* at equilibrium. Then there must exist two niches A and B with fitnesses f_A

and f_B and niche counts m_A and m_B respectively, such that $\frac{f_A}{m_A} > \frac{f_B}{m_B}$. We assumed perfect sharing so that niche counts m_i essentially equal the number of copies of solution i in the current population. We want to show that replacing one member of niche B (i.e., one copy of solution B) with one member of niche A increases the total population fitness[8] F'.

More specifically, we need to show that replacing m_A by m_A+1, and m_B by m_B-1 in the product F' would actually increase F', or in other words

$$(\frac{f_A}{m_A+1})^{m_A+1}(\frac{f_B}{m_B-1})^{m_B-1} > (\frac{f_A}{m_A})^{m_A}(\frac{f_B}{m_B})^{m_B}. \tag{6.12}$$

Dividing both sides of Equation 6.12 by $f_A^{m_A} f_B^{m_B}$ and then multiplying both sides by $(m_A+1)^{m_A+1}(m_B-1)^{m_B-1}$, we obtain

$$\frac{f_A}{f_B} > \frac{(m_A+1)^{m_A+1}(m_B-1)^{m_B-1}}{m_A^{m_A}\, m_B^{m_B}}, \tag{6.13}$$

which can be simplified and factored as

$$\frac{f_A}{f_B} > \frac{m_A}{m_B}\left[\left(1+\frac{1}{m_A}\right)^{m_A}\left(1-\frac{1}{m_B}\right)^{m_B}\right]. \tag{6.14}$$

From our assumption that $\frac{f_A}{m_A} > \frac{f_B}{m_B}$, we know that $\frac{f_A}{f_B} > \frac{m_A}{m_B}$. If the product within the square brackets above can be shown to be never greater than one, then our proof is finished. We note that niche counts m_i must always be ≥ 1, since there is at least one representative of niche i in the population, or the niche would not be of interest. For $m_A \geq 1$, $(1+\frac{1}{m_A})^{m_A}$ is monotonically increasing with m_A, with a limit of e. The other multiplicand, $(1-\frac{1}{m_B})^{m_B}$, also monotonically increases in $m_B \geq 1$, but with a limit of $\frac{1}{e}$. Therefore, the limit of the product of the two terms is $e * \frac{1}{e} = 1$ as $m_A, m_B \to \infty$. And our proof is finished.

We have just shown for any pair of niches that are not at "pair-wise equilibrium" (that is, one has a shared fitness greater than that of the other), increasing the niche count of the niche with the higher shared fitness will increase F'. Since this applies to any and all possible pairs among the k niches, we have shown that F' increases monotonically as the population distribution moves closer to equilibrium. Therefore, at equilibrium, F' is maximal.

[8]Such a replacement also moves the population closer to sharing equilibrium.

6.3.4 Discussion

We have shown that if the population is not at sharing equilibrium (i.e., $\exists \text{niches} A, B : f_A/m_A > f_B/m_B$), then the population fitness F' can be improved by shifting one individual from niche B to niche A, and hence bringing the population closer to equilibrium (i.e., $\forall \text{niches} i, j : f_i/m_i = f_j/m_j$). Hence we have proven that (perfect) sharing equilibrium occurs at the optimum of F', the product of fitnesses.

To the same extent that we could claim that selection optimizes *summed* population fitness, we can claim that niching (sharing) optimizes the *product* of population fitnesses. To what extent this model can be used to help explain or predict niched GA performance remains to be seen, but it is a potentially powerful model. For example, we might be able to portray niched GAs as *hillclimbers* in the space of populations assuming F' as a population fitness function. Taking partial derivatives of the smooth function F' might give us the gradients that a niched GA follows. In addition, the summed population fitness function F has proven useful to many analyses of simple GAs, such as Mahfoud and Goldberg's (1992) proof of the global convergence of their GA[9]. Similarly, a population fitness function such as F' might help in future to prove some important analytical results for niching as well.

At the very least, it would be remarkable to be able to characterize a diverse, dynamically stable, cooperative population as an optimum of a (population) fitness function. Many of us have assumed that once interaction and cooperation were allowed into our GA, the notion of population optimization would have to be abandoned. Perhaps that intuition, like many of our earlier EC intuitions, was wrong.

6.4 Summary

In this chapter we conjectured that niching equilibrium, whose robust existence was proved in the previous two chapters, is not only useful but meaningful as well. We speculated that niching equilibrium could be seen as a solution to a hard set of non-linear equations (for resource

[9]Specifically, Mahfoud and Goldberg consider a population to be a "superstring" of all N individuals concatenated, with the superstring's fitness defined as the sum of all individual fitnesses. Then the global optimum of the superstring must be the string consisting of N copies of the global optimum of the original string-level function. They then map the convergence proof of the metropolis (simulated annealing) algorithm to their PRSA algorithm, to prove that with a non-zero mutation rate, their PRSA is also guaranteed to find the globally optimal superstring, and therefore the globally optimal string.

sharing), and that equilibrium helps GA search by maintaining schemata in proportions that balance the tradeoff between exploration and exploitation (for perfect sharing). And we suggested that sharing can even be seen as optimizing a population, if a suitable population-based fitness function can be found. For perfect sharing, we found that the product of individual (shared) fitnesses is indeed uniquely maximal at and only at perfect sharing equilibrium, while the more traditional population fitness function (the sum of individual fitnesses) does not have a local optimum at niching equilibrium. Finally, we have provided some empirical evidence of the search benefits of sharing, and of the apparent complexities of finding niching equilibrium. Together our analyses and limited evidence do not prove the significance of sharing equilibrium in general, but show at least that niching equilibrium is more meaningful and potentially useful than a random or even uniform population distribution. We hope these indicators will motivate further analyses of niching.

Chapter 7

Further Analyses of Niching

7.1 Introduction

In this chapter, we group together several disparate analyses of niching that provide further insight into the niching phenomenon. Some of the analyses are speculative and constitute ideas for future work, others simply formalize an intuitive argument, while others test promising techniques for adapting and exploiting the power of niching. But all are separate analyses from the central work of this dissertation, which consists of the timing analyses of the two-niche case considered in chapters four and five of this dissertation.

In the first section of this chapter we explore the possible extensions of the two-niche models to the general multi-niche case. We next consider some of the criticisms often made of the sharing approach, responding to such criticisms by calculating the exact costs of sharing, and then probing the real limitations of the approach by considering several variations on the technique to enhance and extend sharing's capabilities.

7.2 Extending the Equilibrium Analysis to Multiple ($k > 2$) Niches

A full model of niching should be valuable in predicting, and helping us control, complicated situations of many competing and "cooperating" niches. It should help us identify cooperating (non-competing) *sets*, or "corporations", of individuals. For example, in Figure 7.1, two sets of niches, set **A**, and set **B**, are both candidates for covering the space well with little overlap.

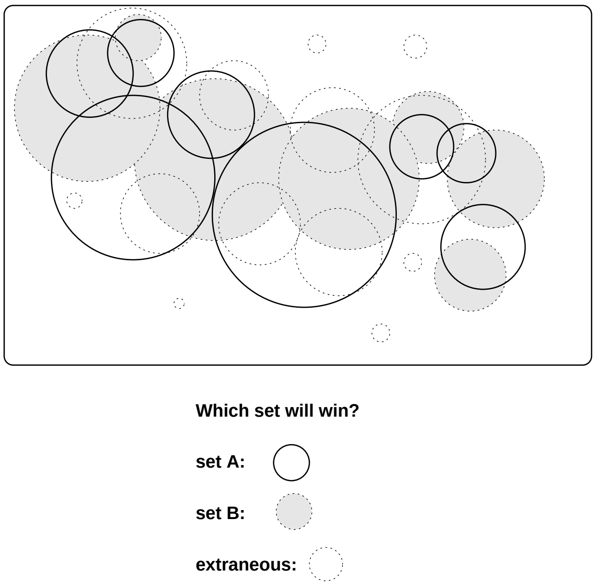

Figure 7.1: A complete, general model of niching should be able to tell us which *set* of cooperating (non-competing) niches above will emerge victorious.

Sets **A** and **B** themselves overlap with each other, and therefore compete. Given a certain population size and number of generations, will one win out over the other? Will set **A** or set **B** fatally lose a niche before the other does? If we want one to survive and the other to disappear, what settings of GA parameter values should we choose, and how long must we wait?

It is hoped that the equilibrium analysis conducted in this dissertation, which helped us answer such questions for the two-niche case, can be extended to help us answer these questions for the arbitrary k-niche cases. For example, niching convergence time for a large number of niches should be equal to the slowest convergence time for any of the isolated, overlapped groups, which in turn might be dominated by the *pair* of niches with greatest overlap within that group.

In the rest of this section we propose and explore a bit some possible guidelines for future generalization of the two-niche analyses to multiple niche situations. We show the independence of niches and "niche clusters" that do not overlap, and thereby suggest that any results on clusters of overlapping niches immediately generalizes to the case of multiple clusters, which is the most general case. We then propose a bounding analysis of all possible niche clusters by considering two bounding cases: that of maximum *degree* of overlap, and that of maximum *order* of overlap.

7.2.1 Independence of Multiple Niche Clusters

One way to decompose the general k-niche case is to divide it into subgroups of overlapping niches. In other words, given a set of k niches to analyze, with arbitrary overlap between any pair of niches, first divide the set into subsets of "non-related" niches (not linked by overlap), just as one decomposes an undirected graph into isolated components. It can be shown that for resource sharing, niching behavior in each isolated component is completely independent of the other components (subsets of niches).

Thus modeling the sets of overlapped niches should be sufficient for modeling all combinations of overlapping and non-overlapping niches. This is the decomposition/integration shown near the bottom of Figure 7.2. It assumes that once we have a model of arbitrary groups of overlapped niches, we can easily generalize that to multiple groups of overlapped niches.

Our results for the two-niche case give us reason to believe in this "independence". After all, niching equilibrium between pairs of non-overlapping niches occurs immediately (within one generation), while between overlapped niches convergence takes logarithmic time (in N, population size). So we might suspect that any "inter-cluster" questions of population distribution would be resolved on a much faster time scale than conflicts within a cluster. Indeed, we can better articulate these intuitions by actually demonstrating such cluster independence for resource sharing.

Consider the $k = 4$ niche case depicted in Figure 7.3. Recall that under our simple but natural definition of resource sharing each discrete unit of a particular resource is divided equally among all competitors. Thus in Figure 7.3, the total resources of f_{AB} are divided equally among individuals of type **A** and of type **B**, while the resources of $f_A - f_{AB}$ are evenly divided among the **A**s, so that

$$f_{sh,A} = \frac{f_A - f_{AB}}{n_A} + \frac{f_{AB}}{n_A + n_B}.$$

Since we calculate shared fitness as the sum of all shares of all resources that an individual can garner, the total of all shared fitnesses must be constant. That is, resource sharing is a

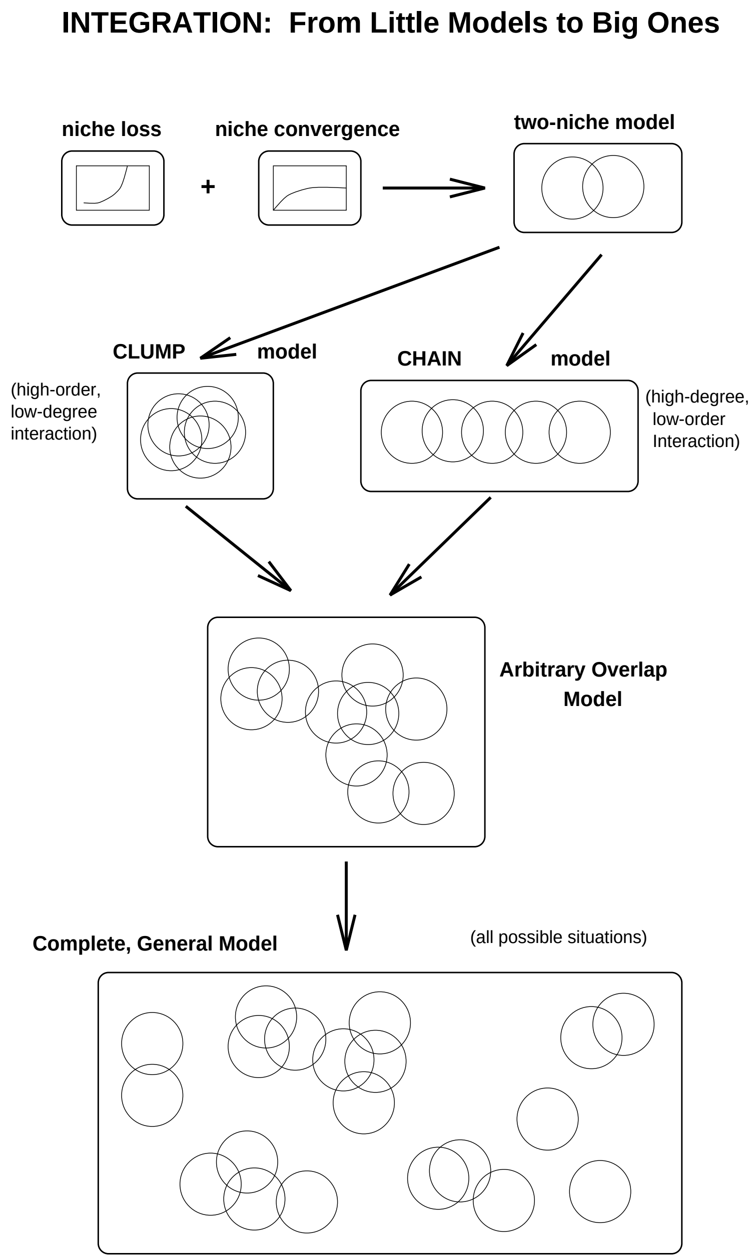

Figure 7.2: Planned breakdown and integration of little models into ever more complete models.

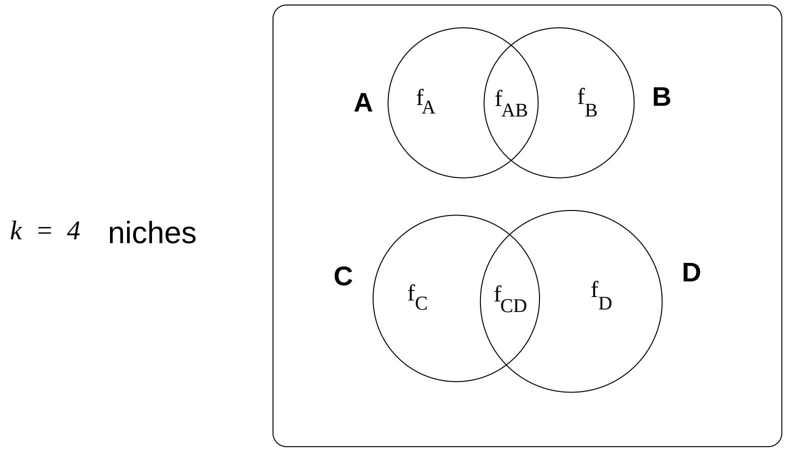

Figure 7.3: Pairs of overlapping niches, **A** and **B**, and **C** and **D**, form independent "niche clusters" **AB** and **CD** that converge to equilibrium at their own internal rates, under resource sharing.

zero-sum game! We have noted this earlier in Chapter 6 when discussing the optimization of a population: the sum of fitness for the population is constant, and so tells us nothing of the dynamics of selection under sharing, such as convergence to an equilibrium[1]. Thus the distribution of the population does not matter, nor does the total population size, *as long as all of the resources are covered by at least one member of the population!* Or, to put it another way, as long as all niches are present, the actual distribution of the population merely changes the distribution of resources among the individuals. Resources are conserved.

To illustrate this conservation of resources, we sum the shared fitnesses in the example in Figure 7.3. We have $n_A + n_B + n_C + n_D = N$, and any time. Given the overlap situation depicted, our sum of shared resources over the entire population follows:

$$\begin{aligned}\textstyle\sum_{X=A,B,C,D} n_X f_{X,sh} &= n_A(\tfrac{f_A - f_{AB}}{n_A} + \tfrac{f_{AB}}{n_A+n_B}) + n_B(\tfrac{f_B - f_{AB}}{n_B} + \tfrac{f_{AB}}{n_A+n_B}) \\ &\quad + n_C(\tfrac{f_C - f_{CD}}{n_C} + \tfrac{f_{CD}}{n_C+n_D}) + n_D(\tfrac{f_D - f_{CD}}{n_C} + \tfrac{f_{CD}}{n_C+n_D}).\end{aligned}$$

Multiplying and simplifying the sum above yields the sum of all fitnesses (resources) with the population numbers n_X canceling:

$$\sum_{X=A,B,C,D} n_X f_{X,sh} = f_A + f_B - f_{AB} + f_C + f_D - f_{CD},$$

[1]Hence we searched out another scalar measure of population fitness (i.e., the product of fitnesses) as an indicator of population convergence.

which of course is constant over time.

One implication of this conservation of resources for the multiple niche case is on the expected proportion equation for selection. Assuming proportionate selection, recall that the expected number of **A**s in the next generation (time $t+1$) is a function of the population distribution at generation t:

$$E[n_{A,t+1}] = \frac{n_{A,t} f_{A,sh,t}}{\sum_{X \in P} n_{X,t} f_{X,sh,t}} N,$$

where $\sum_{X \in P} n_{X,t} f_{X,sh,t}$ is the sum of shared fitnesses of all species X in the population P at generation t. It is this sum that stays constant under resource sharing, so that the expected number of copies of **A** for example is entirely and solely dependent on the current number of copies of **A** and on its shared fitness. The shared fitness of **A** in turn is influenced only by overlapping niches. That is, the number of copies of competing species only, affect **A**. In the example in Figure 7.3, only $n_{A,t}$ and $n_{B,t}$ affect $f_{A,sh,t}$, while $n_{C,t}$ and $n_{D,t}$ do not.

Another way to view this situation is from the point of view of the other niches **C** and **D** for example. They "view" the battle between **A** and **B** as occurring within a single niche with which they do not interact. Adding up the expected number of **A**s AND **B**s,

$$E[n_{A,t+1}] + E[n_{B,t+1}] = \frac{n_{A,t} f_{A,sh,t} + n_{B,t} f_{B,sh,t}}{\sum_{X=A,B,C,D} n_{X,t} f_{X,sh,t}} N,$$

which expands to:

$$= \frac{n_A(\frac{f_A - f_{AB}}{n_A} + \frac{f_{AB}}{n_A + n_B}) + n_B(\frac{f_B - f_{AB}}{n_B} + \frac{f_{AB}}{n_A + n_B})}{\sum_{X=A,B,C,D} n_{X,t} f_{X,sh,t}} N,$$

which simplifies to constant (with respect to time):

$$= \frac{f_A + f_B - f_{AB}}{f_A + f_B - f_{AB} + f_C + f_D - f_{CD}} N.$$

Thus immediately (i.e., within one generation) the "superniche" (or "cluster") **AB** gets its share (above) of the population N, as does the cluster **CD**, both acting like single niches with respect to each other. Further resolution of an equilibrium between the overlapping pairs takes place "internally", and, in this example at least, would be described by our two-niche models.

So niching convergence and equilibrium can be analyzed separately and independently for each "isolated' niche cluster, at least under resource sharing with selection only. Other operators (e.g., recombination) and interactions (e.g., other forms of cooperation) might involve cross-cluster effects. In particular, normal GA crossover treats the entire population as a common mating pool, so that the distributions within clusters can affect the expected offspring across clusters. To reduce the negative aspects of these cross-cluster interactions, many researchers have tried enforcing mating restrictions to discourage or prohibit crossover between radically different chromosomes or genotypes[2].

The independence of niche clusters under selection in resource sharing allows us to reduce the general k-niche case to various cases of clusters of overlapping niches. In the next section we attempt to divide up all possible cases of such clusters, and thence to bound them using dominating, or "bounding", cases of extreme overlap.

7.2.2 Possible Bounding Cases for Niche Clusters

Now that we have shown the independence of isolated (non-overlapped) clusters of niches, what is left to analyze are all possible situations of k niches overlapped such that all are linked either directly or indirectly. Specifically, Figure 7.4 depicts all possible overlap situations (for a given k number of niches) arrayed in a plane. Ordering the space of all possible overlap situations along a two-dimensional plane facilitates the decomposition of the modeling task. Specifically, Figure 7.4 shows how overlap situations can be measured by the *degree* of overlap, and by the *order* of overlap. These two characteristics seem to dominate niche convergence and maintenance time calculations. We can develop small models of the two extremes along those two dimensions (e.g., $k-1$ degree, meaning all k niches are linked in a long, linear chain of overlapping niches, along which niching effects must propagate, or order k, where each of the k niches is overlapped with every other). Such *bounding models* should bound the space of all possible k niche overlaps. I conjecture that extremely high degree, low-order, (CHAINed) overlapped groups, and extremely high-order, low-degree, (CLUMPed) overlapped groups will dominate the time scale analyses of a particular configuration of niches.

[2]It is not yet clear that mating restrictions result in a net gain in performance, as such strategies can degrade the positive aspects of "cross-cluster" mating: improved search! Deb (1989) explicitly recognized this tradeoff between the exploitation of existing niches and the exploration for new ones, which is more recently discussed by Mahfoud (1995b).

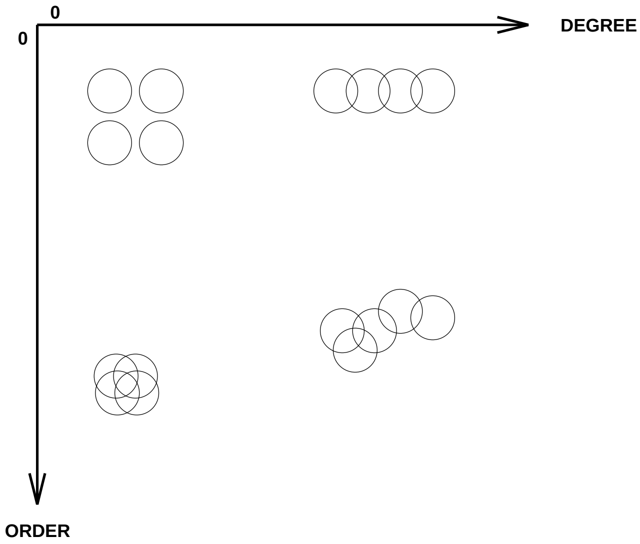

Figure 7.4: All possible configurations of k overlapping niches can be enumerated according to the *order* and *degree* of overlap.

The two bounding k-niche cases (CHAIN and CLUMP) are shown in Figure 7.2 as generalizations of the arbitrary two-niche model. In turn, it is hoped that the two k-niche bounding models can be used to model the *arbitrary overlap* case, which in turn is easily generalized (really a linear combination) to a complete, general model.

7.3 Computational Cost of Sharing

In this section we take the time to conduct a simple, straightforward analysis of the computational efficiency of sharing. We are motivated by recent criticisms of fitness sharing in particular that have led to some simple misunderstandings over the efficiency of this type of niching.

7.3.1 Criticisms of Sharing

Since the introduction of *fitness sharing* to genetic algorithms by Goldberg and Richardson (1987), to induce *niching* for the maintenance of useful diversity, many studies and applications of GAs with sharing-based niching have indicated the efficacy of this approach. During the same period, and most recently over the last two or three years, many alternative niching methods have emerged (e.g., versions of the original *crowding* scheme (De Jong, 1975; Mahfoud, 1992), deme-based distributed GAs (e.g., Collins & Jefferson, 1991), sequential niching (Beasley, Bull, & Martin, 1993), restricted tournament selection (Harik, 1994), etc.). In the course of introducing some of these alternative niching approaches, the authors usually give a brief overview of fitness sharing which includes a short list of its limitations. These stated limitations are then used as partial justification for developing the alternative niching method. But the listed limitations, such as the "$O(N^2)$ computations" argument, or the "even spacing of niches" requirement, are often oversimplified and can be misleading. Many readers, including some of the most recent authors of other niching methods, now simply assume that these limitations are fatal flaws of the sharing based approach.

The critical observations of fitness sharing generally fall into two major categories[3]:

- Sharing is computationally expensive, $O(N^2)$, and
- Sharing requires *a priori* knowledge of niche distribution (e.g., number of niches, distance between niches, evenly spaced niches).

In this section we address the first criticism by examining the *actual* cost of sharing in terms of basic computer operations and in terms of (objective) fitness function evaluations. We will deal with the second criticism in the next section.

7.3.2 Computational efficiency: the N^2 argument

The first criticism is the easiest to analyze, because fitness sharing is indeed $O(N^2)$ computer operations. However, the extent to which that should be considered expensive is a more subtle matter. There are three responses to the $O(N^2)$ argument:

1. comparisons versus function evaluations (apples and oranges),
2. $O(N)$ samples can approximate the $O(N^2)$ calculation, and
3. maintaining diversity *requires* at least $O(N^2)$ comparisons!

The first response is the most obvious. The *exact* calculation of niche counts requires N comparisons for each of the N individuals in the population[4]. But these are simple comparisons,

[3]For just a few of many examples of these criticisms, see (Davidor, 1991; Beasley, Bull, & Martin, 1993; Smith, Forrest, & Perelson, 1993; Yin & Germay, 1993;).

[4]To be more exact, we really require $(N-1)^2/2$ comparisons, if we assume our genotypic or phenotypic distance is a metric H, since then $H(x,x)=0$ and $H(x,y)=H(y,x)$.

not fitness function evaluations, the unit often employed in measuring and comparing the time complexities of various EC operators and algorithms. Although obvious, this distinction is often left out when citing the N^2 argument.

The proper comparison, at least on a serial machine, is to compare the computational cost of a single fitness function evaluation to that of N comparisons, since that is the computational burden of fitness sharing *per individual* in the population. We assume a relatively inexpensive comparison, such as ℓ bit-wise XORs for genotypic sharing[5]. If the computational complexity of a particular fitness function is much greater than $O(N\ell)$ logic operations, then the additional cost of sharing will be negligible, at least on a serial architecture.

Many fitness functions are much more expensive than niche counting, but many are not, such as table lookups. For example, five (identical) table lookups comprised the evaluation function for the massively multimodal problem in (Goldberg, Deb, & Horn, 1992), and so the cost of sharing significantly lengthened their run times on serial machines. For this reason the authors implemented *sampling*, as suggested by Goldberg and Richardson (1987) originally[6], to approximate niche counts. We note that some classic open problems also have very fast (table lookup) fitness functions, including the set covering, traveling salesman, and knapsack problems.

As noted above, sampling to approximate niche counts greatly reduces the time complexity of the sharing to $O(\rho N)$ per generation, where ρ is the number of individuals sampled per niche count calculation. Oei, Goldberg, and Chang (1991) suggest using ρ proportionate to q, the expected/desired number of niches[7]. Goldberg, Horn, & Deb (1992) use $\rho = 200$ with a population size of $N = 5000$ with success. Beasley, Bull, and Martin (1993) estimate sharing's complexity assuming ρ sampling, but unfortunately they also assume a population size $N = qN'$, where N' is the population size needed to find one peak/niche with a simple GA, and q is the number of desired peaks. This conservative, first order "guestimate" of a sufficient population size for search in a GA with sharing is suggested by Goldberg (1989b), but we show evidence

[5] For phenotypic sharing, the decoding of the genome into the phenome might be expensive, but must be done anyway to calculate the fitness. We can therefore assume a fast comparison for phenotypic sharing as well.

[6] This suggestion is analyzed later by Oei, Goldberg, and Chang (1991).

[7] Mahfoud (1995b) calls for analysis of the noise introduced by sampling for niche counts, to determine the minimum sample size possible.

later in this paper that sharing can actually improve on simple GA search without any increase in population size (on a particular problem).

Finally, niching (and indeed any form of cooperation) seems inherently $O(\geq N^2)$. Simulating natural environments on a computer is expensive. Calculating all the effects that individuals can have on each other consumes a great deal of CPU time. Although many of these calculations, such as resource usage, can be implemented as highly parallelized local operations, they all implement an inherently global computation, in which every individual can influence every other. Sharing simulates one critical type of environmental interaction: the sharing of limited resources. Given r resources to be shared, the GA (or nature) *must* perform (or approximate) an $O(rN)$ calculation[8] to fairly distribute the resource. Compared with the tremendous amount of computation required to simulate more complicated interactions (e.g., battles, parasitism, preying, creation and exchange of artifacts, etc.) the $O(N^2)$ or $O(rN)$ comparisons needed to implement fitness or resource sharing make sharing in general an inexpensive but effective type of cooperative evolver[9].

7.4 Advanced Fitness Sharing Techniques

In this section we answer the major criticism of fitness sharing, which concerns the sensitivity of the algorithm's performance to exact *a priori* knowledge of niche distribution. We address this issue primarily by exploring a number of modifications to the basic fitness sharing technique to demonstrate that the limits of the algorithm are not hard or brittle, and that limits can be stretched, and that at such limits the performance of the algorithm degrades gracefully in a controlled manner.

7.4.1 Prior knowledge of niche distribution

Contrary to many statements in the literature, sharing makes no explicit assumptions, such as number of niches. Several authors have mistaken some of the simplifying assumptions Deb

[8]Since we do not have explicit resources in function observation, we must use $O(NN)$ comparisons.

[9]Some niching methods, such as distributed, isolated demes, or deterministic crowding (Mahfoud, 1992), are $O(N)$, but it is disputable whether such algorithms are performing the global computation necessary for true proportionate niching. For example, both deming and deterministic crowding allocate subpopulations to niches according to the distributions of their initial random populations, rather than relative niche fitnesses, as sharing does.

makes in his analyses of niching (Deb, 1989) as assumptions of the algorithm. Rather, the performance of sharing degrades gracefully with increasing number of niches, increasing niche overlap, and increasing fitness difference between best-fit and worst-fit, desired niches. In general there is no threshold of niche overlap, or niche fitness difference, at which niche maintenance collapses. In this thesis we have shown analytically the gradual degradation in niche maintenance times and steady state distributions as the niche overlap and niche fitness ratios are increased. Rather than address a list of false statements about the *a priori* knowledge assumed by sharing, we focus on the actual limitations of sharing, in the next section.

7.4.2 The Real Limitations

Clearly, it is overly simplistic to say that sharing can maintain only a certain number of optima, or can only distinguish optima with a minimum separation or fitness difference between them. Rather, sharing handles the two conflicting objectives of high quality and diversity. When these two objectives can be met simultaneously, sharing has little difficulty. Thus high quality solutions that are very different from each other are the least difficult niches to maintain. But when these objectives conflict severely, sharing is less likely to give us the exact set of solutions we want. If the desired highly fit optima are very similar, and some undesired but almost-as-fit suboptima are located far from the desired optima, sharing can maintain suboptimal niches at the expense of the desired optima.

Under such conditions sharing does not "fail". Rather, sharing finds solutions that are both different and highly fit. If we disagree with the way sharing "weights" these twin objectives, we must find a way of adjusting sharing's priorities to favor either quality or diversity more. In the example above, σ_{sh} can be decreased and N increased to ensure that all of the closely spaced, desired niches are well-populated, along with many of the niches not desired. But if the number of undesired optima is very large, and they are very different from the globals, adjustments to σ_{sh} and N alone might not be adequate, as Goldberg, Deb, and Horn (1992) find.

7.4.2.1 Problem F1: difficult for sharing

The massively multimodal function (Goldberg, Deb, & Horn, 1992) is an extreme case of conflict between the twin objectives of high quality and diversity. Their $\ell = 30$ bit function (F1) contains over 5 million local optima, of which 32 are global. The rest are highly attractive deceptive local optima, many of which are located very far from any global. The minimal separation between globals is 6 bits, while about 3.5 million of the local optima are located $\ell/2 = 15$ bits from the nearest global. Being so distant from the globals and fairly well fit themselves, these and other local optima normally attract a small subpopulation each. Even with only one representative at each local optimum, a population size of over 5 million would be needed to ensure adequate subpopulation sizes at all of the 32 globals. Because the globals are so close together, relative to their distances to the local optima, setting σ_{sh} large enough (e.g., $\sigma_{sh} = 17$ to ensure niche formation only at globals also causes globals to compete with each. This competition results in the maintenance of only four globals, the maximum number that can be $\geq \sigma_{sh}$ bits apart (18 bits in this case) (Goldberg, Deb, & Horn, 1992).

Thus F1 is extremely difficult for sharing, *if we expect sharing to maintain all the globals*, because of the presence of massive numbers of high quality local optima well-separated from the globals.

7.4.2.2 Powersharing

Goldberg, Deb, and Horn (1992) recognize the need for "augmenting" the standard fitness sharing algorithm to effectively "solve" F1. To emphasize the importance of high quality, and therefore discourage niche formation at local optima, they suggest and implement *powersharing*. Under powersharing, the objective (i.e.., unshared) fitness is raised to some exponent $\beta_{sh} > 1$, and then sharing is performed normally. This effectively increases the "carrying capacity" of higher fit niches at the expense of less fit ones. Goldberg, Deb, and Horn (1992) find that $\beta_{sh} = 15$ allows a population of size $N = 5000$ to maintain reasonable subpopulations at all 32 global optima.

Figure 7.5 shows the successful maintenance of niches at all 32 global optima. The upper curve represents the maximum subpopulation size at any of the 32 global optima for each of the first 100 generations. The lower curve is the minimum subpopulation size at any generation. The middle curve is the average subpopulation size. Thus the actual subpopulation size of any single niche wanders within the upper and lower curves, but the mean subpopulation is fairly steady[10].

[10] The subpopulation size at a global optimum is the actual number of copies of that global present in the population.

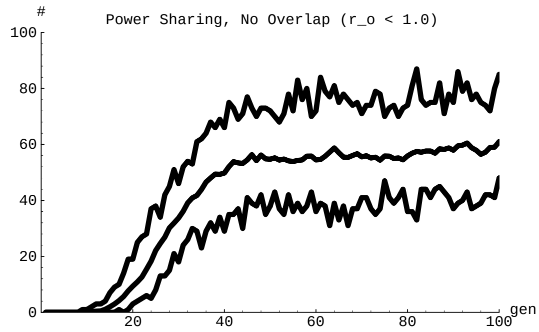

Figure 7.5: Powersharing, with exponent 15, and $N = 5000$, successfully maintains all 32 globals (Goldberg, Deb, & Horn, 1992). The middle plot is the average subpopulation size at the global optima at each generation from 0 to 100. The upper curve tracks the maximum subpopulation size at each generation, while the lower curve is the minimum size. Thus the actual subpopulation at any particular global fluctuates between the upper and lower curves.

Some objections to powersharing include the criticisms that the exponentiation of the fitness function is unnatural, that it introduces yet another parameter to be tweaked, and that it represents "cheating". But increasing niche carrying capacity is not unnatural. And adding another parameter is desirable *if* it gives us fine, predictable control over an algorithm's behavior. Here we want to bias sharing toward higher quality at the expense of diversity in much the same way that we might tune the weights or penalties in any scalarization of a multi-objective or multiple constraint fitness "function". (Only here we hope to have a much better idea of the effects of our tuning!) And just as sharing is robust with respect to the setting of niche radius (σ_{sh}) (Horn, 1993), powersharing is robust with respect to the exponent β_{sh} of fitness. Figure 7.6 illustrates how powersharing degrades gracefully with decreasing β_{sh}. Finally, exponential scaling seems a fairly general approach that requires very little, if any, problem specific knowledge to apply.

7.4.2.3 Rootsharing

Some might object to powersharing as a general augmentation of fitness sharing, since it directly manipulates the (problem specific) fitness function. But we do not need to touch the fitness function to achieve the same bias toward objective fitness that powersharing achieves. Rather than increase the importance of the objective fitness during selection, alternatively we can decrease the role (i.e., the degradation) of niching. Specifically, instead of exponentiating the

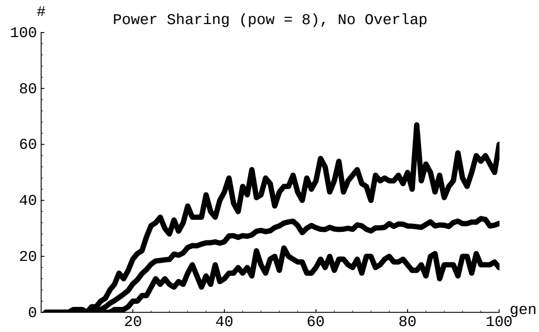

Figure 7.6: Power sharing with exponent 8 shows a marked degradation in the ability to maintain all 32 globals, but its success indicates that power sharing is robust with respect to changes in the exponent of fitness.

numerator (objective fitness f) in the shared fitness computation $f_{sh} = f^{\beta_{sh}}/m$, we can simply take the β_{sh}^{th} root of the denominator, niche count m. Dubbing this technique *rootsharing*, we substitute $f_{sh} = f/\sqrt[\beta_{sh}]{m}$ for the shared fitness calculation.

We show that rootsharing is equivalent to powersharing. Goldberg, Deb, and Horn (1992) use powersharing to change the population distribution at *sharing equilibrium*, which is when all shared fitnesses are at equilibrium: $\frac{f_i}{m_i} = \frac{f_j}{m_j}$, $\forall i, j \in \mathcal{P}$. Under powersharing,

$$\frac{f_i^{\beta_{sh}}}{m_i} = \frac{f_j^{\beta_{sh}}}{m_j}, \tag{7.1}$$

thus allowing the user to increase the equilibrium niche count m_i for higher fit niches f_i by simple increasing β_{sh}. Taking the β_{sh}^{th} root of both sides of Equation 7.1,

$$\frac{f_i}{\sqrt[\beta_{sh}]{m_i}} = \frac{f_j}{\sqrt[\beta_{sh}]{m_j}}, \tag{7.2}$$

which is the equilibrium of rootsharing. Thus powersharing and rootsharing are equivalent at equilibrium. Replacing the equal signs in Equations 7.1 and 7.2 with an inequality, it is clear that powersharing and rootsharing induce the same shared fitness ordering on any given population. So we expect rootsharing and powersharing to behave exactly the same under any rank-based selection method.

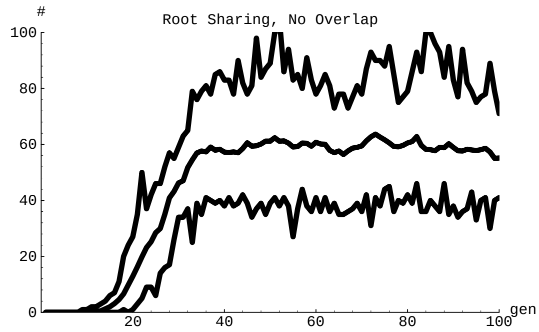

Figure 7.7: As expected, root sharing with root 15 and $N = 5000$, shows the same performance as power sharing with exponent 15, as shown above. Both the power sharing and root sharing runs began with the same random population (i.e., used the same random seed).

Figure 7.7 plots the performance of rootsharing with $\beta_{sh} = 15$ on F1, with all the other GA parameters set the same as in the powersharing ($\beta_{sh} = 15$) run shown in Figure 7.5, including the same random initial population. The performance is almost the same[11].

Thus power and rootsharing are the same niching technique, at least under rank based selection. One can view the strategy of this technique as either increasing the carrying capacity of niches (powersharing), or, equivalently, decreasing the amount or degree of sharing required (rootsharing). Other techniques could probably implement this strategy just as effectively. For example, with an $\alpha_{sh} << 1$ and $\sigma_{sh} >> \ell$, the resulting sharing function $Sh(d)$ would tend to emphasize objective (unshared) fitness. In the limit of $\sigma_{sh} \to \infty$, the effect of sharing disappears completely and we return to the convergence of a simple GA.

7.4.2.4 Elitist sharing

We can also view power/rootsharing as implementing an elitist strategy, promoting fitness over diversity. Powersharing and rootsharing effectively increase the "weighting" of objective fitness over diversity (i.e., niche count) in the two criteria optimization problem. We can take this elitist strategy to an extreme, in a further illustration of the flexibility of the general sharing approach to function optimization.

[11] The differences are attributable to rounding error differences in taking the 15^{th} root of the niche count versus raising fitness to the 15^{th} power. Such differences should not appear at lower β_{sh}.

Horn, Nafpliotis, and Goldberg (1994)'s niched Pareto GA attacks a two objective problem by ranking individuals (through local *Pareto tournaments*) according to the Pareto optimality criterion, thereby inducing equivalence classes. The authors then use sharing within the equivalence classes to spread out the subpopulation in each class. Cieniawski (1993), Fonseca and Fleming (1993), and Srinivas (1994) and Deb (1995) implement similar algorithms with success. By first partially ordering individuals according to some fitness vector, performing selection *between* the induced equivalence classes only, and then implementing *equivalence class sharing* only within each equivalence class, these authors were able to maintain a large and diverse population covering the highest ranked equivalence class (in this case the non-dominated set, or Pareto optimal set).

Inspired by these successes, we attempt to apply equivalence class sharing in scalar function optimization through the use of *elitist sharing*. Under this version of sharing, we conduct binary tournaments as usual, randomly selecting two competitors i, j from the current population. But we only perform sharing if their fitness difference $|f_i - f_j|$ is less than some fixed δ. If $|f_i - f_j| \geq \delta$, we simply choose the larger as the winner. If the difference is smaller than δ, we perform "normal" fitness sharing (i.e., no exponentiation of fitnesses, nor root taking of niche counts). Setting $\delta = 0.03$, which is just less than the actual fitness difference between F1's global optima and the next-best local optima, Figure 7.8 shows successful performance on F1, similar to the performance of rootsharing and powersharing[12].

Since it is doubtful that elitist sharing is as robust with respect to the setting of δ as power and rootsharing are to the setting of β_{sh}, we recommend power or rootsharing over elitist sharing for control over the fitness/diversity tradeoff. The elitist sharing algorithm does help us, however, to understand the effects of power and rootsharing.

7.5 Summary

In this chapter we have made some early progress in extending our niching models to $k > 2$ niches. We have looked at some of the complexities to be encountered, but we have also outlined a plan for incrementally expanding the current analysis, and bounding all the possible

[12]Since F1 has 32 (equal valued) global optima, setting $\delta = 0$ should also work. With $\delta = 0$, elitist sharing is equivalent to the niched Pareto GA (NPGA) applied to a (single objective) function optimization problem. I.e., the NPGA with $k = 1$ objectives reduces to elitist sharing with $\delta = 0$.

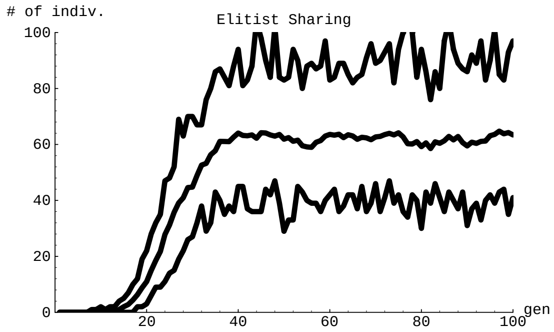

Figure 7.8: Elitist sharing, with $N = 5000$ and $\delta = 0.03$ can stably maintain substantial subpopulations at all 32 global optima with no exponentiation of the fitness function, nor any change to the niche count.

niching situations by the extreme cases of niche clumps and niche chains. We showed that "independent" groups of niches, with no overlaps between the two groups, could be analyzed separately. Also in this chapter, we responded to two common criticisms of sharing that have become oversimplified and inaccurate. The first criticism concerned the computation cost, $O(N^2)$, of computing niche counts for sharing. We made the point that this cost can be very small, even insignificant, compared with the cost of evaluation the objective fitness function. We also showed how sampling can approximate the niche counts using $O(\rho N \ll N)$ comparisons. The second criticism of sharing, that it is brittle with respect to niche distribution, and therefore requires *a priori* knowledge of the number, separation, and spacing of the niches. In response, we demonstrated the robustness of sharing to niche overlap and massive multimodality, and its graceful degradation in performance under increasingly inaccurate *a priori* knowledge. We also illustrated the flexibility of the basic sharing mechanism by incorporating powersharing, rootsharing, and elitist sharing, to solve the difficult, massively multimodal problem with fitness sharing.

The additional analyses presented in this chapter are incomplete. This chapter really serves as an introduction to, and suggestions for, "Future Work".

Chapter 8

The Utility of Equilibrium (Applications)

In this chapter we present three applications of fitness sharing to artificial and real-world problems. The purpose of these demonstrations is to present a range of very different problem areas within which sharing can be of benefit:

- multimodal function optimization (a pipeline network design problem),
- multi objective decision making (a groundwater remediation problem), and
- layout and packing (an artificial one-dimensional layout problem).

Although the theoretical results developed in previous chapters are not ready for direct use in practical problem solving here, some earlier discoveries and insights will be made vivid here (e.g., level-three cooperation, the improvement of search by niching, etc.). The power, robustness, and flexibility of sharing are brought out by these experiments. These empirical data will serve at the very least to inspire and guide further work on the nature of niching.

8.1 Function Optimization: (Search and Multimodality)

In this section we present an application of fitness sharing to multimodal function optimization. It is well-known that this problem has many local optima, several of which are of high merit.

(New and better optima are found every few years; see below.) Fitness sharing is applied here with two goals in mind:

1. find multiple optima, and
2. improve the search for the very best optimum.

We are really demonstrating two capabilities of niching here, the ability to find and maintain several good optima, and the ability to improve our chances of finding the very best solution. While the second goal is generally assumed (i.e., we always want to find the best solution on a single-objective optimization problem), the first goal might or might not be present. In many cases, such as on the NYC tunnels problem below, we are interested in multiple optima because (a) we would like to find several alternative solutions of near or equal (high) utility, and/or (b) we are interested in seeing the first best, second best, third best, etc., for historical reasons, or perhaps to raise our confidence in the thoroughness of the GA search. Whatever the exact motivation, the following application of fitness sharing to the NYC tunnels problem demonstrates improved search *and* the ability to find and keep all previous best-known solutions (i.e., local optima), simultaneously.

8.1.1 The New York City Tunnels Problem

Our objective function is an open problem in water distribution pipeline design, known as the *New York City tunnels problem* (or simply "the NYC problem"). It was developed by Schaake and Lai (1969). Murphy, Simpson, and Dandy (1993) successfully applied a GA to the problem to find three new optima, each superior to the previous known best solution by Morgan and Goulter (1985). For details of the problem, the reader is referred to (Murphy, Simpson, & Dandy, 1993). We provide a brief description below.

The problem is to redesign a pipeline network for distribution of fresh water throughout Manhattan. The goal is to minimize the cost of installing new pipes while meeting minimum water pressure requirements at sixteen end nodes. The old network consists of twenty one large diameter underground pipes. Each of these pipes can be "duplicated", that is a new pipe can be installed parallel to the existing one, thereby increasing capacity on that "link". For each possible duplication, fifteen choices of new pipe diameter are available. These fifteen choices, combined with the sixteenth choice of not duplicating the pipe (pipe diameter 0, if you will), allows us to code each of the 21 decision variables in four bits, resulting in a total chromosome length of $\ell = 21 * 4 = 84$ bits. Thus the search space is size $2^{84} \approx 1.934 \times 10^{25}$. The NYC problem is still an open one.

The fitness function for the NYC problem is computationally costly, as it involves a simulation of the candidate network's performance under a fixed load to determine pressure at each of the nodes. Pressure head constraint violations are penalized by a high cost added to the total cost of installing the specified duplicate pipes. The larger the pipe diameters chosen for the duplicates, the greater the pressures but the more expensive the installation. The output of the fitness function is a total cost, including installation and penalties. But the final costs we report below are only the installation cost component, since that is the objective function by which we compare solutions found by different algorithms. The objective is to minimize this cost while meeting the pressure head constraints.

8.1.2 The Need for Niching

In 1985, Morgan and Goulter found a new optimum costing $\approx$ \$39.20 million. Murphy, Simpson, and Dandy (1993) used a GA with exponential scaling of the fitness function to find three new optima, worth $\approx$ \$38.80, \$39.06, and \$39.17 million, (call them optima 1, 2, and 3, respectively), all superior to the 1985 solution. A fourth optimum, at $\approx$ \$39.22 million (optimum 4) was also found[1], and a fifth optimum of interest was found at \$39.28 million (optimum 5). However, the authors note that the GA *usually* converged to optima 2 and 3, the second and third known best, which are very similar (both genotypically and phenotypically), being slight variations on the same basic network design. Optima 1, 4, and 5, however, were found less often. These optima are very similar to each other, and very different from optima 2 and 3.

8.1.3 Niching Results

A niched GA with sharing, however, proves to be much more likely to locate all five top optima in a single run. Figure 8.1 illustrates a comparative run between a GA with sharing and one without. For both cases, the GA parameters other than sharing are the same: a simple, generational GA with population size $N = 6000$, probability of single point crossover $p_c = 0.8$,

[1]Although optimum 4 is slightly more expensive than Morgan and Goulter's solution, the latter was shown by Murphy, Simpson, and Dandy (1993) to be "barely" infeasible.

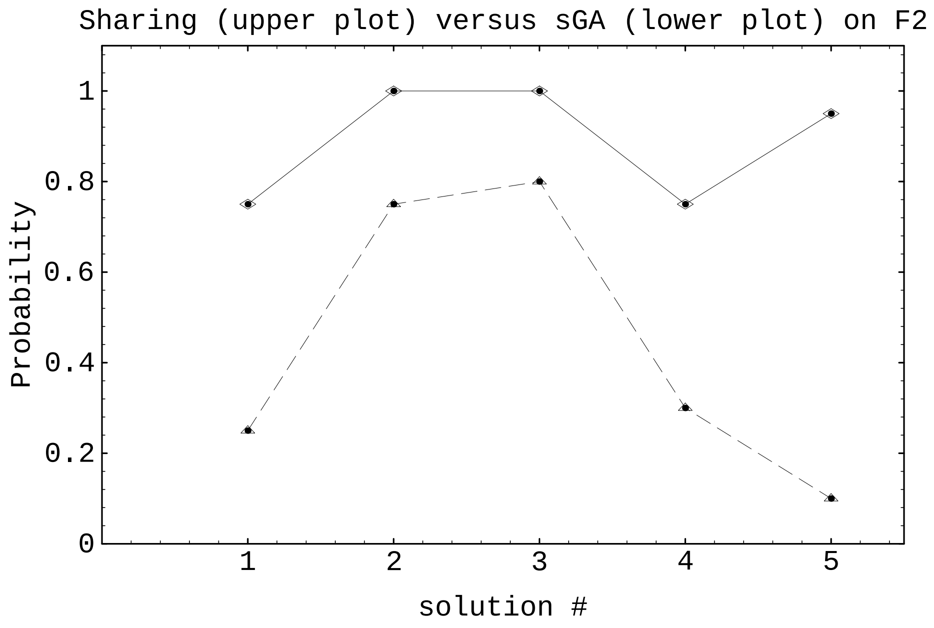

Figure 8.1: Holding all other GA parameters constant (e.g., N, β_{sh}, p_c, etc.), adding sharing increases the likelihood of locating any of the top five known solutions of problem F2. *Probability* is the percentage of 20 different trials in which a particular solution appeared at least once within the first 200 generations.

no mutation ($p_m = 0$), and the fitness function is raised to the power $\beta_{sh} = 20$ (prior to sharing). The GA with sharing uses genotypic comparisons with $\sigma_{sh} = 7$ bits. We run the GA with and without sharing for twenty different trials (i.e., different random initial populations), each time terminating the run at 200 generations. We then examine every individual created during a trial and record the number of trials (out of 20) in which a particular optimum (1-5) is found (i.e., appears in the population at any of the 200 generations). Calling this frequency of appearance "probability", we plot the sampled probabilities of finding the top five known optima in Figure 8.1. Thus each plotted point (k, p) is the percentage p of the twenty trials in which optimum k is found.

The upper curve in Figure 8.1 plots the performance of sharing, while the lower curve plots the performance of the same GA with sharing turned off. It is clear that sharing exhibits a much higher probability of success in locating *any* of the five optima, but especially the more "difficult" optima (1, 4 and 5).

8.2 Multi-objective Optimization

In this section we apply a GA with niching to multi objective problems. We develop a version of the GA specifically designed to find the *Pareto-optimal* set of non-dominated tradeoff solutions, and call this algorithm the *Niched Pareto GA*. This approach is part of recent and apparently growing trend in multi objective GAs that seek a population of alternative compromises to multiple conflicting objectives. Such *Pareto GA*s subsume some method of maintaining quality diversity under selection pressure; in other words, a niching method. Our choice for niching method is of course sharing, but this is a common choice among many recent Pareto GAs published in the last two years (Horn, 1997). Thus many of the lessons learned below should be be of interest to an audience of Pareto GA practitioners.

8.2.1 Introduction

Genetic algorithms (GAs) have been applied almost exclusively to single-attribute[2] problems. But a careful look at many, if not most, of the real-world GA applications reveals that the objective functions are really multiattribute. Typically, the GA user finds some ad-hoc function of the multiple attributes to yield a scalar fitness function, a general approach that we call **aggregative** in the rest of this section.

8.2.1.1 Aggregative Approaches

Often-seen tools for combining multiple attributes are constraints, with associated thresholds and penalty functions, and weights for linear combinations of attribute values. But penalties and weights have proven to be problematic. The final GA solution is usually very sensitive to small changes in the penalty function coefficients and weighting factors.

One approach to improve the GA's handling of multiple attributes is to use a more systematic and robust method for combining the multiple attributes into a single scalar fitness function. At the very least, such a method should account for nonlinearities in attribute interactions. The technique known as MultiAttribute Utility Analysis (MAUA) is a recent development in the field of decision analysis. MAUA prescribes a multiplicative combination of individual *utility*

[2]Throughout this section we use the terms "attribute", "objective", and "criteria" interchangeably. They all describe a scalar value to be maximized or minimized. "Decision variable" refers to the parameters of the problem encoded in the genome of the genetic algorithm.

values, which are functions of attribute values and the uncertainties in those values. Thus MAUA addresses multiple criteria and uncertainty, but does not address the issue of search. The MAUA approach has some limitations, however. First, it only incorporates pairwise, multiplicative nonlinearities and not higher-order nonlinear dependencies. Second, to estimate the coefficients of the singular and pairwise terms, MAUA requires a lengthy interview with a decision maker in which the decision analyst poses a series of lottery questions. Still, MAUA is currently one of the best, most sophisticated of the aggregative approaches.

8.2.1.2 Pareto Approaches

A few studies have tried a different approach to multicriteria optimization with GAs: using the GA to find all possible tradeoffs among the multiple, conflicting objectives. These solutions (tradeoffs) are *non-dominated*, in that there are no other solutions superior in all attributes. In attribute space, the set of non-dominated solutions lie on a surface known as the *Pareto optimal frontier*[3]. The goal of a *Pareto GA* is to find and maintain a representative sampling of solutions on the Pareto front. Previous studies, such as Schaffer's 1984 dissertation (Schaffer, 1984), have succeeded in finding some points on the front, in particular near the extreme ends, where one attribute is optimized. But to find a *representative* sampling, with the final population spread all along the front, niching is required (Goldberg, 1989).

In the remainder of this section, we develop a version of the GA to find the Pareto optimal frontier in a multi-objective problem.

8.2.2 Previous Work

Although there has been much work using weighted sums and penalty functions to turn multi-objective problems into single-attribute problems for the GA, and much work using MAUA and other decision analysis techniques to search small, tractable spaces, there has been comparatively little effort expended on using GAs to search large, intractable *multiattribute* spaces. We have conducted a limited literature search, and have turned up two interesting approaches.

[3]We assume familiarity with the concept of Pareto optimality, but note here that the Pareto front often goes by the names Pareto optimal set, non-dominated frontier, *efficient* points (Sawaragi, Nakayama, & Tanino, 1985), and *admissible* points (Guddat, Vasquez, Tammer, & Wendler, 1985). We endeavor to be consistent in using the word "Pareto" to describe the set, front, frontier, or surface.

In his 1984 doctoral dissertation, Schaffer (1984) proposed his Vector Evaluated GA (VEGA) for finding multiple solutions to multi-objective (vector valued) problems. Schaffer was motivated by the difficulty a "scalar GA" had in the set covering problem. The GA could not maintain diverse rules that together classified a set of patterns. He theorized that it was impossible to assess the quality of rules adequately and fairly with a single scalar value. Thus he created VEGA to find and maintain multiple solutions, each one addressing one of the multiple attributes. VEGA tried to achieve this goal by selecting a fraction of the next generation using one of each of the attributes. Thus, in a two-attribute problem, say cost and reliability, VEGA would select half of the next generation's population using lowest cost as the selection criterion (scalar fitness function) and the other half using highest reliability. Although Schaffer reported some success, VEGA seems capable of finding only extreme points on the Pareto front, where one attribute is maximal, since it never selects according to *tradeoffs* among attributes.

In his review of GA history, including Schaffer's VEGA, Goldberg (1989) suggested the use of non-domination ranking and selection to move a population toward the Pareto front in a multiobjective problem. He also suggested using some kind of niching to keep the GA from converging to a single point on the front. A niching mechanism, such as sharing (Goldberg and Richardson, 1987), would allow the GA to maintain individuals all along the non-dominated frontier.

Ritzel (1992) took Goldberg's advice. He used non-dominated ranking and selection combined with deterministic crowding (Mahfoud, 1992) as the niching mechanism. His problem was an open one in groundwater remediation, with two attributes: cost and reliability. Though the actual Pareto front was unknown, Ritzel used the best tradeoff surface found by a domain-specific algorithm, MICCP, to judge the performance of the GA. Briefly, Ritzel found that selection according to Pareto non-domination was superior to both VEGA and non-domination with deterministic crowding, at least for *finding* points near or on the front found by MICCP. However, Ritzel did not investigate how well these algorithms *maintained* a distribution along their fronts. In addition, Ritzel did not try other niching mechanisms, such as fitness sharing.

8.2.3 Our Strategy: GA $\Rightarrow$ MODM (Pareto Optimal Set)

Our strategy for a multiobjective GA is to use a "Pareto GA" to find all possible tradeoffs (i.e., the Pareto-optimal front) and then to use this as input to one of the *multiobjective decision* *making* (MODM) methods, which are algorithms or techniques such as MAUA, that elicit the multiobjective decisions that the human decision maker (DM) must make in order to choose their single best solution from among the tradeoffs. Thus the GA here is a front-end to the multiobjective decision making process, not participating directly in the decision making, but helping by performing the search necessary to narrow down the choices. Note also that whatever MODM method is used, it does not participate in the search. Search is the GA's job. Again, the GA is used to find the Pareto optimal set P for a multicriteria problem, and then, if necessary, MODM is performed.

Finding P is a challenging problem. Later in this section we develop a GA that is capable of finding P: the *Niched Pareto GA*. Here we discuss how P can help MODM.

8.2.3.1 Usefulness of P

As de Neufville points out in Chapter 20 of (de Neufville, 1990), "...procedures... for multiobjective analysis...have the great advantage of not requiring any explicit measure of preference. They are therefore much simpler [than most MODM methods] and should be used whenever possible." P can be of greater general use than the best solution for a particular decision maker, since P is independent of any decision maker and his or her utility functions. The best solution for each decision maker might be different, but all best solutions will be in P. Therefore, P can be reused with each new decision maker, saving analysis time.

More specifically, knowing the Pareto optimal set P prior to performing MODM could be useful in a number of ways:

- For small P, make MODM unnecessary
- A check on MODM
- Help with MODM (reduce amount of MODM necessary
- Help with group decision making

If P is small enough, the decision makers might be able to directly choose a solution by inspection. Even when P is very large, the shape of the curve (i.e., the front) might make the region of interest obvious. For example, de Neufville (1990) notes how decision makers can often agree that solutions at the "knee" of the front, if such a region exists, represent the best compromise. Choice by direct inspection of P eliminates the need for MODM.

Even if P is too large or the best choice is otherwise unclear, and a MODM method is required, P can help the MODM process. P can be a simple check on the regular MODM procedure, since any solution obtained by any MODM method must be in P. In addition, just a few multiobjective decisions up front could reduce the number of attributes such that the Pareto set P' of the reduced problem is amenable to choice by inspection. Thus, some multiobjective aggregative method could be performed on just a subset of the attributes, perhaps just two, replacing them with a single attribute (the multiattribute utility function of those attributes for that decision maker).

In this section, we do not treat explicitly the issue of group decision making. But we note here that a small P might alleviate the known difficulty of decision analysis with multiple decision makers with very different utility functions. In particular, Arrow has shown that it is in general impossible to find a single aggregative utility function for a group. Nor can we even find, in general, a single partial order of preference. But here P can help. If P is small enough that all of it can be viewed, or we can visualize the surface and focus in on the most promising choices, then our group of decision makers might be able to come to a consensus on the order of preference *for just that set P*.

8.2.4 A New Algorithm: The Niched Pareto GA

The specifics of the *Niched Pareto GA* are localized to implementation of selection for the genetic algorithm. Selection is used in GAs to determine which individuals of the current population will have their genotypic information passed to the next generation. One of the most widely implemented selection techniques for GAs is tournament selection. In tournament selection a set of individuals is randomly chosen from the current population and the best of this subset is chosen to be represented in the next population. This allows the best individuals to have a high probability of being represented in future generations. Also, by adjusting the size of the tournament we can exert some control over the amount of selection pressure and hence convergence speed. Thus the smallest tournament size of two (binary tournament) exhibits slower convergence than any larger tournament size.

Tournament selection assumes that we want a single answer to the problem. After a certain number of generations the population will converge to a uniform one. Here we implement a form of selection that will allow us to have multiple answers to a problem with multiple attributes. In addition, we want these answers to be representative of the Pareto optimal front. To accomplish this we have altered tournament selection in two ways. First we added *Pareto domination tournaments*. Second, when we have a non-dominant tournament (i.e., a tie), sharing is implemented to determine the winner.

8.2.4.1 Pareto domination tournaments

In order to obtain a Pareto optimal surface, tournament selection must be altered to use multiple attributes. What we have implemented is Pareto domination tournaments. Note that the domination relation establishes a *partial order* on the set of all phenotypes (i.e., attribute vectors)[4]. An individual with an attribute vector higher in the partial order than that of another individual is preferred (i.e., the first individual dominates the second).

The binary relation of domination leads naturally to a binary tournament (tournament size 2) in which two randomly selected individuals are compared. If one dominates the other, it wins. Initially, we used such a small *local domination criterion*, but we soon found that it produced insufficient domination pressure. Our resulting fronts seemed too fuzzy. There were too many dominated individuals in later generations. It seemed that a sample size of two was too small to estimate an individual's true "domination ranking[5]".

Because we wanted more domination pressure, and more control of that pressure, we implemented a sampling scheme as follows. Two candidates for selection are picked at random from the population. A comparison set of individuals is also picked randomly from the population. Each of the candidates are then compared against each individual in the comparison set. If one candidate is dominated by the comparison set, and the other is not, the latter is selected for reproduction. If neither or both are dominated by the comparison set, then we must use sharing to choose a winner, as we explain later.

[4]In the single-attribute case (normal function optimization), this ordering becomes a *total ordering*.

[5]Note that any partial order determines a unique ranking, in which *maximal* individuals are ranked first, then removed. The remaining individuals are reordered, and the maximal individuals of this set are ranked second, and removed, etc. This is the domination ranking scheme suggested by Goldberg (1989).

This simple scheme might be improved in any number of ways, such as by performing some kind of non-domination ranking (Goldberg, 1989) on the comparison set and the candidates and using the ranks as fitnesses. But our scheme was quick to implement and should give roughly the same performance as any more sophisticated procedure. Most important, the sample size t_{dom} (size of comparison set) gives us control over selection pressure, or what we call *domination pressure*. It turns out, empirically, that the performance of the Niched Pareto GA is somewhat sensitive to the amount of domination versus sharing pressure applied.

Below we present pseudocode for Pareto domination tournaments assuming that all of the attributes are to be maximized. S is an array of the N individuals in the current population. *random_pop_index* is an array holding the N indices of S, in a random order:

```
function selection              /* Returns an individual from the current population S */
begin
    shuffle(random_pop_index);              /* Re-randomize random index array */
    candidate_1 = random_pop_index[1];
    candidate_2 = random_pop_index[2];
    candidate_1_dominated = false;
    candidate_2_dominated = false;
    for comparison_set_index = 3 to t_dom + 3 do        /* Select t_dom individuals randomly
from S */
        begin
            comparison_individual = random_pop_index[comparison_set_index];
            if S[comparison_individual] dominates S[candidate_1]
                then candidate_1_dominated = true;
            if S[comparison_individual] dominates S[candidate_2]
                then candidate_2_dominated = true;
        end     /* end for loop */
    if ( candidate_1_dominated AND ¬ candidate_2_dominated )
        then return candidate_2;
    else if ( ¬ candidate_1_dominated AND candidate_2_dominated )
        then return candidate_1;
    else
        do sharing;         /* Insert sharing code from below */

  end .
```

It will not always be the case that there will be a clear winner among any given set of candidates. A problem will arise if both candidates are on the current non-dominated front since neither will be dominated. Even off the front, a small sample size t_{dom} could mean that neither is dominated. And of course both could be dominated. How is a winner then chosen in such a "tie"? If we just randomly allow one of the candidates to be picked, genetic drift will cause the population to converge to a single region of the Pareto front. To prevent this we implement a form of sharing when there is no preference between two individuals. This form of sharing should maintain the genetic diversity along the population fronts and allow the GA to develop a reasonable representation of the Pareto optimal front.

8.2.4.2 Sharing on the Non-dominated Frontier

We apply the established technique of fitness sharing (Goldberg & Richardson, 1987) to maintain a population distributed along the Pareto optimal frontier. Such application raises some interesting questions, however. In what space should distance be measured? What metric should we use? How do we degrade fitness when fitness is a vector, not a scalar? How do we estimate niche size? Below we address each of these questions.

Normal Fitness Sharing We recall from Chapter 4 that when sharing is combined with the more popular tournament selection the niched GA exhibits chaotic behavior (Oei, Goldberg, and Chang, 1991). We saw the oscillations that naive tournament sharing produces in the two-niche case, in Chapter 4. But the wild fluctuations in niche subpopulations induced by the "naive" combination of sharing and tournament selection can be avoided. Oei, Goldberg, and Chang suggest the use of tournament selection with *continuously updated sharing*, in which niche counts are calculated not by using the current population, but rather the partly filled next generation population. This method was used successfully by Goldberg, Deb, and Horn (1992) on a "niching-difficult" problem.

We recall from Chapter 7 that sampling the population can be sufficient to estimate the niche count and so avoid the $O(N^2)$ comparisons needed to calculate *exactly* the m_i. We incorporate both techniques (continuously updated sharing and niche count sampling) in the Niched Pareto GA.

Sharing in Attribute Space In any application of sharing, we can always implement genotypic sharing, since we always have a genotype (the encoding). Typically, however, we know more about the problem, such as the decoded decision variables, which represent a phenotype. Deb's work (1989) indicated that in general, phenotypic sharing was superior to genotypic sharing. Intuitively, we want to perform sharing in a space we "care more about", that is, some phenotypic space.

In a multiobjective problem, we have two candidate phenotypes: the decision variables and the attributes. We might be interested in finding different combinations of decision variables that yield similar attribute vectors, but our primary concern is in finding diverse, but non-dominated, attribute vectors. It therefore makes sense to perform our sharing in *attribute space*. We are interested in maintaining diversity along the phenotypic Pareto optimal front, which exists only in *attribute space*. Although it is possible that genotypic sharing[6], would give us the same diversity along the front, we run across the problem that more than one set of genotypes will give us the same attribute vector. This might lead to large "gaps" in our distribution along the front.

Later, we discuss the possibility of using *nested sharing* to achieve diversity in both the attribute vectors (i.e., on the front) and decision variable vectors of the solutions.

Equivalence Class Sharing Equivalence class sharing is implemented when there is no clear winner among the prospective candidates. In the case of the Niched Pareto GA, when the candidates are either both dominated or both non-dominated, it is likely that they are in the same equivalence class (in the partial order induced by the domination relation). Because we are interested in maintaining diversity along the front, and most of the individuals in these equivalence classes can be labeled "equally" fit, we do not implement any form of fitness degradation according to the niche count. Instead, the "best fit" candidate is determined to be that candidate which has the least number of individuals in its niche and thus the smallest niche count.

This technique might also be called *flat fitness sharing*, since the same effect is produced in normal (single-attribute) fitness sharing when two individuals have the exact same fitness. When $f_i = f_j$ (i.e., the fitness function is flat for individuals i and j), then their shared fitnesses, f_i/m_i and f_j/m_j respectively, are ordered solely by their niche counts m_i and m_j. This situation would arise on a "plateau" of constant fitness, where the radius of the plateau was much larger

[6]Or phenotypic sharing using the decision variables as the phenotype.

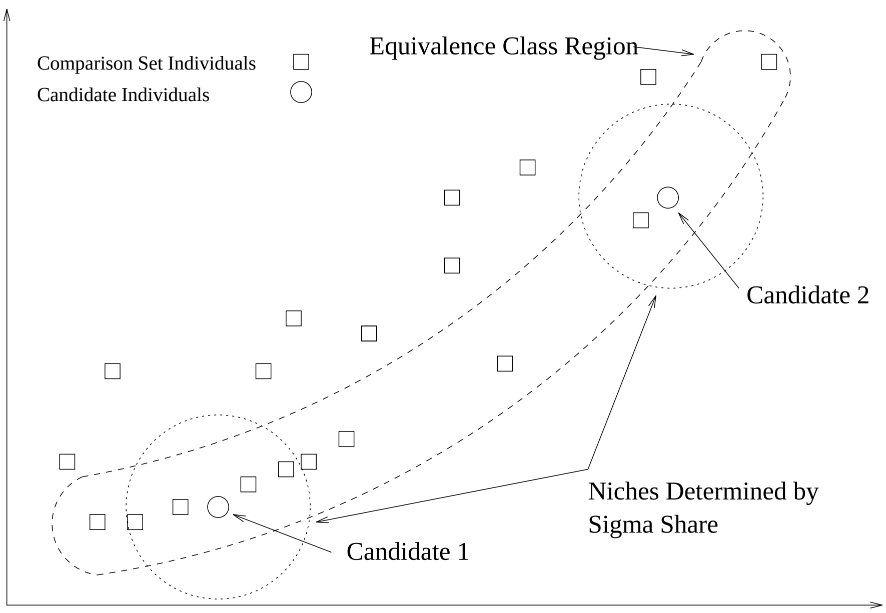

Figure 8.2: Equivalence class sharing.

than σ_{sh}. Although we found no studies that analyzed or reported the behavior of fitness sharing in such regions, we speculate that the population would spread out across the plateau. We should expect to find individuals σ_{sh} units apart from one another, assuming a sufficiently large population to cover the plateau, and assuming crossover and mutation do not remove excessive numbers of individuals from the plateau each generation.

For our multiattribute application, it is hoped that using this form of sharing will encourage individuals to spread out along the front, rather than accumulate in a small region and eventually converge to a single individual.

Figure 8.2 illustrates how this form of sharing should work between two non-dominated individuals. Here we are maximizing along the x-axis and minimizing on the y-axis. In this case the two candidates for selection are not dominated by the comparison set. Thus the two candidates are in the Pareto optimal subset of the *union* of the comparison set and the candidates (the dashed region in Figure 8.2). From a Pareto point of view, neither candidate is preferred. But if we want to maintain useful diversity (i.e., a representative sampling of the Pareto frontier), it is apparent that it would be best to choose the candidate that has the smaller niche count. In this case, that is candidate 2.

Implementation of Equivalence Class Sharing We have to add three lines to our pseudocode for Pareto domination tournament selection given above, to implement equivalence class sharing:

```
function selection
begin
   ⋮
      else if nichecount[candidate_1] > nichecount[candidate_2]
         then return candidate_2;
         else  return candidate_1;
end .
```

Sizing Niches on the Pareto Frontier The performance of fitness sharing is sensitive to the setting of the niche radius σ_{sh} and the population size N. In single-objective, or scalar, fitness sharing, Deb (1989) gives some rough guidelines for setting these parameters. Such guidelines were put to a severe test in Goldberg, Deb, and Horn (1992), and succeeded in predicting niching success and failure. Briefly, Deb suggests fixing the σ_{sh} at some known minimal separation between desired optima, then sizing N for the equilibrium state $f_i/m_i = f_j/m_j$, where all shared fitnesses are equal, and i and j represent two different, desired peaks. The idea is to size N large enough so that the subpopulations m_i are sufficient to withstand the noise of the selection method during "steady state" (Horn, 1993).

Although Deb's sizing does not take into account search, it does provide us with an estimated lower bound on the population size given some idea of the minimal separation of good niches. To adapt Deb's approach to niching on the Pareto front, we also need to have some idea of the separation of our desired "optima". Fortunately, we do have some information in that respect.

Again, we will leave search out of our analysis for now and size only for the steady state. This will give us a lower bound. The desired steady state for the Niched Pareto GA is a population distributed evenly over the entire non-dominated frontier. There are no local optima on the frontier. Instead, we want a more or less uniform distribution to *represent* the possibly continuous frontier. Therefore we can simply take our population size N, divide it into the total surface area of the frontier, A_{pareto}, and extract the required σ_{sh} (or we could derive N for a *desired* separation σ_{sh}):

$$A_{niche}[\sigma_{sh}] \approx \frac{A_{pareto}}{N}.$$

Here we use the term *area* since the Pareto front is a surface of dimension $n-1$ in a search space with volume of dimension n, where n is the number of attributes. Similarly, niche volume is of dimension n, but the intersection of a niche hypersphere with the frontier is an area of dimension $n-1$.

For the A_{niche} term[7], we can try to calculate exactly the $n-1$ dimensional slice of the n dimensional hypersphere that would be on the frontier. Or we can use the approximation $(\sigma_{sh})^{n-1}$, so that $(\sigma_{sh})^{n-1} \approx \frac{A_{pareto}}{N}$. Note that for the two-dimensional case, $\sigma_{sh} = \frac{A_{pareto}}{N}$.

As for the term A_{pareto}, we might have some idea of the size of the Pareto front. Such is the case with the three applications presented later in this section. But if we don't, we still have upper and lower bounds on A_{pareto}, assuming that we have bounds on our attribute values. Assume we know the extreme values (i.e., best and worst values) of our search space along the attribute axes: $\{(a_1^{best}, a_1^{worst}), (a_2^{best}, a_2^{worst}), ..., (a_n^{best}, a_n^{worst})\}$. The minimum length Pareto frontier is the hyperplane connecting these extremes. For example, in the two-dimensional case using Euclidean distance, the minimum Pareto front is simply the hypotenuse of the two attribute axes:

$$min(A_{pareto}) = \sqrt{|a_1^{best} - a_1^{worst}|^2 + |a_2^{best} - a_2^{worst}|^2}.$$

We can also find an upper bound on the area of the Pareto surface. In general, the Pareto surface will be more convoluted than the minimal surface described above. However, the requirement of non-domination limits these convolutions. The surface must be monotonic (i.e., all first-order partial derivatives never change signs). Therefore an asymptotic upper bound on the area of the Pareto surface for the two-dimensional, Euclidean distance case is simply the

[7]This term actually refers to the area of the niche that does not overlap with other niches and so is a hypersphere of diameter σ_{sh}. The *total* area of a niche is a hypersphere of *radius* σ_{sh}.

sum of the attribute ranges:

$$max(A_{pareto}) < |a_1^{best} - a_1^{worst}| + |a_2^{best} - a_2^{worst}|.$$

This is an asymptotic bound since no Pareto surface could actually have this area, although there are allowable surfaces arbitrarily close to it in size. Together, these two bounds should be of some use in sizing σ_{sh} and population N for the Niched Pareto GA when we have no other information or estimate of A_{pareto}.

8.2.4.3 Scaling Attributes

As S. Ranjithan pointed out (1993, personal communication), vastly different scales for the attributes can severely affect the distribution of individuals induced by sharing. Most metrics treat each component of a spatial coordinate vector the same. If we use such metrics for sharing in raw, unscaled attribute space, we might end up comparing apples to oranges. For example, in a two attribute problem we might be trading off cost in the range of \$0 to \$1 million, for reliability ranging from 0 to 1. Our niche sizing method suggested above would be dominated by the largest ranging attribute (cost, in our example). Sizing σ_{sh} for the "larger" attribute means that we will have sparse sampling of the smaller attribute, since σ_{sh}, and hence a single niche, could cover the entire range of the smaller attribute. Conversely, sizing niches for the smaller attribute would lead to convergence of the population to a small subrange of the larger attribute, since many more than N niches could fit into the larger attribute range.

The way to avoid this unintentional niching bias is to scale the attributes so that they all range over the same set of values, say 0 to 1. Note that this is only possible if we have some idea of a maximum and mininum value for each attribute, but this is often the case. Note also that after normalizing all attribute ranges to the range $\{0..1\}$, the above niche sizing method[8] leads to an upper bound of $2/N$ and a lower bound of $\sqrt{2}/N$ for σ_{sh}.

In our applications below, we use the metropolitan, or city-block, metric. This is the Holder metric of degree $p = 1$.

[8]Assuming 2 attributes and Euclidean distance.

8.2.5 Application to Three Problems

The interaction of the niching and dominance parameters (σ_{sh} and Pareto tournament size, respectively) calls for further analysis of the behaviour of this new algorithm. But preliminary empirical results are promising. We find the Pareto frontier for an artificial, two-attribute function over bit strings (number of ones versus number of adjacent "01" or "10" pairs). We also apply our Niched Pareto GA to Schaffer's F2, a simple two-attribute function used to test Schaffer's multicriteria GA algorithm (VEGA) in his 1984 dissertation. The Niched Pareto GA demonstrates performance superior to VEGA in finding and maintaining more of the Pareto set. Finally, we test the algorithm on a real world, open problem whose non-dominated frontier is not known[9]

8.2.5.1 Problem 1: A Simple Test Function

We begin testing our new algorithm by constructing a simple, artificial problem with an easily calculated Pareto optimal front. We call problem 1 "unitation versus pairs" for the two attributes of unitation and complementary adjacent pairs. Both attributes are simple functions of finite length ℓ bit strings. Unitation $Unit[s]$ is simply the number of ones in the string s, thus $Unit[01110010] = 4$. Pairs $Prs[s]$ is the number of pairs of adjacent complementary bits, either 01 or 10. Thus $Prs[01110010] = 4$. The problem is to maximize both attributes. Among strings of a particular unitation, those strings with the greatest "mixing" of ones and zeroes dominate those who "clump" their ones (e.g., 01010 dominates 01100, though both have the same unitation). At unitation $\geq \ell/2$ a tradeoff exists, since adding more ones increases $Unit$ but must decrease Prs.

In Figure 8.3, we plot the feasible region of the two-dimensional attribute space $Unit$ versus Prs for a 12-bit problem ($\ell = 12$). P indicates a point on the Pareto front, while "-" indicates a feasible point that is dominated by some member(s) of the Pareto set. Note that much of the attribute space is infeasible (e.g., a string cannot have any adjacent complementary pairs when it has unitation 12, since it has no zeroes). Note also that a single point in attribute space can

[9] These applications are early tests of the Niched Pareto GA. Many of the insights and guidelines discussed in the previous section were not available at the time of these runs. Consequently, we did not always scale attributes or properly size niches, population, etc., as we have since learned to do.

```
Unitation
12 |    P
11 |        -   P
10 |        -   -   -   P
9  |        -   -   -   -   -   P
8  |        -   -   -   -   -   -   -   P
7  |        -   -   -   -   -   -   -   -   -   P
6  |        -   -   -   -   -   -   -   -   -   -   P
5  |        -   -   -   -   -   -   -   -   -   -
4  |        -   -   -   -   -   -   -   -
3  |        -   -   -   -   -   -
2  |        -   -   -   -
1  |        -   -
0  |    -
----------------------------------------------------------
        0   1   2   3   4   5   6   7   8   9  10  11  12    Pairs
```

Figure 8.3: Problem 1's discrete, two dimensional attribute space, with feasible (-) and Pareto (P) points indicated.

represent as few as one individual string (or zero, for infeasible points!) or many strings (e.g., $(unit, prs) = (1, 2)$).

Although the unitation versus pairs problem is deliberately abstract, one can imagine a realistic analogy. For example, the strings might encode the types of structural elements in some design, with a zero representing an element of high cost and a one representing a less expensive element. Imagine further that adjacent, different elements complement each other and increase the overall strength of the design. Thus our two objectives are to minimize cost (maximize unitation) while maximizing strength (adjacent complementary pairs).

Next we test the Niched Pareto GA on the unitation versus pairs problem. In Figure 8.4, we plot the initial population (generation 0) of 100 randomly generated 12-bit individuals[10]. The numbers plotted in attribute space are the numbers of individuals (strings) with those attribute values. Notice how we obtain a spread of individuals (except at the "less likely" attribute coordinates, such as the all-ones or all-zeroes positions), with no bias towards or away from the Pareto frontier.

[10] Our population size $N = 100$, tournament size $t_{dom} = 10$, niche size $\sigma_{sh} = 2.0$, and we are using a single-point crossover probability $p_c = 0.9$ and mutation rate $p_m = 0.01$.

```
Unitation                             Generation 0
12 |    0
11 |        0   2
10 |        0   0   1   4
9  |        0   0   1   2   1   2
8  |        0   0   0   4   8   1   2   1
7  |        0   0   1   2   5   3   6   1   0   0
6  |        0   0   2   2   6   3   5   6   0   0   0
5  |        0   0   6   2   2   2   1   6   0   0
4  |        0   2   0   1   1   3   0   1
3  |        0   0   0   0   0   1
2  |        0   1   0   0
1  |        0   0
0  |    0
----------------------------------------------------------
        0   1   2   3   4   5   6   7   8   9  10  11  12    Pairs
```

Figure 8.4: Distribution of the randomly generated initial population.

Figure 8.5 shows the population after 100 generations. We can see that the GA has succeeded in finding all but one member of the Pareto set, and appears to be maintaining substantial subpopulations at each such point. Moreover, there are few (none here) dominated individuals in the current population. Although not shown, we have plotted population distributions over many generations, and noticed that the GA does indeed maintain roughly equal size subpopulations at each Pareto point over many generations. Dominated solutions regularly appear, due to crossover and mutation, but are not maintained.

We have observed similar behaviour over many runs of the GA on different initial population distributions (all random, but using different random seeds). We have also successfully tried larger problems ($\ell > 12$), with correspondingly larger population sizes N. Figure 8.6 shows the population distribution in the 200th generation of a 28-bit problem [11], $N = 400$. In this run (Figure 8.6), our niche size σ_{sh} was 5.0 and our tournament size t_{dom} was 10.

To illustrate the importance of sharing, we present in Figure 8.7 the results of running the Niched Pareto GA with sharing turned off (call it the *Pareto GA*). That is, when two competitors in a domination tournament tie (i.e., neither or both are dominated by the comparison set), we

[11] Again, the GA misses one extreme point on the front, namely the strings with the highest number of pairs, and unitation of $\ell/2$. However, we have observed several runs where the GA did maintain an apparently stable subpopulation at this extreme point as well.

```
Unitation                                  Generation 100
12 |  26
11 |       0  11
10 |       0   0   0  14
9  |       0   0   0   0   0  12
8  |       0   0   0   0   0   0   0  16
7  |       0   0   0   0   0   0   0   0   0  21
6  |       0   0   0   0   0   0   0   0   0   0   0
5  |       0   0   0   0   0   0   0   0   0   0
4  |       0   0   0   0   0   0   0   0
3  |       0   0   0   0   0   0
2  |       0   0   0   0
1  |       0   0
0  |   0
----------------------------------------------------------
       0   1   2   3   4   5   6   7   8   9  10  11  12      Pairs
```

Figure 8.5: Stable subpopulations on the Pareto front.

```
Unitation                                          Generation 200
28 |  14
27 |       0  29
26 |       0   0   0  29
25 |       0   0   0   0   0  30
24 |       0   0   0   0   0   0   1  25
23 |       0   0   0   0   0   0   0   0   0  28
22 |       0   0   0   0   0   0   0   0   0   0   0  32
21 |       0   0   0   0   0   0   0   0   0   0   0   0   0  30
20 |       0   0   0   0   0   0   0   0   0   0   0   0   0   1   0  30
19 |       0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  32
18 |       0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1  32
17 |       0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   1  27
16 |       0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   1   2  26
15 |       0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  27
14 |       0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
13 |       0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
12 |       0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
11 |       0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
10 |       0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
9  |       0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
8  |       0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
7  |       0   0   0   0   0   0   0   0   0   0   0   0   0   0
6  |       0   0   0   0   0   0   0   0   0   0   0   0
5  |       0   0   0   0   0   0   0   0   0   0
4  |       0   0   0   0   0   0   0   0
3  |       0   0   0   0   0   0
2  |       0   0   0   0
1  |       0   0
0  |   0
---------------------------------------------------------------------------------------------------------------------------------
       0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28     Pairs
```

Figure 8.6: Success on a much larger problem, $\ell = 28$.

```
Unitation                                        Generation 100
28 |   0
27 |       0   0
26 |       0   0   0   0
25 |       0   0   0   0   0   0
24 |       0   0   0   0   0   0   0   1
23 |       0   0   0   0   0   0   0   1   0  12
22 |       0   0   0   0   0   0   0   0   0   2   1  26
21 |       0   0   0   0   0   0   0   0   0   0   0   2   1  58
20 |       0   0   0   0   0   0   0   0   0   0   0   0   1   2   1  77
19 |       0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   5   1  86
18 |       0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   8   0  67
17 |       0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   5   2  29
16 |       0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   5   0   3
15 |       0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   2
14 |       0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
13 |       0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
12 |       0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
11 |       0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
10 |       0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
9  |       0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
8  |       0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
7  |       0   0   0   0   0   0   0   0   0   0   0   0   0   0
6  |       0   0   0   0   0   0   0   0   0   0   0   0
5  |       0   0   0   0   0   0   0   0   0   0
4  |       0   0   0   0   0   0   0   0
3  |       0   0   0   0   0   0
2  |       0   0   0   0
1  |       0   0
0  |   0
------------------------------------------------------------------------------------------------------------------------------------
       0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28     Pairs
```

Figure 8.7: Premature convergence when sharing is turned off.

randomly choose one as the winner. Figure 8.7 shows that after 100 generations, the population is distributed over only a portion of the Pareto frontier. Not only is the Pareto GA not *maintaining* the whole front, it never even *finds* the other portions of the tradeoff curve in the first place. There is no sharing to spread out the population of the early generations. This search failure leads us to describe the Pareto GA behaviour as *premature convergence*. Eventually, the entire population should converge to a single point on the front, due to drift on the frontier.

Finally, we note that this problem is GA-easy (in that it is easy to find points on the front), but not necessarily easy for the Niched Pareto GA! Since there are actually many more solutions at middle points on the front, and only one or two at each end point of the front, it should be harder to maintain equal size subpopulations at the extreme points.

8.2.5.2 Problem 2: Schaffer's F2

Next we compare our algorithm to Schaffer's VEGA by running it on one of the test functions from Schaffer's dissertation (Schaffer, 1984). This is the simple function F2, with a single

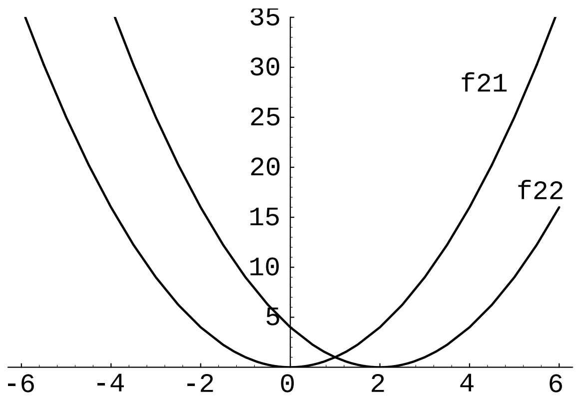

Figure 8.8: Schaffer's function F2, $P = \{x \mid 0 \leq x \leq 2\}$

decision variable, the real-valued x, and two attributes, $f21$ and$f22$ to be minimized:

$$f21(x) = x^2 \qquad\qquad f22(x) = (x-2)^2$$

The decision variable is mapped to a 14-bit string as a binary-coded integer. Thus 00000000000000 $=$ x_{min} and 11111111111111 $= x_{max}$, where $x_{min} = -6.00$ and $x_{max} = 6.00$. We plot $f21$ and $f22$ over this range of x in Figure 8.8. It is clear that the Pareto front is where the tradeoff exists between the two functions. That is, for $0 \leq x \leq 2.00$, one of the functions is decreasing to its best while the other is increasing away from its best.

For our purpose of comparison, we reproduce some of Schaffer's graphs illustrating VEGA's performance on F2. Figure 8.9 shows a plot of the population distribution at generation 0, taken from (Schaffer, 1985), and at generation 3, to be found in either (Schaffer, 1984) or (Schaffer, 1985). The population size is rather small, $N = 30$. Schaffer used the results shown in Figure 8.9 to demonstrate that VEGA's selection scheme was moving the population towards points on the front, and that he was getting a distribution on that front. However, as can be seen on the right of Figure 8.9, the distribution on the front is rather clumpy. We might hope that subsequent generations would show a better spread, as little convergence has taken place in most GAs after only two generations. However, Schaffer gives no further results on F2. This leads us to speculate that VEGA, with no explicit niching mechanism, is likely converging to a single point or small region on the front rather rapidly.

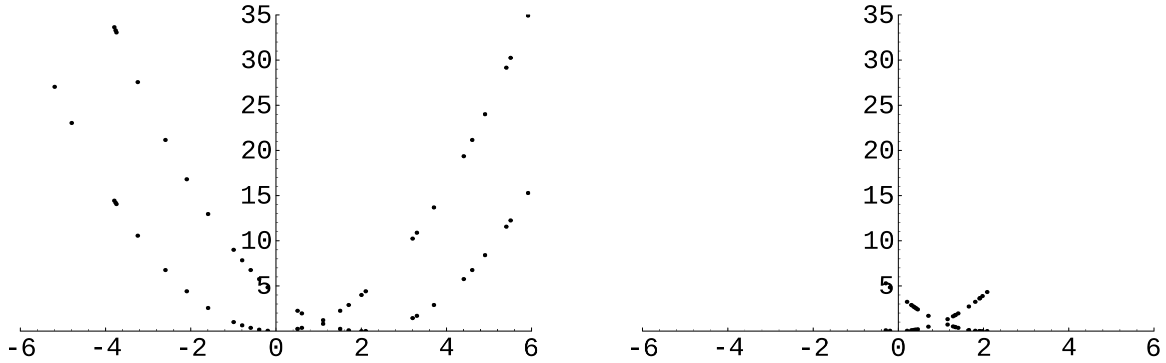

Figure 8.9: VEGA on Schaffer's function F2, generations 0 (left) and 3 (right).

Next we try the Niched Pareto GA on F2. Like Schaffer, we use a small population size $N = 30$. Our niche size $\sigma_{sh} = 0.1$ and tournament size $t_{dom} = 4$. As Figure 8.10 illustrates, the Niched Pareto GA is able to maintain a fairly even spread of solutions along the Pareto front. There are a few dominated individuals in the population (to the right of $x = 2.00$), as in the VEGA run above, but most individuals are on the front. Although our population has several gaps in its distribution on the front, it appears more evenly distributed than generation 3 of the VEGA run[12]. Most importantly, the Niched Pareto GA exhibits stability in this population distribution for many more generations than were indicated for VEGA ($200 > 3$).

We should point out that F2 is clearly an easy problem for the GA. The initial population contains many individuals on the front already. However, this front is much denser than that of problem 1 above, challenging the Niched Pareto GA to maintain N subpopulations of size 1 along the front.

8.2.5.3 Problem 3: Open Problem in Hydrosystems

Neither of these two abstract problems provides a significant challenge to the Niched Pareto GA. Thus we chose a larger, real-world (i.e., unsolved) application for our third test problem: optimal well placement for contaminant monitoring[13]. Although we have not yet had time to

[12] Larger population sizes result in much better coverage (smaller gaps) of the Pareto front. But we use 30 as our population size here for a fair comparison with the figure from Schaffer (1984).

[13] This open problem was developed and explored by Wayland Eheart and his colleagues at the Civil Engineering Department at the University of Illinois at Urbana-Champaign. We are grateful to Dr. Eheart and his students for providing us with this challenging, multiattribute design problem.

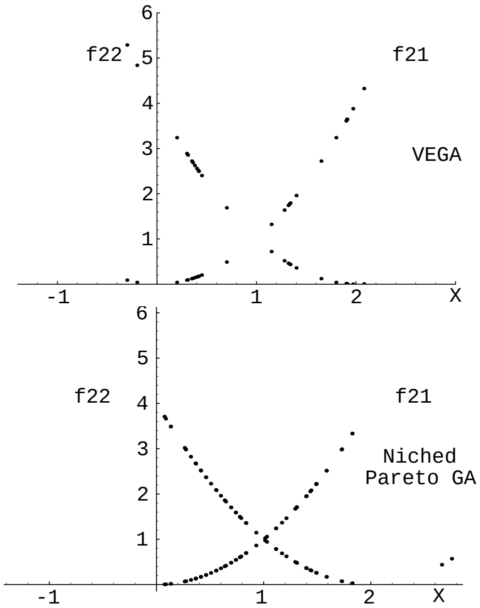

Figure 8.10: VEGA versus the Niched Pareto GA. Top: Expanded view of VEGA's performance on F2's Pareto frontier, generation 3. Bottom: The Niched Pareto GA's performance on F2, generation 200.

experiment with various parameter settings, nor have we compared our solutions to those found via other algorithms, our first few runs indicate that the Niched Pareto GA is working. That is, as the generations go by, the population moves toward the Pareto frontier (i.e., in the right direction) while spreading out in attribute space (i.e., maintaining high quality diversity). Thus the GA is finding and maintaining new and improved tradeoffs among the two attributes.

The problem is to place a set of k out of a possible w wells in order to maximize the number of *detected* leak plumes from a landfill into the surrounding groundwater, and to minimize the volume of cleanup involved. These two objectives conflict. Simply optimizing for the minimum volume of cleanup will give us an answer of $\{0, 0\}$ where we detect no plumes and therefore have no volume of contaminant to clean up. If we maximize the number of detected plumes (which would also be the environmentally correct thing to do) our volume of cleanup would skyrocket (which would displease the businessman). Therefore, there is a tradeoff involved. This is a perfect example where finding the Pareto optimal front is desired. By finding the front, we can give the decision maker a set of possible choices. He or she can then decide which choice is best.

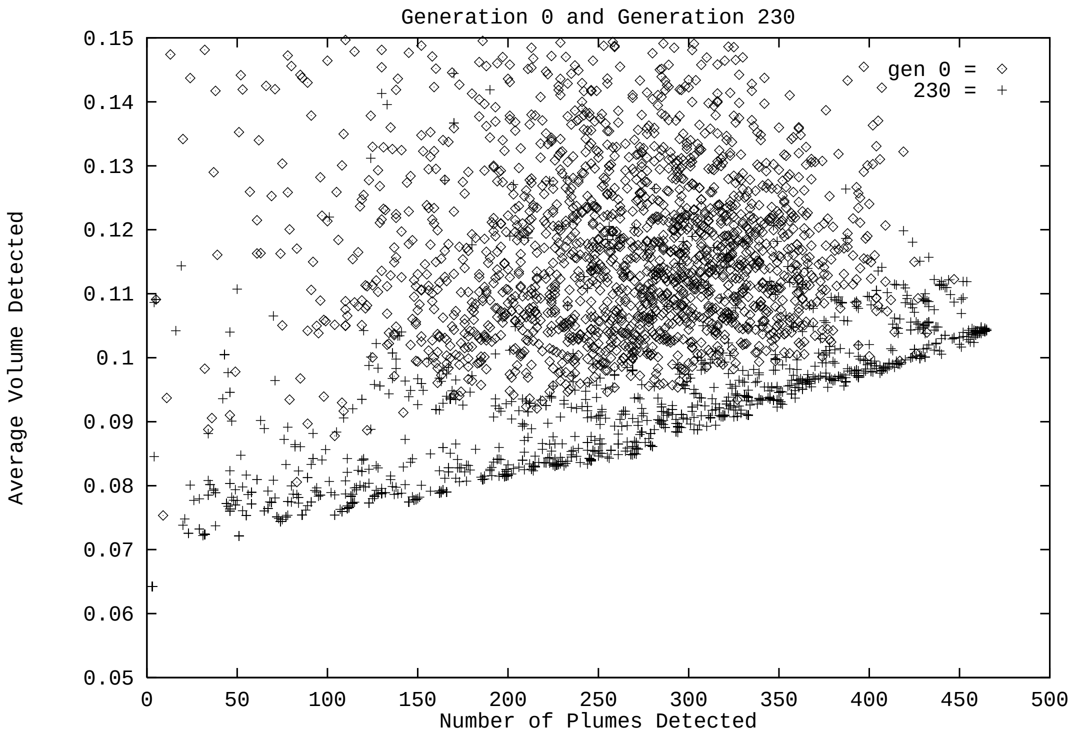

Figure 8.11: Results from a single run for $k = 20$ and $w = 396$. The initial population distribution (generation 0, diamonds) as compared with the distribution at generation 230 (crosses).

It is important to note that this problem is intractable. The search space is of size $\binom{w}{k}$. In our specific example we have $w = 396$ and $k = 20$. The whole search space is then $\binom{396}{20}$ which is 2.269×10^{33}! This makes it impossible to know the actual Pareto optimal front from enumeration and it makes apparent the need for a front end to MODM.

In this application we were given a set of simulations of leaks from a land fill site. Monte-Carlo simulation was used to develop a set of possible leak plumes, the set of wells that detect each plume, and the volume leaked when each well detected the contaminant plume. Using these data, we constructed a fitness function to return the number of plumes and average volume detected by any set of wells given to the function. Several preliminary runs were performed to determine whether or not the Niched Pareto GA could find Pareto fronts for this real world problem.

For our specific problem we applied a Niched Pareto GA to pick a set of $k = 20$ wells from a possible set of $w = 396$ wells. Our values for σ_{sh} and the dominance tournament size t_{dom} were 40 and 40, respectively[14].

Figure 8.11 shows both the initial population (indicated by diamonds) and the 230th generation population (indicated by crosses). Looking at Figure 8.11 one can see that our random initial population (the diamonds) is distributed throughout the search space. It is assumed that the Pareto optimal front will lie below and to the right of these initial values. Figure 8.11 shows that after 230 generations (the crosses) the Niched Pareto GA has found an apparent front that is indeed improved over the initial population. It is promising to see that even after a large number of generations we are maintaining diversity over most of an apparent front. Also, there is definite improvement, from generation 0 to generation 230, as to the location of the front and the decrease in the number of dominated individuals in the population.

These results show that our phenotypic sharing and dominance tournaments are working together. We do not know yet whether this is the actual Pareto optimal front or a sub-optimal front of solutions. To test for this, a more rigorous study of the function with many variations of σ_{sh} and t_{dom} could be conducted. Below, we discuss other ways of testing the front found by the Niched Pareto GA. We have shown, however, that a tradeoff curve better than a random sampling can be developed by the Niched Pareto GA on an open problem.

8.2.6 Discussion

The initial implementation and testing of the Niched Pareto GA leads to some intriguing ideas for immediate and long term extensions. In particular, we consider how to properly adjust the domination pressure, vis-à-vis the niching pressure, how to find multiple solutions at a single point on the front, and the critical issue of search.

8.2.6.1 Setting Selection Pressure

Earlier we showed how to adapt Deb's sizing estimates for σ_{sh} and population size N. Our experimental runs showed that the performance of the Niched Pareto GA was robust with

[14]Because we had not yet developed any guidelines for setting these two parameters at the time of the runs, we simply experimented with combinations over broad ranges. We have since developed some insights into the proper settings for σ_{sh} and dominance tournament size t_{dom}. These are discussed elsewhere in the section.

respect to σ_{sh}, with similar performance over a range of values. But so far we have not yet developed any guidelines for selection pressure (i.e., the size of the domination tournaments t_{dom}). Our experience with the three applications above has given us some empirical insight into the importance of this parameter. Although we found that the algorithm was fairly robust with respect to t_{dom}, we found significantly different behavior once t_{dom} exceeded this large range of values.

In general, increasing the size of the dominance tournament seemed to increase pressure towards the front, as we would expect. The higher domination pressure caused the population to move more rapidly in the direction of the front (that is, improvement in both attributes). But it also seemed to curtail exploration. Too high a value for the dominance tournament size (e.g., $t_{dom} = 200$ for the hydrosystems application above), and the population "prematurely converged" to only a portion of the front, usually near the middle. The effect was the same as when sharing is turned off, as in Figure 8.7. Conversely, with too small a dominance pressure (e.g., $t_{dom} = 2$), the final population consisted of many dominated solutions. That is, with too little selection pressure with respect to sharing, the population failed to achieve a tight distribution on the front, and was distributed well behind, as well as on, the front. For an example of this behavior, compare Figure 8.12, in which $t_{dom} = 2$, to Figure 8.6.

Just as tournament size t_{size} is critical to selection pressure and premature convergence in a regular GA with tournament selection, so t_{dom} directly effects the convergence of the Niched Pareto GA. Therefore, this value must be chosen carefully for each problem that uses the Niched Pareto GA. From our limited experiments, we have order of magnitude guidelines:

- When $t_{dom} \approx 1\%$ of N, we saw too many dominated solutions.
- When $t_{dom} \approx 10\%$ of N, we witnessed a tight and complete distribution.
- When $t_{dom} \gg 20\%$ of N, the algorithm prematurely converged.

Until analytical guidelines are developed, we will continue to explore the algorithm's behavior over a range of t_{dom} on each new problem.

8.2.6.2 Genotypic versus Phenotypic Diversity: Nested Sharing

The Niched Pareto GA implements phenotypic sharing, where the attribute vector is the phenotype. We are first and foremost interested in maintaining diversity along the tradeoff surface.

```
Unitation                                      Generation 100
28 |   0
27 |       0   3
26 |       0   1   3   3
25 |       0   0   2   3   5   2
24 |       0   1   3   4   2   3   2   5
23 |       0   0   1   0   4   2   1   4   4   3
22 |       0   0   0   4   2   2   4   1   6   3   4   2
21 |       0   0   1   1   0   1   2   5   4   6   5   3   1
20 |       0   0   1   0   0   2   0   3   1   4   5   4   5   5   3
19 |       0   0   0   0   0   0   1   1   4   3   2   5   4   6   6   4   3   4
18 |       0   0   0   0   0   0   0   2   0   2   4   2   4   4   5   6   3   5   2   0
17 |       0   0   0   0   0   1   1   1   1   1   0   1   4   4   5   5   8   4   6   3   4   1
16 |       0   0   0   0   0   0   0   1   0   1   0   3   2   5   5   1   3   4   3   4   5   5   3   1
15 |       0   0   0   0   0   0   0   0   0   1   0   0   0   1   3   1   4   2   6   1   4   2   4   0   1   0
14 |       0   0   0   0   0   0   0   0   0   0   0   0   0   2   0   1   3   1   3   3   0   4   4   0   1   0   0
13 |       0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   2   2   1   0   2   0   1   1   1   1   0
12 |       0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   1   1   1   0   0
11 |       0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   1   0
10 |       0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0
9  |       0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
8  |       0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
7  |       0   0   0   0   0   0   0   0   0   0   0   0   0   0
6  |       0   0   0   0   0   0   0   0   0   0   0   0
5  |       0   0   0   0   0   0   0   0   0   0
4  |       0   0   0   0   0   0   0   0
3  |       0   0   0   0   0   0
2  |       0   0   0   0
1  |       0   0
0  |   0
-----------------------------------------------------------------------------------------------------------------------------------
       0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28    Pairs
```

Figure 8.12: Population spread on problem 1 when domination pressure is too low ($t_{dom} = 2$).

But if we achieve such diversity, we might then also be interested in finding diverse solutions at given tradeoff points. In other words, we might want to know about the several different ways of achieving the same set of attribute values. We would then be spreading out our solutions in both attribute space and the space of decision variables[15].

Simultaneously searching for two kinds of diversity is a goal that transcends the multiattribute focus of this section. We present some possible techniques which, if successful, could be applied to sharing in regular, scalar GAs as well:

- **Pointwise expansion -** We pick a point on the Pareto front (i.e., in attribute space) and search for diversity in decision variable space. We first run the Niched Pareto GA to find a front. We then run a niched GA in which the objective fitness function is distance, measured in attribute space, to the chosen point on the front. This distance is to be minimized. Sharing is performed in decision variable space. This method only explores

[15]The decision variables are often considered to be the genotype, even though there might be an extra step involved in decoding from a binary string, for example, to real-valued decision variables. For the purposes of this discussion, we do not distinguish between the genotype of the low level encoding and the phenotype of the decoded decision variables. Both are much more similar to each other than to the "high level" phenotype of the attribute vector.

one point, or small region, of the front at a time, but brings the entire population to bear on that search.

- **Threshold sharing -** If two individuals are in the same equivalence class, and we have calculated their niche counts (in attribute space) and their niche counts are within some fixed δ of each other, then we perform the second level of sharing (in decision variable/genotype space). We do this by simply calculating and comparing their *genotypic* niche counts. This approach introduces a new parameter, δ.

- **Sequential sharing -** Generate a new population as before (with attribute sharing). From this intermediate population, generate the next population by performing sharing in decision variable space. In other words, perform selection using the Niched Pareto GA with sharing in attribute space. Then perform selection using the Niched Pareto GA with sharing in decision variable space. Finally, perform crossover and mutation.

- **Simultaneous sharing, multiplicative -** (This algorithm was suggested by Georges Harik.) Multiply the niche count in attribute space by the niche count in decision variable space and use this product as the overall niche count for comparing two individuals in the same equivalence class. It might be necessary to raise the attribute niche count to some power to reflect the relative importance of maintaining diversity along the Pareto front first and foremost.

- **Simultaneous sharing, additive -** Similar to the algorithm above, but add the two different niche counts instead of multiplying them. This is similar to combining the attribute and decision variable spaces into a single space (which is yet another approach to multi-level or nested sharing!). Or we could simply change the metric to encompass both spaces. But in all cases, we would likely have to use weights, or else scale attributes and decision variables, to emphasize the primary importance of attribute space diversity.

All of these ideas could be adapted for multi-level sharing in a scalar search space, and are thus of general interest. In the case of the Niched Pareto GA, we should first make sure we can find and maintain the Pareto front before we expand our search in other dimensions. However, we are ultimately very interested in decision variable diversity because it would add robustness to our solutions[16].

8.2.6.3 The Issue of Search

Throughout this section we have concentrated on the usefulness of the final solutions, P. We have not discussed the critical issue of search[17]. But in this dissertation we have presented some empirical and theoretical evidence that the diversity maintained by niching does improve search. In this multiobjective application, does "Pareto-diversity" (i.e., diversity within P using distance in objective space as a metric) really help search?

Both the aggregative and Pareto approaches are searching the space of decision variable settings $\vec{x}$. But while the MAUA approach explores the behavior of some aggregative utility function $U[\vec{x}]$ over that space, the Pareto approach looks at how the currently non-dominated set grows or shrinks over $\vec{x}$. Which type of search is easier? Which exploits the best information? The answers might be problem dependent. But we have some intuitions.

Our intuition in the case of Pareto optimization is that the diversity along the currently non-dominated frontier actually helps the search for new and improved tradeoffs, thus extending the frontier. Of course, spreading the population out all along the front limits the resources available to explore a particular region of the front. However, individuals from very different parts of the front might be crossed to produce offspring that dominate a portion of the front lying between their parents. That is, information from very different types of tradeoffs could be combined to yield other kinds of good (non-dominated) tradeoffs. Indeed, we see some evidence for this in application problem 3. Because equivalence class sharing cannot be expected to maintain more than one copy of an individual (i.e., niche counts are approximately 1 for all niches at steady state), and because we used high crossover rates (typically 0.7-0.9), the maintenance of the front over hundreds of generations was largely due to the constant generation and regeneration of individuals on the front from the crossover of two different parents. In other words, most

[16]If for some reason one of the given solutions becomes infeasible at a later date, we would have an alternative solution with the same performance.

[17]Almost all analysis of niching methods, has focused exclusively on the steady state behavior of niching, rather than on the search for that steady state (e.g., Deb, 1989; Horn, 1993; Mahfoud, 1993). This exclusive focus can be justified, however, since we must be sure first that there exists a steady state before we worry about how a niched GA might reach it.

crosses of parents on or near the front yielded offspring also on or near the front. This behavior is evidence that *Pareto diversity* helps *Pareto search*.

8.2.6.4 Conclusions

It is not clear what is the best way to handle multiple objectives with a genetic algorithm. Indeed, different approaches might be called for by different situations (e.g., small Pareto set, group decision making). But the Pareto approach is independent of the decision maker. The Pareto optimal set P depends only on the objective function $A[\vec{x}]$, which maps decision variable vectors to attribute vectors. No matter what multiattribute utility function $U[\vec{a}[\vec{x}]]$ a decision maker brings to the problem, the global optimum of U will always be found in P. And if P is small enough or otherwise easily visualized, the decision maker might be able to choose the best alternative directly, without an assessment of U. We introduced the Niched Pareto GA as an algorithm for finding and maintaining solutions all along the Pareto front P. But P might be very large and/or convoluted, and therefore impossible for the Niched Pareto GA to maintain or even find.

We have introduced several techniques that are applicable outside the set of multiobjective problems:

- Partial order optimization
- Equivalence class sharing
- Nested sharing

The domination tournament does not rely strictly on a domination relation, but rather on any antisymmetric, transitive relation. Thus it can be used to optimize any partially ordered space. Equivalence class sharing (or flat fitness sharing) is useful not only on the Pareto optimal frontier, but in any equivalence class induced by a partial order. Even in a totally ordered set (i.e., scalar function optimization), such sharing is useful in large niches, such as high plateaus, to obtain a representative sampling of the entire highly-fit region. Nested, or multilevel, sharing has only been discussed, not demonstrated, in this section. However, if successful, it would allow us to find diversity in several spaces simultaneously. Nested sharing represents a generalization of fitness sharing to multiple spaces akin to the generalization of the scalar GA to multiple objectives.

8.3 Packing/Covering/Layout Problems

This section is highly speculative, but the problem area of packing, covering, and layout problems is so general that the initial results presented below deserve some attention. We consider for now only the very abstract problem of trying to fit as many copies of some identical n-dimensional shape into an n-dimensional volume or *substrate*. Possible real-world instances of this problem type include the stamping of identical metal pieces from a flat (two-dimensional) sheet of metal, with the goal of maximizing the number of such pieces from a given sheet. Another possible example is maximizing the number of planks or blocks of wood cut from a single three-dimensional length of tree trunk. The common characteristics of such problems include the identical shape of the pieces needed (although this constraint could be relaxed in our algorithm below), the arbitrary shape of the piece, and the arbitrary shape of the substrate. It is the arbitrariness of these shapes that make the layout or packing problem so difficult.

Sharing seems appropriate here, as individuals can compete for the same portion of substrate (i.e., pieces can "overlap"). Overlap must be severely punished, as any feasible solution cannot allocate the same portion of substrate to two different pieces (without violating the integrity of a piece!). On the other hand, we want to entertain multiple competing solutions during the search process, all in the same population. Thus the utility of a decreasing level of tolerance for increasing overlap (i.e., sharing).

Below we describe a simple first step in testing the applicability of sharing to these types of problems. We use a one-dimensional version of the problem, so that we do not have to worry about (i.e., encode) the *orientation* of our pieces on the substrate. The substrate is simply a length of material, while the pieces are also lengths of material (only smaller than the substrate!).

Even in one dimension, niching generates surprisingly complex associations of cooperating and competing individuals, and groups of individuals, all arising under the the application of niching and selection alone. As we shall see, these phenomena strongly resemble the level-three and level-four types of cooperation conjectured in Chapter 3.

8.4 An Artificial One-dimensional Packing Problem

Let us assume a one-dimensional fitness function f:

$$f(x) = \begin{cases} 1 & \text{if } 40 \leq x \leq 200) \\ 0 & \text{otherwise} \end{cases}$$

where x is an integer and $0 \leq x \leq 255$. Thus f is simply a discrete "hat" function: zero everywhere except on a plateau of high fitness. We assume an encoding of the single decision variable x as a finite length chromosome, such as a fixed length bit string interpreted as a binary coded integer (e.g., $\ell = 8$ bits gives all $2^8 = 256$ integers from 0 to 255). Thus x can be seen as the coordinate of the center of each (identical) piece, with the length of the piece being $2\sigma_{sh}$ (that is, σ_{sh} is the radius of each piece).

All chromosomes that decode to an integer from 40 to 200 are global optima. Thus there are 161 global optima, many of which would be present in the initial, random population. The function can be seen as representing the substrate. The substrate would stretch from location $40 - \sigma_{sh}$ to $200 + \sigma_{sh}$, with a total length then of $160 + 2\sigma_{sh}$. Thus any piece with center coordinate $x < 40$ or $200 < x$ would have fitness 0 because part of the piece would extend beyond the edges of the substrate.

Were we to run a simple GA on f, we would expect global optima to quickly dominate the population. Genetic drift would then slowly lead to convergence of one of the (equally good) global optima. What would happen if we applied a GA with fitness sharing? Because it brings out strongly certain aspects of sharing, we give this kind of niching application a name: we call it *flat fitness sharing*, or sharing on a flat fitness function. As we pointed out in the previous section, such sharing occurs within equivalence classes on multiobjective problems under the Niched Pareto GA.

8.4.1 Fitness Sharing Results

Let us apply a niched GA, using fitness sharing, and fixing the population size $N = 2560$. Thus in the initial random population we *expect* to have ten copies of each of the 256 different possible individuals, giving us a good chance of representing the entire search space in generation 0. We fix $\sigma_{sh} = 20$, and perform sharing (that is, measure distance) in the domain of f (that is, the decoded integer x).

Figure 8.13 shows eight snapshots of population distributions over the domain of f. Each small graph plots the number of copies of each of the 256 different possible individuals in a particular generation. In the upper left plot of Figure 8.13, we see the random initial population distribution, nearly uniform as expected. By generation 5 (upper right plot), the non-global optima have been eliminated from the population. Through generation 50, we see an "edge effect", where the niches at the very edges of the "hat" (plateau) accumulate population share more quickly than any other niches. Meantime "internal niches" are in turn affected by the growth of the "edge niches". Those niches exactly $\sigma_{sh} = 20$ units distant from the edges grow almost as quickly as the edge niches. Similarly, niches 20 units in from these niches grow almost as quickly, and so on. By generation 100, the only niches remaining in the population are those that are exactly a multiple of σ_{sh} distant from both edges. Thus the steady-state population consists of non-overlapping (non-competing) niches. Together, these nine niches might be said to collectively (cooperatively) "represent" the globally optimal plateau, by uniformly sampling the surface.

Clearly what is happening is that the "edge effect" propagates in toward the middle of the plateau. There is no preference among any of the globals. However, the niches at the edges of the plateau benefit from reduced competition. Since all neighbors to one side have objective fitness of zero, the edge niches find themselves "neighborless" on one side within at most five generations. Having to share less (that is, having a lower niche count) than other niches centered on "internal" global optima, the edges quickly recruit members. Their increased niche populations then degrade the shared fitnesses of neighboring niches less than σ_{sh} distant from the edges. Niches that are exactly σ_{sh} distant from the edges are not degraded by the edge populations, and instead benefit from the suppression of niches between them and the nearest edge. A chain of such alternating competition and cooperation is eventually established across the plateau, until only the nine non-overlapping niches, anchored at the edges by the edge niches, are all that survive.

The implicit cooperation at work in this example can be visualized more dramatically. In Figure 8.14, attention is called to *groups* of cooperating (non-competing) niches by drawing a line to connect the plotted points of niches that are separated by *exactly* σ_{sh}. Thus there are 19

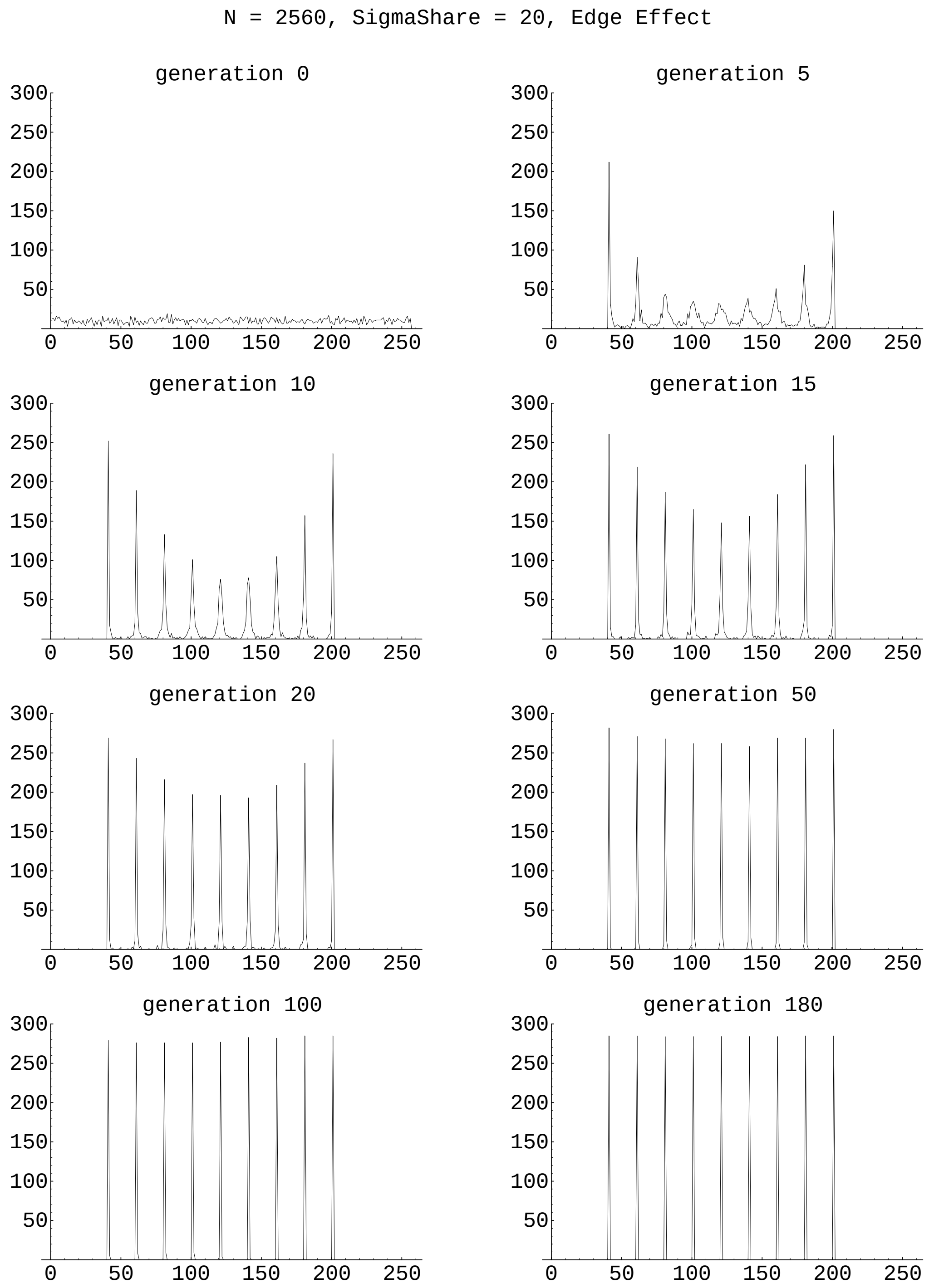

Figure 8.13: Niching (fitness sharing) on a simple "hat" function can lead to *edge effect* cooperation.

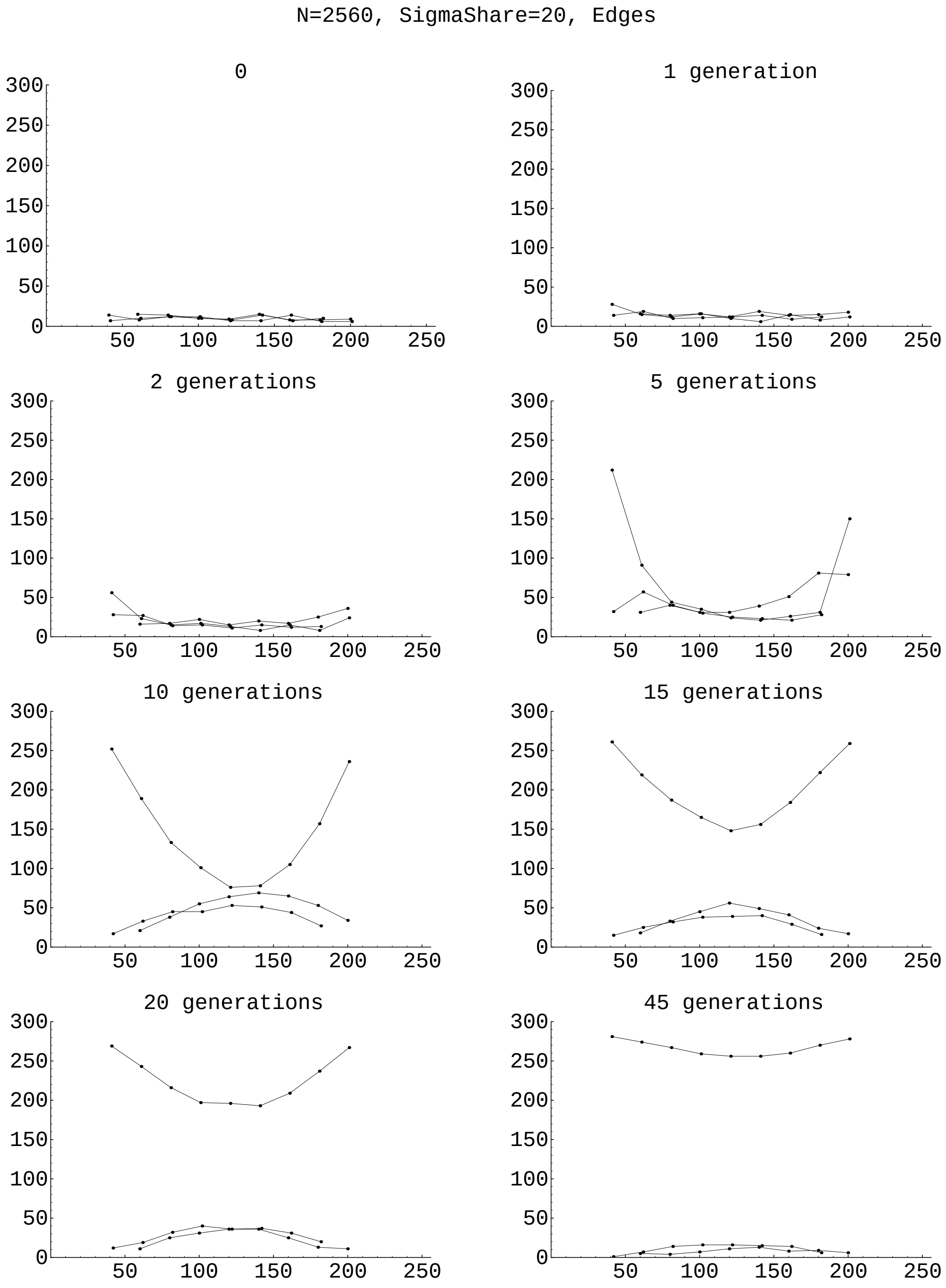

Figure 8.14: Connecting "cooperative" niches by lines demonstrates dramatically that cooperation among some individuals means competition among *groups* of cooperating individuals.

possible "lines" of eight niches, and one line of nine niches (the group containing the two edge niches). A few of the possible 20 cooperative groups are depicted in Figure 8.14. We can see that the one "best" group, containing the edge niches, wins out over the other competing groups, being "pulled up" by the ends. This figure illustrates how simple competition (for limited resources, namely "function real estate") among individuals can lead to complex cooperation within a *group* of individuals, which then leads to competition *between* groups of cooperating individuals.

8.4.2 Discussion: Generalizing the Approach

Sharing shows some promise of solving difficult layout problems by utilizing the level-four type of indirect cooperation conjectured in the layered cooperation-competition model of Chapter 3 section 3.2, which gives rise to the "long-distance" cooperation of the CHAIN suggested in Chapter 7, section 7.2.2. Thus this function is an excellent aid in future analysis of niching phenomena. Being able to model, predict, and ultimately control a complex niching scenario such as this is a good challenge for a complete model of niching. Future work on multi-niche ($k > 2$ niches) should try to explain and predict the dynamics, or at the least the outcome, of Figure 8.14.

As for more practical uses of this type of function for niching, there is no apparent reason why this method could not be extended to handle arbitrarily complex shapes encountered in two, three, and higher dimensions. Also, since time is a one-dimensional, continuous, finite resource, some scheduling problems can be seen as a "packing" of fixed length time intervals (e.g., jobs) onto the fixed length "substrate" representing the available time (e.g., on a particular machine). Thus flat fitness niching might be used for "temporal niching" to solve scheduling problems.

For all such flat fitness functions, however, resource sharing might be more appropriate than fitness sharing, since the substrate clearly is a finite, shareable resource that could be discretized to any degree. But fitness sharing might work just as well, since it could be seen as a continuous approximation to the discrete resource sharing method. Again, the flat function is an excellent candidate for helping us theoretically understand, as well as practically apply, sharing-based niching.

8.5 Summary

We have summarized in this chapter three very different applications of sharing, from search in a standard optimization problem, to multiobjective decision making, and finally a speculative outline of an application to layout, packing, and possibly scheduling problems. In addition, through this dissertation we have shown the effectiveness of resource sharing for classification problems. These represent only a sample of possible applications of sharing. The efficient (perhaps optimal) "covering" of diverse finite resources can be useful in many situations, from finding the entire tradeoff surface in a multi-criteria decision problem to dividing up antigens to antibodies, or to the decomposition of a complex task into smaller subtasks to be handled by cooperative, specialist agents.

Chapter 9

Conclusions

It is time to summarize the major results of this dissertation, to point out the specific contributions made herein, and finally to call attention to the logical extensions of this analysis for future work.

9.1 Summary

The major thrust of this dissertation was the thorough exploration of a particular, well-known niching method: sharing. We concentrated on the performance of such niched genetic algorithms in the case of niche overlap, a situation not covered in most previous analyses. By examining the change in expected performance of the algorithm as niche overlap increased, we were able to discern a boundary between cooperative niching and competitive selection.

The first part of the analysis defined sharing in such a way as to abstract the mechanism from specific implementations such as fitness sharing (for optimization) and resource sharing (for classification or immune system models). This early analysis, contained in the end of Chapter 2 but mostly in Chapter 3, pointed out the necessary differences between fitness and resource sharing (fitness sharing cannot take advantage of explicit resources in the environment) as well as the basic similarities (e.g., both methods reduce to *perfect sharing* when there is no niche overlap). We also placed sharing in the larger contexts of cooperation-competition hierarchies, models of "context dependent function optimization," and the work in theoretical population biology dealing with species use of finite resources. We found that even a simple GA can be considered an implementation of sharing, in which the finite resource of N population slots is shared among competitors.

Because of the dynamic complexities resulting from niche overlap, we focused most of the study on the two-niche case. But within that boundary we examined both types of sharing, fitness and resource, always looking first to the case of no overlap (perfect sharing) to provide a benchmark. We found that all three cases of sharing indicated a *niching equilibrium* population distribution. This distribution contains diversity, unlike the equilibrium distribution of a simple GA. In Chapter 4 we calculated these equilibrium points, and then analyzed the ability of GA selection to maintain equilibrium. We found that under perfect sharing and even under niche overlap, sharing is expected to maintain multiple niches for very long periods of time, with expected niche loss times growing exponentially in population size N. We examined the expected distribution around equilibrium, and we found that the "tightness" of this distribution, as well as the expected niche loss times, degrades with increasing niche overlap and/or with increasing disparity in relative niche fitness. Thus the performance of sharing degrades gracefully with increasing competition.

In Chapter 5 we looked at the times to reach equilibrium, and found these to be very fast, growing only logarithmically in N. So convergence to equilibrium in the niched GA behaves as takeover does in the simple GA, while niche maintenance times are so long as to be effectively infinite, for larger N. But again, as with niche maintenance times, niche convergence times degrade (i.e., lengthen, in this case) as niche overlap and fitness difference increase. By setting niche convergence times and niche maintenance times approximately equal, we were able to define bounding curves on a control map for niching, curves that bound the regions of cooperative niching behavior and competitive niching selection. Thus our foray into the realm of overlapped niches revealed that sharing is a niching method tolerant of overlap, but enough overlap and/or fitness disparity and sharing gives way to selection, and competition dominates. Being able to predict which combinations of overlap and fitness difference are cooperative and which lead to competition, should allow us to use competitive evolution to evolve the kinds of cooperative behavior we seek.

In Chapter 6 we left the niching control map and associated models while we considered the meaning of niching equilibrium. We found that simply calculating niching equilibrium becomes much more computationally intensive when considering more than two overlapping niches, especially for resource sharing. Yet the niched GA converges quickly to such an equilibrium, suggesting that the niched GA might be performing useful and intense computation in coming to equilibrium. We began a number of other suggestive analyses, such as a simple analysis of schemata maintenance at equilibrium that suggested sharing could help GA search by preventing premature loss of building blocks. We also found that under perfect sharing, the equilibrium population distribution can be considered optimal under a simple, multiplicative figure of merit based on the entire population. Although our results on the significance of niching equilibrium are preliminary and speculative, these early results should open the door to further analysis of the utility and applicability of this unique population distribution.

In Chapter 7 we demonstrated the robustness and flexibility of the basic sharing mechanism by answering some common, but overly simplistic criticisms of sharing implementations. Some of these criticisms we answered directly, such as that pertaining to the computational cost of sharing. We demonstrated that it need not be expensive, and that any added computation can be viewed as the price to pay for the complex and potentially beneficial agent interactions being simulated by sharing. We also demonstrated extensions to the simple well-known sharing methods, extensions such as powersharing and elitist sharing. These simple additions can increase the robustness of sharing's performance. Although sharing does have its limits and will fail beyond those limits, it seems unfair to call sharing "brittle".

Finally in Chapter 8 we applied sharing methods to three very different types of problems, including search and multimodal optimization, multiobjective optimization, and layout/packing problems. Sharing has been used in many published applications. We only tried to demonstrate here the breadth of applicability of this basic mechanism, and to show some examples of techniques for incorporating sharing in full GA implementations.

9.2 Contributions

Sharing seems not only a natural thing to do, but also a logical choice for augmenting the basic mechanisms of the simple GA. Sharing appears to work well with the traditional, established operators: selection, mutation, and recombination. Some might argue that yes, *niching* is a logical extension of GAs, but not necessarily with sharing as the niching method. But sharing is closely patterned on nature, and is extremely simple in concept and execution, yet complex in what it can produce, as is the case with the other three fundamental operators of GAs.

More specifically, the case for sharing as a standard operator of GAs rests on both conceptual and practical contributions from this dissertation.

9.2.1 Conceptual Contributions

To those studying, modeling, and designing GAs, we suggest the following conclusions of this dissertation to our understanding of niching:

1. Sharing is ubiquitous: much of what we want to do with GAs can be achieved via sharing implementations,
2. Sharing is fast, stable, and resilient,
3. Sharing plus selection alone is performing complex, useful computation,
4. Sharing supports the evolution of cooperation, and
5. Sharing is open to being modeled, predicted, and controlled.

Below we discuss how each of these conclusions follows from earlier chapters.

9.2.1.1 Sharing is Ubiquitous

Perhaps "ubiquitous" is too strong a word, but we did find that sharing can be used in vastly different contexts: to maintain the fixed population size in simple GAs, to find multiple alternative solutions, to solve multiobjective problems, to improve search, to cover a space or pack a volume, and to learn concepts. In chapter 3 we found that a simple GA, with its fixed-size population, can be emulated by a single-resource GA with sharing and a varying population size. We could argue that the designers of the simple GA were emulating the more natural single-resource GA, whether this emulation was conscious or not.

The implicit use of sharing to induce niching is present in many implementations of classifier systems, in which credit for proper classification of examples is in some way split among competing rules. Sharing also appears in immune system models, in which antibodies stochastically compete for antigens. And sharing can be used to maintain building blocks in the population until they can be recombined into larger building blocks, thereby potentially improving all simple GA search applications.

We found that fitness sharing and resource sharing are closely related, with perfect sharing as the overlap between them. They can be seen as two instances of the general, ubiquitous method of sharing.

9.2.1.2 Sharing is Fast, Stable, and Resilient

The niching induced by sharing behaves in many ways like the simple GA within each of the subpopulations in each niche. This means fast convergence to a stable, long-term equilibrium population. Niching convergence time is low, growing only logarithmically in population size N. Niching equilibrium is stable, with maintenance times that grow exponentially in N, a relatively tight distribution around the unique equilibrium, and a fast restoration of equilibrium in response to any perturbation from it.

Sharing is also "computationally reasonable", and perhaps even "efficient." Sharing can be implemented using $O(N)$ chromosome comparisons per generation to approximate the actual niche counts, with no additional objective function evaluations at all (beyond those required by the simple GA). And with convergence to equilibrium happening on a logarithmic time scale, sharing plus selection is really only $O(N \ln N)$ both in terms of function evaluations and in terms of chromosome comparisons. (Of course search can take longer, as we have not considered *mixing times* here.)

Sharing equilibrium is robust. We have shown that sharing degrades gracefully with increasing niche overlap, increasing fitness disparity, and decreasing population size. Additionally, we have shown that sharing can be tuned, enhanced, or augmented to handle extremely difficult niching situations in a number of ways (e.g., powersharing, rootsharing, elitist sharing). Contrary to some recent criticisms, sharing can indeed handle non-uniformly-spaced niches and massive multimodality.

9.2.1.3 Sharing Performs Intensive Computation

Consider that almost all of the analysis, and much of the applications, of sharing in this dissertation did not utilize recombination, nor mutation for that matter. Sharing and selection alone were capable of generating complex and intriguing dynamics. It seems that sharing (with selection) accomplishes a difficult task (at least if there is niche overlap it is difficult): finding a particular population distribution (i.e., sharing equilibrium) from among the $O(N^N)$ different possible population distributions. When there is no overlap among niches, we can model sharing's search over population distributions as the optimization of the product of shared fitnesses. But with zero niche overlap, convergence is instantaneous (one generation) and therefore "easy". With some overlap, convergence takes some time, and it is not yet clear what kind of optimization is performed by sharing. But we have found that a population distribution can represent a distributed solution to a problem. In the case of classifier systems, the equilibrium population can represent a disjunctive normal form concept representation. In the multiobjective case, the equilibrium can be a sampling of a tradeoff surface. We have also found that even calculating directly the appropriate population distribution for niching equilibrium can involve solving a system of degree-k polynomials, where k is the number of niches involved. It seems we have only begun to appreciate and apply the natural algorithm of sharing-induced niching.

9.2.1.4 Sharing Evolves Real Cooperation

We showed that sharing supports a "level 0" type of implicit cooperation, in which cooperation is simply the lack of competition. We also found surprisingly complex types of "higher level" cooperative relationships, such as the CHAINs at levels three and four of our cooperative-competitive model in chapter 3, predicted in chapter 7, and clearly seen at work in the layout application in chapter 8.

9.2.1.5 Sharing is Amenable to Modeling, Prediction, and Control

Although we probably found more to perplex us about sharing than we found we could "nail down", still we saw that sharing is not beyond analysis, and that simple models can be adequate predictors. For example, our Markov chain analyses resulted in relatively simple closed-form expressions for niche maintenance times. Not only can we predict niche maintenance times, but we can now understand niche maintenance in simple terms: "sharing maintains niches for exponentially long times (in N)". Putting niche maintenance and niche convergence results together, we can now support the view that sharing induces a kind of localized selection within a niche, in which convergence occurs quickly (logarithmic time, in N). We can also rigorously support simple, powerful statements like "sharing induces a unique equilibrium which will be reached quickly and maintained for long periods of time," and "niching degrades with increasing overlap and/or fitness difference." We can back up such qualitative statements quantitatively. For example, we can derive bounds on niche overlap and fitness ratio for expected niche maintenance at a given population size.

Sharing is only as useful as our ability to control it. Although we are far from a complete and general model for niching, this dissertation documents some promising early results:

- For Two-Niches:
  - There exists a unique niching equilibrium.
  - The time to converge to niching equilibrium grows logarithmically in N.
  - The time until loss of niching equilibrium grows exponentially in N.
  - Niching equilibrium degrades with increasing overlap, increasing fitness disparity, and decreasing N.
  - On maps of possible two-niche scenarios there exist three regions: clear niching success, clear niching failure, and the gray area of possible success or failure. Our boundaries delineating these regions represent a significant improvement over previous theoretical bounds.
- For $k > 2$ Niches (more speculative):
  - Mutually overlapped clusters of niches can lead to complex behavior of GA selection (specifically, non-monotonic convergence to equilibrium);
  - Convergence to equilibrium occurs independently in clusters, such that modeling can concentrate on clusters of overlapping niches;
  - Convergence within a cluster is linked through chains of overlapped clusters, with changes in one niche affecting all other linked niches; and
  - Communication time through long chains seems to dominate cluster convergence.

We hope that our partial models constitute a *foundation* for future completion of a general model of all significant multi-niche sharing behavior, both resource and fitness.

9.2.2 Practical Contributions

Here we offer concrete, practical implications of our results for users of GAs and niching. We briefly summarize specific suggestions, techniques, and guidelines for GA users. While the more general contributions discussed above are of greatest interest to GA theorists and researchers, the contributions listed below are targeted at GA designers and implementors, in other words those *applying* GAs to real problems *now*.

Use sharing to converge quickly to equilibrium. We now know the effect of sharing on selection. Sharing will cause the GA to converge rapidly to a diverse population of high fitness individuals. It will keep the population there over many generations, unless additional diversity is introduced (through exploration).

Use sharing even when the distribution of niches is not completely known. Contrary to some frequently stated criticisms of sharing, it is not brittle. Sharing is tolerant of niche overlap, and thus does not require evenly spaced niches. Nor does sharing require equally fit optima. Sharing degrades gracefully with increasing niche overlap, niche fitness difference, and decreasing population size.

Do not assume that sharing is too expensive. The addition of sharing does not have to require any additional fitness function evaluations. The $O(N^2)$ figure often given is simply in terms of *comparisons* of genotypes or phenotypes for similarity. Even this cost can be reduced to $O(N)$ by sampling for the niche counts, although at the expense of adding some noise to selection.

Use sharing to evolve the desired degrees of cooperation. Sharing's tolerance of overlap and fitness difference can be predicted and controlled. Furthermore, this control can be used to force sharing to distinguish cooperative groups from competitive groups. Thus sharing can automatically recognize and promote cooperation in an evolving population, while using competition between cooperative groups to find the *best* such groups. Thus sharing is generally useful for any application involving the evolution of cooperation.

Use sharing to stop premature convergence and to improve search. Even for simple GA optimization of a single objective function, in which only one (global) optimum is desired, sharing can improve GA performance (i.e., the probability of finding the optimum).

Use sharing to find and keep multiple optima. Sharing can be used to search for and maintain multiple alternative optima simultaneously, and *without* necessarily degrading the search for the single best optimum.

Apply sharing to a flat fitness plateau. Within a large equivalence class, much larger than N say, sharing can use the "edge effect" to distinguish certain cooperative groups. The favored groups can be useful solutions to certain types of problems. For example, the niches might represent a broad sampling of a large, convoluted tradeoff surface in a multiobjective problem. Or they might represent an optimal packing of shapes into some finite volume.

Combine GAs with sharing and Pareto selection to find Pareto optimal sets. The Niched Pareto GA is one possible way of combining sharing with Pareto selection in a GA. It has been shown to evolve improved tradeoff surfaces on open problems.

Take full advantage of resource (example) sharing in classifier systems. Resource sharing seems well-suited to promoting disjunctive concepts represented by multiple distinct rules. Such sharing has already been successfully incorporated into many evolutionary classifier systems. It appears to be a vital component of many LCSs.

Enhance the basic sharing methods. Techniques such as powersharing, rootsharing, elitist sharing, etc., can be used to make sharing more robust, stretching the limits of its applicability.

Yet sharing does have its limits. It is hoped that our bounding models of sharing's performance have shown us those limits, and that they will now allow us to successfully use sharing well within those limits.

9.3 Future Work

It is hoped that the results presented herein inspire four major directions for future investigations of sharing:

1. extensions to the model of cooperative-competitive niching,
2. further inquiry into the significance and possible utility of sharing equilibrium,
3. widespread creative experimentation with sharing and application to new problems, and
4. further attempts to tie our use of sharing in GAs to the role of sharing in nature.

Specific suggestions for extending the niching model in this dissertation can be found in the text. In general, the niching control map needs to be refined, and it needs to be used in a prescriptive way, so that it can tell practitioners how to set key niching parameters so as to place the cooperation-competition boundary where they want it. Furthermore, the model must be extended to arbitrary numbers of overlapping niches, not just niche pairs. Then the niching model will approach completeness.

As for the meaning of niching equilibrium, Chapter 6 results were tentative but suggestive. It seems that even without recombination and mutation, that is without any exploration at the chromosome level, selection plus sharing is performing complex calculations, exploring the space of population distributions. If these distributions represent good solutions to hard problems, then selection plus sharing might be a search algorithm in and of itself. Mapping the population distribution into a problem encoding seems plausible (e.g., using "voting" among individuals to make a population-based decision for classification). Population-based solutions is a little explored area of evolutionary computation, with the small exceptions of classifier systems and immune systems.

Sharing seems generally beneficial to most types of evolutionary computation, in terms of search, maintenance of diversity, and location of multiple alternative solutions. "Standard" or simple sharing could be incorporated more widely into both experimental and production GAs by GA implementors. In the meantime, GA designers and experimenters should feel emboldened to try more creative new approaches with sharing. Many types of problems have explicit finite resources. For example, in scheduling problems time on a particular machine is a finite resource. Furthermore, it is often a "flat fitness surface": imagine using equivalence class sharing to "pack" jobs onto the one dimensional fitness plateaus representing each processor's available time. We might call such a technique *temporal niching.*

Many of the interactions in learning classifier systems might be subsumed by sharing. For example, we assumed equally general rules in our model of resource sharing in the LCS. But we could model the interactions between more general and more specific rules by giving the more specific rules a greater **share** of each example, with the argument from natural metaphor being that the more general rule can't earn as much of a share since it is busy covering more territory. In many LCS systems, this punishment of overly general rules is implemented via a "tax" for each firing of the rule. Yet it seems that sharing can subsume the role of taxes in an LCS.

By looking to sharing first, before necessarily implementing a whole new mechanism, we might find many more creative and natural uses of this one general mechanism. As we pointed out in Chapter 3, sharing can be used to enforce the fixed population size required of almost every GA implementation, as it does in nature. Just as more and more users are turning to evolutionary computation for a new yet old way to solve hard problems, so we practitioners of EC might turn to niching via sharing as a new yet old and natural way to meet many of the demands of our increasingly sophisticated and complex evolutionary algorithms.

References

Ankenbrandt, C. A. (1991). An extension to the theory of convergence and a proof of the time complexity of genetic algorithms. In G. E. Rawlins (Ed.), *Foundations of Genetic Algorithms.* San Mateo, CA: Morgan Kaufmann, 53–68.

Beasley, D., Bull, D. R., & Martin, R. R. (1993). A sequential niche technique for multimodal function optimization. *Evolutionary Computation, 1*(2). 101–126.

Bethke, A. D. (1981). Genetic algorithms as function optimizers. (Doctoral dissertation, University of Michigan at Ann Arbor). *Dissertation Abstracts International, 41*(9), 3503B. (University Microfilms No. 81-06101)

Booker, L. B. (1982). Intelligent behavior as an adaptation to the task environment. *Dissertation Abstracts International, 43*(2), 469B. (University Microfilms No. 8214966)

Booker, L. B. (1989). Triggered rule discovery in classifier systems. In J. D. Schaffer, (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms (ICGA 3).* San Mateo, CA: Morgan Kaufmann. 265–274.

Cavicchio, D. J. (1970). *Adaptive search using simulated evolution.* Ph.D. thesis, University of Michigan, Ann Arbor, (University Microfilms No. 25-0199).

Cedeño, W., Vemuri, V. R., & Slezak, T. (1994). Multiniche crowding in genetic algorithms and its application to the assembly of DNA restriction-fragments. *Evolutionary Computation, 2*(4). 321–345.

Cieniawski, S. E. (1993). *An Investigation of the Ability of Genetic Algorithms to Generate the Tradeoff Curve of a Multi-objective Groundwater Monitoring Problem* (unpublished masters thesis, University of Illinois at Urbana-Champaign, Urbana, IL).

Cieniawski, S. E., Eheart, J. W., & Ranjithan, S. (1995). Using genetic algorithms to solve multiobjective groundwater monitoring problem. *Water Resource Research, 31*(2), 399–409.

Collins, R. J., & Jefferson, R. J. (1991). Selection in massively parallel genetic algorithms. In R. K. Belew & L. B. Booker (Ed.s), *Proceedings of the Fourth International Conference on Genetic Algorithms.* San Mateo, CA: Morgan Kaufmann, 249–256.

Davidor, Y. (1991). A naturally occurring niche & species phenomenon: the model and first results. In R. K. Belew & L. B. Booker (Ed.s), *Proceedings of the Fourth International Conference on Genetic Algorithms.* San Mateo, CA: Morgan Kaufmann, 257–263.

Dawid, H. (1994). A Markov chain analysis of genetic algorithms with a state dependent fitness function. *Complex Systems, 8.* 407–417.

De Jong, K. A. (1975). An analysis of the behavior of a class of genetic adaptive systems (Ph.D. thesis, University of Michigan, Ann Arbor). *Dissertation Abstracts International, 36*(10), 5140B. (University Microfilms No. 76-9381)

De Jong, K. A. (1992). Are genetic algorithms function optimizers? In R. Männer & B. Manderick, (Ed.s), *Parallel Problem Solving From Nature, 2.* Amsterdam: North-Holland, 3–13.

De Jong, K. A. (1993). Genetic algorithms are NOT function optimizers. In L. D. Whitley (Ed.), *Foundations of Genetic Algorithms, 2.* San Mateo, CA: Morgan Kaufmann, 5–17.

De Jong, K. A., Spears, W. M., & Gordon, D. F. (1995). Using Markov chains to analyze GAFOs. In L. D. Whitley, & M. D. Vose (Ed.s), *Foundations of Genetic Algorithms 3* (FOGA 3). San Francisco, CA: Morgan Kaufmann, 115–137.

de Neufville, R. (1990). *Applied Systems Analysis: Engineering Planning and Technology Management.* McGraw-Hill Publishing Company.

Deb, K. (1989). *Genetic algorithms in multimodal function optimization.* Masters thesis and TCGA Report No. 89002. Tuscaloosa, AL: Department of Engineering Mechanics, University of Alabama.

Deb, K., & Goldberg, D. E. (1989). An investigation of niche and species formation in genetic function optimization. In J. D. Schaffer (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms.* San Mateo, CA: Morgan Kaufmann, 42–50.

Deb, K., Horn, J., & Goldberg, D. E. (1993). Multimodal deceptive functions. *Complex Systems, 7*, 131–153.

Dobzhansky, T. (1937). *Genetics and the Origin of Species.* Morningside Heights, New York, NY: Columbia University Press.

Fonseca, C. M., & Fleming, P. J. (1993). Genetic algorithms for multiobjective optimization: formulation, discussion and generalization. In S. Forrest (Ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms.* San Mateo, CA: Morgan Kaufmann, 416-423.

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., & Perelson, A. S. (1993). Using genetic algorithms to explore pattern recognition in the immune system. *Evolutionary Computation, 1*(3). 191–211.

Giordana, A., & Neri, F. (1995). Search-intensive concept induction. *Evolutionary Computation, 3*(4). 375–416.

Giordana, A., Saitta, L., & Zini, F. (1994). Learning disjunctive concepts with distributed genetic algorithms. *Proceedings of the first IEEE conference on evolutionary computation, IEEE world congress on computational intelligence, volume 1.* Piscataway, NJ: IEEE Service Center, 115–119.

Goertzel, B. (1993). *The Evolving Mind.* Langhorne, PA: Gordon and Breach.

Goldberg, D. E. (1983). *Computer-aided gas pipeline operation using genetic algorithms and rule learning* (Ph.D. thesis, University of Michigan).

Goldberg, D. E. (1987). Simple genetic algorithms and the minimal deceptive problem. In L. Davis (Ed.), *Genetic algorithms and simulated annealing* (pp. 74–88). London: Pitman Publishing.

Goldberg, D. E. (1989a). *Genetic algorithms in search, optimization, and machine learning.* Reading, MA: Addison-Wesley.

Goldberg, D. E. (1989b). Sizing populations for serial and parallel genetic algorithms. In J. D. Schaffer, (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms.* San Mateo, CA: Morgan Kaufmann, 70–79.

Goldberg, D. E. (1989c). Genetic algorithms and Walsh functions: part I, a gentle introduction. *Complex Systems, 3*, 129–152.

Goldberg, D. E. (1989d). Genetic algorithms and Walsh functions: part II, deception and its analysis. *Complex Systems, 3*, 153–171.

Goldberg, D. E. (1990). A note on Boltzmann tournament selection for genetic algorithms and population-oriented simulated annealing. *Complex Systems, 4,* 445–460.

Goldberg, D. E. (1991). Construction of high-order deceptive functions using low-order Walsh coefficients. *Annals of Mathematics and Artificial Intelligence, 5*, 35–48.

Goldberg, D. E. (1993). Making genetic algorithms fly: a lesson from the Wright brothers. *Advanced Technology for Developers*, 2, February. 1–8.

Goldberg, D. E. (1994a). Genetic and evolutionary algorithms come of age. *Communications of the Association for Computing Machinery, 37(3)*, March 1994, 113–119.

Goldberg, D. E. (1994b). personal communication.

Goldberg, D. E., & Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. In G. E. Rawlins (Ed.), *Foundations of Genetic Algorithms* (FOGA). San Mateo, CA: Morgan Kaufmann. 69–93.

Goldberg, D. E., Deb, K., & Clark, J. H. (1992). Genetic algorithms, noise, and the sizing of populations. *Complex Systems, 6*, 333–362.

Goldberg, D. E., Deb, K., & Horn, J. (1992). Massive multimodality, deception, and genetic algorithms. In R. Männer & B. Manderick, (Ed.s), *Parallel Problem Solving From Nature, 2.* Amsterdam: North-Holland, 37–46.

Goldberg, D. E., Deb, K., & Korb, B. (1991). Don't worry, be messy. In R. K. Belew, & L. B. Booker (Ed.s), *Proceedings of the Fourth International Conference on Genetic Algorithms* (pp. 24–30). San Mateo, CA: Morgan Kaufmann.

Goldberg, D. E., Horn, J., & Deb, K. 1992. *What makes a problem hard for a classifier system?* (IlliGAL Report No. 92007). Urbana, IL: Department of General Engineering, University of Illinois at Urbana-Champaign. Presented as an extended abstract at *The First International Workshop on Learning Classifier Systems.* Houston, Texas, USA, October 1992.

Goldberg, D. E., & Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimization. In J. Grefenstette, (Ed.), *Proceedings of the Second International Conference on Genetic Algorithms*, Hillsdale, NJ: Lawrence Erlbaum Associates, 41–49.

Goldberg, D. E., & Segrest, P. (1987). Finite Markov chain analysis of genetic algorithms. In J. Grefenstette, (Ed.), *Proceedings of the Second International Conference on Genetic Algorithms*, Hillsdale, NJ: Lawrence Erlbaum Associates, 1–8.

Goldsmith, S. Y. (1989). *Steady-state analysis of a simple classifier system.* Ph.D. thesis, University of New Mexico, Albuquerque.

Greene, D. P., & Smith, S. F. (1994). Using coverage as a model building constraint in learning classifier systems. *Evolutionary Computation, 2*(1). 67–91.

Guddat, J., Vasquez, F. G., Tammer, K., & Wendler, K. (1985). *Multiobjective and Stochastic Optimization Based on Parametric Optimization.* Mathematical Research, Band 26. Berlin, DDR: Akademie-Verlag.

Harik, G. (1994). *Finding multiple solutions in problems of bounded difficulty* (IlliGAL Report. No. 94002). Urbana, IL: Department of General Engineering, University of Illinois at Urbana-Champaign.

Hightower, R. R., Forrest, S., & Perelson, A. S. (1995). The evolution of emergent organization in immune system gene libraries. In L. J. Esherman (Ed.), *Proceedings of the Sixth International Conference on Genetic Algorithms.* San Francisco, CA: 344–350.

Hofbauer, J., & Sigmund, K. (1988). *The Theory of Evolution and Dynamical Systems.* Cambridge, G.B.: Cambridge University Press.

Hoffmeister, F., & Bäck, T. (1990). *Genetic algorithms and evolutionary strategies: similarities and differences* (Technical Report "Grüne Reihe" No. 365). Dortmund, Germany: Department of Computer Science, University of Dortmund.

Holland, J. H. (1971). Processing and processors for schemata. *Proceedings of the Symposium on Associative Information Techniques.* 127–146.

Holland, J. H. (1975). *Adaptation in natural and artificial systems.* Ann Arbor, MI: University of Michigan Press.

Holland, J. H. (1985). Properties of the bucket-brigade algorithm. In Grefenstette, J. J. (Ed.), *Proceedings of an International Conference on Genetic Algorithms and Their Applications (ICGA 1).* Pittsburgh, PA: Carnegie-Mellon University. 1–7.

Horn, J. (1993). Finite Markov chain analysis of genetic algorithms with niching. In S. Forrest, (Ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms.* San Mateo, CA: Morgan Kaufmann, 110–117.

Horn, J. & Goldberg, D. E. (1995). Genetic algorithm difficulty and the modality of fitness landscapes. In L. D. Whitley, & M. D. Vose (Ed.s), *Foundations of Genetic Algorithms 3* (FOGA 3). San Francisco, CA: Morgan Kaufmann, 243– 269.

Horn, J., & Goldberg, D. E. (1996). Natural niching for evolving cooperative classifiers. In J. R. Koza, D. E. Goldberg, D. B. Fogel, & R. L. Riolo (Ed.s.), *Genetic Programming, Proceedings of the First Annual Conference 1996.* Cambridge, MA: The MIT Press. 553–564.

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

Horn, J. & Nafpliotis, N. (1993). *Multiobjective optimization using the niched Pareto genetic algorithm* (IlliGAL Report No. 93005). Urbana, IL: Department of General Engineering, University of Illinois at Urbana-Champaign.

Horn, J., Nafpliotis, N., & Goldberg, D. E. (1994). A niched Pareto genetic algorithm for multiobjective optimization. *Proceedings of the first IEEE conference on evolutionary com-*

putation, IEEE world congress on computational intelligence, volume 1. Piscataway, NJ: IEEE Service Center, 82–87.

Horn, J. (1997). Multicriteria decision making. (to appear in) T. Bäck, and D. Fogel, (Ed.s), *The Handbook of Evolutionary Computation*, Oxford University Press and the Institute of Physics Publishing.

Huberman, B. A. (1988). The ecology of computation. In B. A. Huberman (Ed.), *The Ecology of Computation.* Amsterdam, Holland: Elsevier Science Publishers B. V. 1–4.

Huberman, B. A., & Hogg, T. (1988). The behaviour of computational ecologies. In B. A. Huberman (Ed.), *The Ecology of Computation.* Amsterdam, Holland: Elsevier Science Publishers B. V. 77–115.

Keeney, R. L., Lathrop, J. F., & Sicherman, A., (1986). An analysis of Baltimore Gas and Electric Company's technology choice. *Operations Research.* Volume 34, Number 1, January-February 1986, 18-39.

Keeney, R., & Raiffa, H., (1976). *Decisions with Multiple Objectives.* Wiley, New York.

Mahfoud, S. W. (1991). Finite Markov chain models of an alternative selection strategy for the genetic algorithm. *IlliGAL Report No. 91007.*

Mahfoud, S. W. (1992). Crowding and preselection revisited. In R. Männer & B. Manderick (Ed.s), *Parallel Problem Solving From Nature, 2.* Amsterdam: North-Holland, 27–36.

Mahfoud, S. W. (1993). Simple analytical models of genetic algorithms for multimodal function optimization. One-page summary in S. Forrest, (Ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms.* San Mateo, CA: Morgan Kaufmann, 643. Full paper available as IlliGAL Report No. 93001. Urbana: Department of General Engineering, University of Illinois at Urbana-Champaign.

Mahfoud, S. W. (1994). Genetic drift in sharing methods. *Proceedings of the first IEEE conference on evolutionary computation, IEEE world congress on computational intelligence, volume 1.* pp. 67–72.

Mahfoud, S. W. (1995a). Population size and genetic drift in fitness sharing. In L. D. Whitley & M. D. Vose, (Ed.s), *Foundations of Genetic Algorithms, 3 (FOGA 3).* San Francisco, CA: Morgan Kaufmann. 185–224.

Mahfoud, S. W. (1995b). *Niching Methods for Genetic Algorithms* (Ph.D. thesis, University of Illinois at Urbana-Champaign). Available as IlliGAL Report No. 95001. Urbana: Department of General Engineering, University of Illinois at Urbana-Champaign.

Mahfoud, S. W., & Goldberg, D. E. (1992). A genetic algorithm for parallel simulated annealing. In R. Männer & B. Manderick, (Ed.s), *Parallel Problem Solving From Nature, 2.* Amsterdam: North-Holland, 301–310.

McCallum, R. A., & Spackman, K. A. 1990. Using genetic algorithms to learn disjunctive rules from examples. In B. W. Porter & R. J. Mooney, (Ed.s), *Machine Learning: Proceedings of the Seventh International Conference.* Palo Alto, CA: Morgan Kaufmann. 149–152.

Michielssen, E., & Weile, D. S. (1995). Electromagnetic system design using genetic algorithms. In G. Winter, J. Périaux, M. Galán, & P. Cuesto (Ed.s), *Genetic Algorithms in Engineering and Computer Science.* Chichester, UK: John Wiley & Sons. 345–369 (Chapter 18).

Morgan, D. R., (1990). Decision making under uncertainty using a new chance constrained programming technique: a groundwater reclamation application. *Unpublished doctoral dissertation, University of Illinois, Department of Civil Engineering.*

Morgan, D. R., & Goulter, I. C. (1985). Optimal urban water distribution design. *Water Resources Research, 21*(5), 642–652.

Murphy, L. J., Simpson, A. R., & Dandy, G. C. (1993). *Pipe network optimization using an improved genetic algorithm.* Research Report No. R109, Department of Civil and Environmental Engineering, University of Adelaide, Australia.

Neri, F., & Saitta, L. (1995). Analysis of genetic algorithms evolution under pure selection. In L. J. Eshelman, (Ed.), *Proceedings of the Sixth International Conference on Genetic Algorithms.* San Francisco, CA: Morgan Kaufmann. 32–39.

Neri, F., & Saitta, L. (1996). An analysis of the "universal suffrage" selection operator. *Evolutionary Computation, 4*(1), 87–107.

Ngo, J. T., & Marks, J. (1993). A Markov chain framework for the simple genetic algorithm. *Evolutionary Computation, 1*(3), 269–298.

Nix, A. E., & Vose, M. D. (1992). Modeling genetic algorithms with Markov chains. *Annals of Mathematics and Artificial Intelligence, 5*(1).

Oei, C. K., Goldberg, D. E., & Chang, S. (1991). *Tournament selection, niching, and the preservation of diversity* (IlliGAL Report No. 91011). Urbana, IL: Department of General Engineering, University of Illinois at Urbana-Champaign.

Olafsson, S. (1996). Resource allocation as an evolving strategy. *Evolutionary Computation, 4*(1), 33–55.

Pareto, V., (1896). *Cours d'Économie Politique, 1.* Lausanne, Switzerland: F. Rouge.

Pitt, L., & Valiant, L. G. (1988). Computational limitations on learning from examples. *Journal of the Association for Computing Machinery, 35*(4), 965–984.

Radcliffe, N. J., & Surry, P. D. (1994). Co-operation through hierarchical data mining. *EPCC technical report tr9409.* Edinburgh Parallel Computing Center.

Ranjithan, S. (1993). *Personal communication.*

Ritzel, B., (1992). Genetic algorithms in multiple objective optimization. *Unpublished analytical class project for GE 493, University of Illinois at Urbana-Champaign, Spring 1992.*

Roughgarden, J. (1979). *Theory of Population Genetics and Evolutionary Ecology: An Introduction.* New York, New York: Macmillan Publishing Co., Inc.

Sawaragi, Y., Nakayama, H., & Tanino, T., (1985). *Theory of Multiobjective Optimization.* Mathematics in Science and Engineering, Volume 176. Academic Press, Incorporated (Harcourt Brace Jovanovich, publishers).

Schaake, J. C., & Lai, D. (1969). *Linear programming and dynamic programming applications to water distribution network design.* Report 116, Hydrodynamics Laboratory, Department of Civil Engineering, MIT, Cambridge, MA.

Schaffer, J. D., (1984). Some experiments in machine learning using vector evaluated genetic algorithms. *Unpublished doctoral dissertation, Vanderbilt University.*

Schaffer, J. D., (1985). Multiple objective optimization with vector evaluated genetic algorithms. In J. Grefenstette, ed., *Proceedings of an International Conference on Genetic Algorithms and their Applications.*, 93–100.

Smith, R. E., Forrest, S., & Perelson, A. S. (1993). Searching for diverse, cooperative populations with genetic algorithms. *Evolutionary Computation, 1*(2). 127–150.

Smith, R. E., & Valenzuela-Rendón, M. (1989). A study of rule set development in a learning classifier system. In J. D. Schaffer, (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms*, San Mateo, CA: Morgan Kaufmann, 340–346.

Srinivas, N. (1994). *Multiobjective optimization using nondominated sorting in genetic algorithms* (unpublished masters thesis, Indian Institute of Technology, Kanpur, India).

Srinivas, N., & Deb, K. (1995). Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation, 2*(3). 221–248.

Suzuki, J. (1993). A Markov chain analysis on a genetic algorithm. In S. Forrest (Ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms.* San Mateo, CA: Morgan Kaufmann, 146–153.

Suzuki, J. (1997). A further result on the Markov chain model of GAs and its application to a SA-like strategy. (To appear in) R. K. Belew & M. D. Vose, (Ed.s), *Foundations of Genetic Algorithms IV.* San Francisco, CA: Morgan Kaufmann.

Tamaki, H., Kita, H. & Kobayashi, S. (1996). Multi-objective optimization by genetic algorithms: a review. In *Proceedings of the Third IEEE International Conference on Evolutionary Computation (ICEC '96).* Piscataway, NJ: IEEE Service Center. 517–522.

Thierens, D., & Goldberg, D. E. (1993). Mixing in genetic algorithms. In S. Forrest (Ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms.* San Mateo, CA: Morgan Kaufmann, 38–45.

Thierens, D. (1995). Analysis and Design of Genetic Algorithms. *Unpublished doctoral dissertation, Catholic University of Leuven, Belgium.*

Valiant, L. G. (1984). A theory of the learnable. *Communications of the Association for Computing Machinery, 17*(11), 1134–1142.

Vose, M. D. (1993). Modeling simple genetic algorithms. In L. D. Whitley (Ed.), *Foundations of Genetic Algorithms 2* (pp. 63–73). San Mateo, CA: Morgan Kaufmann.

Vose, M. D. (1995). Modeling simple genetic algorithms. *Evolutionary Computation, 3*(4). 453–472.

Wilson, S. W. (1986). *Classifier learning of a Boolean function.* (Research Memo RIS No. 27r). Cambridge: The Rowland Institute for Science.

Wilson, S. W. (1987). Classifier systems and the animat problem. *Machine Learning, 2.* 199–228.

Wilson, S. W. (1994). ZCS: A zeroth level classifier system. *Evolutionary Computation, 2*(1). 1–18.

Wilson, S. W. (1995). A classifier system based on accuracy. *Evolutionary Computation, 3*(2). 149–175.

Wilson, S. W. & Goldberg, D. E. (1989). A critical review of classifier systems. In J. D. Schaffer, (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms*, San Mateo, CA: Morgan Kaufmann. 244–255.

Yates, D. F., & Fairly, A. (1994). Evolutionary stability in simple classifier systems. In T. C. Fogarty (Ed.), *Evolutionary Computing.* Berlin: Springer-Verlag. 28–37.

Yin, X., & Germay, N. (1993). Improving genetic algorithms with sharing through cluster analysis. In S. Forrest (ed.), *Proceedings of the fifth international conference on genetic algorithms* (p. 100). San Mateo, CA: Morgan Kaufmann.

Vita

Jeffrey Horn

Home:
701 Summit Avenue, Apt. 11
Marquette, Michigan 49855
Phone: 906/227-4789

E-mail: jhorn@nmu.edu

Office:
Department of Mathematics
and Computer Science
1401 Presque Isle Avenue
Marquette, Michigan 49855-5340
Phone: 906/227-1607
Fax: 906/227-2010

EDUCATION

Ph.D. Computer Science: University of Illinois at Urbana-Champaign, IL, USA
October 1997.

M.S. Computer Science: University of Illinois at Urbana-Champaign, IL, USA
August 1995.

B.A. Computer Science: Cornell University, Ithaca, NY, USA
May 1985. *Concentration in Operations Research and Industrial Engineering*

ACADEMIC EXPERIENCE

8/96 – Present: **Assistant Professor, Northern Michigan University**. Teach upper level computer science classes (algorithms, theory, programming languages). Continue leading active research program in the artificial evolution of cooperative agents which together solve difficult problems in engineering and machine learning.
9/95 – 6/96: **Graduate Researcher, Illinois Genetic Algorithms Laboratory**. Model and predict the evolution of optimal, cooperative populations in genetic algorithms. Supervise the Illinois Genetic Algorithms Laboratory (IlliGAL). Supported by the Air Force Office of Scientific Research. (Supervisor: David E. Goldberg)

+ Demonstrated the correspondence between *implicit* and *explicit* niching,
+ Derived closed form bounds on niching convergence times,
+ Proved that cooperation maximizes a multiplicative population function, and

+ Managed hardware, software, facilities, and personel at the IlliGAL, establishing lab procedures for maintenance, internal communication, internet (ftp, WWW) presence, technical report service, and an archival library of thousands of papers on evolutionary computation.

6/92 – 8/95: **NASA Graduate Fellow, University of Illinois at Urbana-Champaign**. Performed research on the modality of fitness landscapes and the implications for genetic algorithms. Also began research into the nature and applicability of evolved cooperation under genetic algorithm selection pressure. (Faculty advisor: David E. Goldberg; NASA Technical Advisors: Robert Savely and Leh Wang)

+ Proposed and won $66,000 research grant,
+ Managed $18,000 fund for travel and other research support,
+ Presented results at international conferences and workshops,
+ Wrote annual research reports for NASA,
+ Visited NASA Johnson Space Center to present results immediately practicable to NASA JSC scientists and engineers,
+ Integrated results on multimodality and genetic algorithms (MS thesis), and
+ Proved the existence of steady-state cooperative populations in highly competitive genetic algorithms and classifiers systems.

1/92 – 5/92: **Graduate Research Assistant, Center for Complex Systems Research, the Beckman Institute, University of Illinois at Urbana-Champaign**. Tracked the evolution of complexity and organism interdependence in artificial life simulations. (Supervisor: Norman Packard)

+ Developed novel measure of complexity for stimulus-response strategies, and
+ Published new method at the European Conference on Artificial Life in Paris.

1/82 – 12/83: **Computer Science Course Consultant, Department of Computer Science, Cornell University**. Undergraduate assistant to course instructors for first through third year computer science classes (CS 100, 211, and 280).

+ Led regular tutorial sessions,
+ Held scheduled office hours for student support,
+ Supervised computer labs,
+ Proctored exams, and
+ Graded exams, programs, and problem sets.

INDUSTRY EXPERIENCE

8/89 – Present: **Independent Computer Consultant**, Urbana, IL. Conduct seminars, publish articles, and deliver written and verbal expertise directly to clients, in the areas of computer networking and evolutionary computation.

8/89 – 12/91: **Network Designer, University of Illinois Computing and Communications Services Office**, Urbana, IL. Designed local area and building-wide computer networks for academic departments and campus units, including gateways to the campus backbone and the world-wide internet. Coordinated the installation and maintenance of such "subnets".

3/88 – 3/91: **Instructor, Systems Technology Forum**, Fairfax, VA. Developed and conducted three-day seminars on data communications, local area networks, and network protocols and architectures for groups of data communications professionals at locations across the United States.

10/88 – 7/89: **Manager, Ernst & Young**, Fairfax, VA. Managed teams of professional consultants engaged in local area and campus networking projects for clients such as Martin Marietta Energy Systems (Oak Ridge National Laboratories), and the International Finance Corporation.

+ Developed proposals,
+ Managed project budgets of $50,000 to $150,000,
+ Supervised project teams of one to five members,
+ Interacted daily or weekly with senior managers of clients, and
+ Took several fixed price projects from start to finish, on-time and under budget.

7/85 – 9/88: **Consultant, Network Strategies, Inc.**, Fairfax, VA. Provided consulting expertise on network protocols and architectures to clients in industry and state and local government.

+ Taught regular in-house seminars for technical training,
+ Brought in leads that eventually resulted in engagements,
+ Developed in-house project management software system, and
+ Created and maintained company library of communications standards.

Summers 1983,84: **Software Developer**, NY. Developed small and large-scale software systems to support electrical engineers at Underwriters Laboratories, Melville, NY, and medical researchers at the Institute of Chronobiology, Cornell Medical Center, White Plains, NY.

> (software projects include: automated meter calibration, test, and diagnosis; large meter statistics database; data communications protocol for reliable control of remote device over noisy serial line; modular automated graphing tools to control large plotter; synchronization and control of I/O buffers and registers; design and implementation of protocols for data transfer between automated assay machine and a PDP/11; Fortran to C conversion of research software)

SUMMARY OF TEACHING EXPERIENCE

Served three years as a professional instructor of intensive technology seminars.
Developed and/or significantly revised three industry seminars.
Led several class sessions for senior and graduate level university courses.
Worked as tutor/proctor/grader/lab consultant for first to third year CS courses.

PROFESSIONAL ACTIVITIES

Academic

Review Board, *Applied Intelligence* journal. 1995–present.

Program Committee, *Genetic Programming* (GP 97). 1997.

Program Committee, *Seventh International Conference on Genetic Algorithms* (ICGA 7). 1997.

Program Committee, *IEEE International Conference on Evolutionary Computation* (IEEE ICEC 4). 1997.

Program Committee, *Genetic Programming* (GP 96). 1996.

Program Committee, *Foundations of Genetic Algorithms* (FOGA 4). 1996.

Program Committee, *Sixth International Conference on Genetic Algorithms* (ICGA 6). 1995.

(Reviewed approximately twelve journal submissions, several book chapters, and over twenty conference and workshop papers.)

Member, ACM (1989–90) and ASEE (1993–94).

Industry

Program Advisory Committee (industry advisor on networking), *Federal Office Systems Exposition '89* (FOSE '89). Designed the *Integration Through LANs* session track.

Session Chair, "Merging LANs and Minicomputers: Choosing the Best of Both Worlds," *FOSE '89.*

Session Chair, "GOSIP: Government OSI Protocols," *Communication Networks '89* (ComNet '89).

Session Chair,"Fiber Optic Solutions for Backbone Nets," *INTERFACE '88.*

Session Chair, "Beyond the PC LAN: Growth Options and Strategies," *FOSE '88.*

AWARDS

General Electric Teaching Incentive Grant, 1996.

National Aeronautics and Space Administration (NASA) Graduate Student Research Fellow, 1992–1995.

University of Illinois Conference Travel Grant, 1991.

National Council of Teachers of English (NCTE) Achievement Award in Writing, 1981.

National Merit Scholarship Semi-finalist, 1981.

RESEARCH GRANTS

Horn, J. (1996–1997). Northern Michigan University Faculty Mini-Grant, $1500.

Horn, J., & Goldberg, D. E. (1992–1995). *Generalized deception and genetic algorithms.* NASA to the University of Illinois, $66,000.

Goldberg, D. E., Kargupta, H., & Horn, J. (1995–1996). *Rapid solutions to hard problems using messy genetic algorithms on IBM workstations.* IBM (Shared University Research grant) to the University of Illinois, 1000 cpu hours supercomputing time.

RESEARCH GOALS

Genetic Algorithms and the Evolution of Cooperation
The use of genetic algorithms/evolutionary computation to evolve cooperative organizations of individuals to solve hard problems. *Research Needs:* complete model of different levels of cooperation and competition in an evolving population; characterization of cooperation-competition tension and quantification of phase transition; proof of convergence to cooperative populations; effect of cooperative diversity on evolutionary search.

Machine Learning and Computational Intelligence
The application of cooperative artificial evolution to learn semi-decomposable concept descriptions (e.g., disjunctive normal form). The use of cooperative evolution in the discovery of new agents, in the formation and selection of cooperative and competing groups of intelligent agents, and in the teaming of specialists and generalists in collaborative systems. *Research Needs:* search-intensive concept induction; adaptive determination of optimal tradeoff among learning criteria (e.g., accuracy, coverage, and conciseness of concept description); distributed concept representations.

Engineering and Computer Science Problem Solving
The application of evolutionary algorithms to the probablistic solution of real-world engineering optimization and design problems and to the PAC-like approximate solution of general, intractable problems in computer science. *Research Needs:* robust, reliable, population-based evolutionary search of large multimodal solution spaces; simultaneous discovery of multiple, al-

ternative solutions; enhanced reliability and success of search by maintenance of useful diversity; multiobjective problem solving using populations to sample and represent tradeoff surfaces.

PUBLICATIONS

Thesis

- Horn, J. (1995). Genetic algorithms, problem difficulty, and the modality of fitness landscapes. Masters thesis, Department of Computer Science, University of Illinois at Urbana-Champaign. *Illinois Genetic Algorithms Laboratory (IlliGAL) Report Number 95004.*

Refereed Journals

- Horn, J., Goldberg, D.E., & Deb, K. (1994). Implicit niching in a learning classifier system: nature's way. *Evolutionary Computation, 2*(1), 37–66. (Early version distributed as *IlliGAL Report No. 94001.*)

- Deb, K., Horn, J., & Goldberg, D.E. (1993). Multimodal deceptive functions. *Complex Systems, 7*(2), 131–153. (Early version distributed as *IlliGAL Report No. 92003.*)

Refereed International Conferences

- Horn, J. (1996). Genetic algorithms (with niching) in search, optimization, and machine learning. In L. D. Whitley, & M. D. Vose (Ed.s), *Foundations of Genetic Algorithms, 4* (to appear).

- Horn, J., & Goldberg, D. E. (1995). Genetic algorithm difficulty and the modality of fitness landscapes. In L. D. Whitley, & M. D. Vose (Ed.s), *Foundations of Genetic Algorithms, 3 (FOGA 3).* San Francisco, CA: Morgan Kaufmann, 243–269. (Early version distributed as *IlliGAL Report No. 94006.*)

- Horn, J., Goldberg, D.E., & Deb, K. (1994). Long path problems. In Y. Davidor, H.-P. Schwefel, & R. Männer (Ed.s), *Lecture Notes in Computer Science, Volume 866: Parallel Problem Solving from Nature-PPSN III.* Berlin: Springer-Verlag, 149–158. (Early version distributed as *IlliGAL Report No. 92011.*)

- Horn, J., Nafpliotis, N., & Goldberg, D.E. (1994). A niched Pareto genetic algorithm for multiobjective optimization. *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, Volume 1.* Piscataway, NJ: IEEE Service Center, 82–87. (Original, long version distributed as *IlliGAL Report No. 93005.*)

- Horn, J. (1993). Finite Markov chain analysis of genetic algorithms with niching. In S. Forrest (Ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms.* San Mateo, CA: Morgan Kaufmann, 110–117. (Early version distributed as *IlliGAL Report No. 93002.*)

- Horn, J. (1992). Measuring the evolving complexity of stimulus-response organisms. In F. J. Varela, & P. Bourgine (Ed.s), *Toward a Practice of Autonomous Systems (Proceedings*

of the First European Conference on Artificial Life. Cambridge, MA: The MIT Press (A Bradford Book), 365–374.

- Goldberg, D.E., Deb, K., & Horn, J. (1992). Massive multimodality, deception, and genetic algorithms. In R. Männer, & B. Manderick (Ed.s), *Parallel Problem Solving From Nature, 2*. Amsterdam, The Netherlands: North-Holland, 37–46. (Early version distributed as *IlliGAL Report No. 92005*.)

Invited Chapters/Papers

- Horn, J. (forthcoming). Multiple criteria decision making. *Handbook of Evolutionary Computation*. Oxford University Press.

- Horn, J., & Goldberg, D. E. (1996). Natural niching for evolving cooperative classifiers. In J. R. Koza, D. E. Goldberg, D. B. Fogel, & R. L. Riolo (Ed.s), *Genetic Programming 1996*. Cambridge, MA: The MIT Press. 553–564.

Refereed International Workshops

- Horn, J., & Nafpliotis, N. (1993). Niching along the Pareto optimal front: multiobjective optimization using genetic algorithms with sharing. *The Fifth International Conference on Genetic Algorithms, Workshop on Niching Methods*, University of Illinois at Urbana-Champaign, Urbana, IL, July, 1993. (presented abstract)

- Horn, J. (1992). Interactions among organisms. *Artificial Life III*, Sweeney Convention Center, Santa Fe, NM, June, 1992. (presented abstract)

- Goldberg, D.E., Horn, J., & Deb, K. (1992). What makes a problem hard for a classifier system? *The First International Workshop on Learning Classifier Systems*, Johnson Space Center, Houston, TX, October, 1992. (presented long abstract, now distributed as *IlliGAL Report No. 92007*.)

Additional Technical Reports (not yet published)

- Goldberg, D. E., Kargupta, H., Horn, J., & Cantu-Paz, E. (1995). Critical deme size for serial and parallel genetic algorithms. 13 pp. *IlliGAL Report No. 95002*.

- Horn, J., & Nafpliotis, N. (1993). Multiobjective optimization using the niched Pareto genetic algorithm. *IlliGAL Report No. 93005*. 32 pp. (Summary published as (Horn, Nafpliotis, & Goldberg, 1994) above.)

Industry and Trade Publications

- Horn, J. (1989). Peripherals assume role of multiuser servers. *Network World, 6*(11). 1,35,37–38,49. (feature article)

- Horn, J. (1988). GOSIP: the end of TCP/IP? *UNIX World, V*(11). 119–120. (guest column)

- Horn, J. (1988). Minicomputer makers spruce up LAN offerings. *Network World, 5*(39). 1,56-58,62,82. (feature article)
- Horn, J. (1988). Networking for the future of office automation. *INTERFACE '88 Papers Proceedings.* New York, NY: McGraw Hill, 192-196.
- Horn, J. (1988). The long, bumpy migration path to OSI. *Network World, 5*(35). 27. (opinion column)
- Horn, J. (1987). Get protocol-smart. *Network World, 4*(33). 26. (opinion column)
- Horn, J. (1987). The pillage of OSI. *Network World, 4*(9). 27,34. (opinion column)
- Chartoff, M. R., & Horn, J. (1988). Migration to an OSI-based FDDI multi-site network. *ENTERPRISE Conference Proceedings.* Dearborn, MI: Society of Manufacturing Engineers, 4-75–4-97.
- Passmore, L. D., & Horn, J. (1988). GOSIP to govern federal nets. *Network World, 5*(9). 1,35,38–40. (feature article)
- Chartoff, M. R., & Horn, J. (1987). Factory network architectures: collisions or token resistance? *Network World, 4*(46). 43–44,46–48. (feature article)
- Passmore, L. D., & Horn, J. (1987–1989). Monthly column in a Japanese Ministry of International Trade and Industry (MITI) newsletter, analyzing trends in US communications standards.

Industry Conference Presentations

- Horn, J. (1989). Wide area LANs. *Information Resources Management Conference, 1989* (IRMCO '89). Richmond, VA, September 5–8, 1989. (IRMCO's typical 300–400 attendees are senior federal managers, with average GS-15 grade level.)
- Horn, J. (1988). OSI migration tools: when to use them. *OSI Product Integration Conference*, McLean, VA, Nov. 29 – Dec. 2, 1988.
- Horn, J. (1988). LANs and LAN issues. *Information Resources Management Conference, 1988* (IRMCO '88). Richmond, VA, September 6–9, 1988.
- Horn, J. (1988). Migration to an ISO-based FDDI multi-site network. *Enterprise Networking Event 1988 International* (ENE '88i). Baltimore, MD, June 5–9, 1988.
- Horn, J. (1988). Overview of LANs and LAN issues. Tutorial presented at *The State of Maine Networking Conference.* Augusta, ME, May 17, 1988.
- Horn, J. (1988). Beyond the PC isLANd. *FOSE '88.* Washington, D.C., March 8–10, 1988.
- Horn, J. (1987). DECnet on Ethernet. *VAX Business Users Forum.* New York, NY, October 26–28, 1987.
- Horn, J. (1987). Internetwork bridging of 802 LANs. *National Networks Conference*, Washington, D.C., June, 1987.

Quotations

Quoted in trade periodicals such as *Computer World*, *Network World*, *Digital Review*, and *Communications*, and in industry newsletters such as *OPEN OSI Product & Equipment News*, and *SNA Communications Report*.

CONSULTING ACTIVITIES (selected examples)

Genetic algorithm patent application. (1994 – present). Client confidential.
Building backbone network and long term protocol integration strategy. (1989). Center for Naval Analysis.
Campus-wide data communications architecture. (1989). Letterkenney US Army Depot.
Minicomputer-PC-LAN office automation strategy. (1988). International Finance Corporation.
Oak Ridge internetworking backbone and protocol interoperability plan. (1988). US Department of Energy.
Sustainable base network architecture. (1988). US ARMY FEDSIM (Fort Belvoir, VA).
Long-range data communications plan for computer integrated manufacturing (CIM). (1988). Martin Marietta Energy Systems.
Electronic data interchange (EDI) migration plan. (1987). Uniform Code Council.
Operating procedures for implementing the national telecommunications service priority system. (1986). US National Communications System.
Analysis of dedicated data links to overseas embassies. (1986). US Department of State.
Integrated data and telecommunications operations and procedures. (1986). The World Bank.
Defense Data Network architectures and protocols seminar development. (1985). The US Defense Communications Agency.

PROFESSIONAL TRAINING

Introduction to Data Communications (three day seminar).
Local Area Networks (three day seminar).
Network Protocols and Standards (three day seminar).
The Defense Data Network (five day seminar).
Systems Network Architecture (three day seminar).
T1 Networking (three day seminar).
X.25 Wide Area Networks (three day seminar).
Network Design (three day seminar).
Network Management Systems (three day seminar).
Ernst & Whinney Professional Consultants Training Course (five day seminar).