Natural Niching for Evolving Cooperative Classifiers

Jeffrey Horn
Department of Computer Science
and the Illinois Genetic Algorithms Laboratory
University of Illinois at Urbana-Champaign
117 Transportation Building
104 South Mathews Avenue
Urbana, IL 61801-2996
Phone: 217/333-2346
Fax: 217/244-5705
Email: jeffhorn@uiuc.edu
WWW: http://www.uiuc.edu/ph/www/illigal

David E. Goldberg
Department of General Engineering
and the Illinois Genetic Algorithms Laboratory
University of Illinois at Urbana-Champaign
117 Transportation Building
104 South Mathews Avenue
Urbana, IL 61801-2996
Phone: 217/333-0897
Fax: 217/244-5705
Email: deg@uiuc.edu
WWW: http://www.uiuc.edu/ph/www/illigal

PREPRINT
(camera-ready)

As accepted for publication 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.

Natural Niching for Evolving Cooperative Classifiers

Jeffrey Horn
Department of Computer Science
and the Illinois Genetic Algorithms Laboratory
University of Illinois at Urbana-Champaign
117 Transportation Building
104 South Mathews Avenue
Urbana, IL 61801-2996
Phone: 217/333-2346
Fax: 217/244-5705
Email: jeffhorn@uiuc.edu
WWW: http://www.uiuc.edu/ph/www/illigal

David E. Goldberg
Department of General Engineering
and the Illinois Genetic Algorithms Laboratory
University of Illinois at Urbana-Champaign
117 Transportation Building
104 South Mathews Avenue
Urbana, IL 61801-2996
Phone: 217/333-0897
Fax: 217/244-5705
Email: deg@uiuc.edu
WWW: http://www.uiuc.edu/ph/www/illigal

ABSTRACT

An *evolutionary classifier*, such as a learning classifier system (LCS) or a genetic programming boolean concept learner, must maintain a population of diverse rules that together solve a problem (e.g., classify examples). To maintain "cooperative diversity" while applying a selection operator to the population of rules, as in the *Michigan-style* LCS, the evolutionary algorithm must incorporate some kind of *niching* mechanism. The natural way to accomplish niching in an LCS is to force competing rules to *share* resources (i.e., rewards). The *implicit* or "natural" niching and speciation induced by such resource sharing, is shown to be robust in the face of severe selective pressure, low population sizes, and overlapping rule coverage. Specifically in this paper we analyze the two-niche (two competing/cooperating rules) case. We find closed form approximations for niche maintenance and niche convergence times, giving us the beginnings of a first predictive model for interacting (cooperating) rules in an evolving population. Finally, we make the case for niching/speciation as a basic, indirect form of cooperation that is fundamental to, and underlying, all other types of more direct cooperation, and which the LCS must therefore promote. Although we focus on the LCS as an example of a specific and well-known evolutionary classifier, all of our results are general enough to apply to any evolutionary algorithm, such as genetic programming (GP), that applies selection to a population of diverse classifiers.

1 Introduction

The unique power of the learning classifier system, or LCS (see (Holland, 1971, 1975, 1985, 1992; Booker, 1982; Goldberg, 1983, 1989a; Wilson, 1986; Wilson & Goldberg, 1989)), is its ability to discover, through artificial genetic search, a *group* of cooperating rules that together represent a concept, or implement a set of behaviors, that solve the problem at hand. In the case of the so-called *Michigan-style* LCS, which is the subject algorithm of the following analysis, the search for cooperative rule sets takes place within a single population of competing and cooperating rules. Thus cooperation and competition occur simultaneously among different rules (individuals) and sets of rules (subpopulations). This model of intertwined, multi-level cooperative and competitive interactions seems natural, based on real-world examples of ecologies, economies, and societies. Yet its potential power of expressing complex concepts and behaviors is offset by the tremendous complexity of its population dynamics.

Although the subject algorithm of our analysis is the Michigan-style LCS, our results are generally applicable to all types of evolutionary (i.e., selection-based) algorithms which adapt a population of diverse classifiers (e.g., rules). Because of our simplifying assumptions, the analysis is not specific to any type of recombination, mutation, or encoding. The classifiers in the population can be string encoded rules (as in an LCS), decision trees (as in GP), fuzzy logic rules, or neural networks. The analysis here assumes only fitness proportionate reproduction, and a fitness assignment that is proportionate to the *reward* (credit) for correct classifications.

In this paper we continue earlier work (Horn, Goldberg, & Deb, 1994) that attempts to lay a foundation for a complete analytical model of cooperative-competitive evolution by isolating what is arguably the lowest but most fundamental type of cooperation: problem decomposition via *niching*. That is, various rules, or *species*, solve different parts of the problem, earning separate, distinct rewards (i.e., *niches*). Rules that compete to

perform the same task (e.g., classify the same examples) compete for the same rewards, while rules that do not overlap in their tasks/rewards are cooperating, in an indirect manner. As we shall show, such indirect or implicit cooperation is far from weak in its ability to survive strong selective pressure for many generations.

In this paper we focus exclusively on one particular method of niching: reward *sharing*. The *implicit niching* induced by the natural and simple method of dividing up reward/credit/fitness among all individuals who have earned it, is a powerful promoter of cooperation. It is capable of maintaining cooperative (i.e., non-competing) rule sets even under constant and rigorous application of genetic algorithm selection and search.

We first describe briefly the existing work on the use and analysis of sharing in the LCS. We then proceed to make a strong case for the critical need for sharing-induced niching in the Michigan-style LCS. Finally, we expand our understanding of the nature, capabilities, and limitations of such niching-induced cooperation by extending the results of Horn, Goldberg, and Deb (1994). Specifically we perform a critical timing analysis of niche convergence and maintenance. The ultimate goal of this line of research is a complete predictive model of niching in evolutionary computation systems. The results to date, and the expected model, are generally applicable to all types of evolutionary computation where the goal is successful search for emergent cooperation via selective competition.

2 Background

In this section we briefly review the nature of niching, the need for niching in an LCS, existing work on niched GAs, and some previous work using niching in classifier systems.

2.1 The Need for Niching in the LCS

In a simple GA (or GP) 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. Such evolution has gone by the name of *co-evolution*, *coadaptation*, and *context-dependent function optimization*. Co-evolution is more natural, more realistic, potentially more powerful, and certainly more complex than the convergence of a simple GA. In the LCS, we ask the GA to search through the space of all possible rules to find and maintain a diverse, cooperative subpopulation.

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).

2.2 Previous work

The important role of niching in evolutionary classification systems has been recognized by many researchers. Perhaps the earliest and most consistent advocate of the critical role of some type of niching in an LCS was Booker (1982, 1989). He implemented niching using an indirect form of sharing based on partial match scores of individuals with rewards, and introduced mating restrictions to limit mixing between niches. Wilson also demonstrated the ability of niching, via sharing of reward/credit, to maintain diverse disjuncts in a population representing a disjunctive boolean concept (Wilson, 1986) or a survival strategy (Wilson, 1987). In his more recent work (Wilson, 1994, 1995), he simplifies the LCS algorithm and isolates key dynamics, such as niching.

Smith and Valenzuela-Rendón (1989) identified the general need for niching in the LCS, but rather than use the *implicit niching*, via reward sharing, suggested by Booker, Wilson, and others, they applied *explicit niching*, via *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 apparently ran 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}. We conjecture that implicit niching using reward sharing should avoid this particular type of separation problem, but this remains

to be shown. Mahfoud (1995a, 1995b) has recently had success also using explicit niching for GA-based classification of boolean concepts, although he does not call his algorithm an LCS. There is some evidence in his results that the separation problem crops up, but again this remains to be shown.

Perhaps the clearest demonstration of explicit niching for GA-based disjunctive concept learning is by McCallum and Spackman (1990). Their "sharing examples" (or "splitting examples") method is identical to the natural, simple method of equaling sharing reward/credit that is the subject of our analysis. They demonstrate stable steady state populations under GA selection. Unfortunately, Neri and Saitta (1995) appear to have misinterpreted McCallum and Spackman's description. Neri and Saitta show that their interpretation of splitting examples does not arrive at a diverse steady state but rather converges to a uniform population. However, Neri and Saitta go on to show implicit niching, correctly interpreted from (Horn, Goldberg, & Deb, 1994) does maintain a stable, diverse steady-state population of concept disjuncts. They go on to show how their *universal suffrage* niching method also maintains the desirable steady-state distributed concept representation. Furthermore, their analysis introduces some new techniques for modeling niching.

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 The Problem at Hand

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.

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., finite rewards). 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).

3.1 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 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[1], 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.

3.2 What's Left? A Hard Problem: Covering

After all the above simplifications and assumptions, it is essential that we verify that we have 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. We are trying to cover the positive examples with a small set of accurate rules. Figure 1 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 1, 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

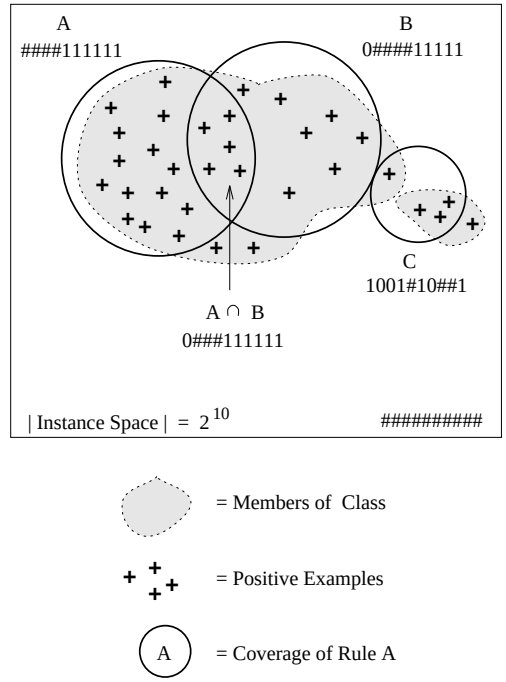

Figure 1: Binary classification as a covering problem. The task of the LCS is to find classifiers (rules, represented as circles) to cover the examples (positive only) of the unknown concept (single class, represented as shaded regions).

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 1 shows, rules can vary in specificity (coverage of instance space) as well as in number of examples[2] 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 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

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

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

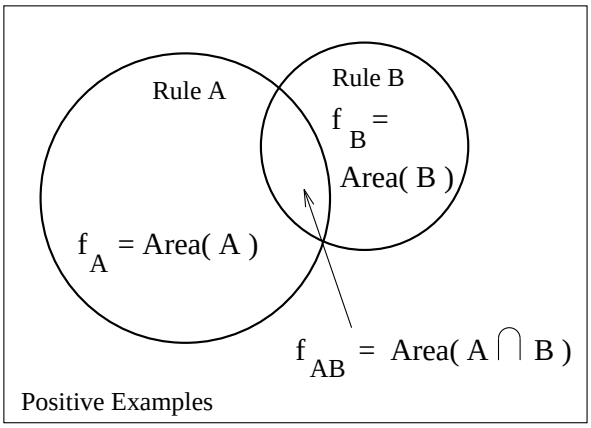

Figure 2: By assuming rules of equal specificity (number of #'s), we can use diameter in the space of examples to indicate a rule's accuracy as "coverage of *examples*". We can then visualize our definitions for the objective fitness of rules, f_A and f_B, and the "fitness" of the overlap in rule coverage, f_{AB} as areas. (Fitness might be measured as number of examples covered.)

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 1, 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 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.

3.3 Definitions

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). We also define the fitness ratio $r_f \equiv \frac{f_A}{f_B}$ and the ratio of overlap $r_o \equiv \frac{f_{AB}}{f_A}$. Figure 2 illustrates some of these definitions. Note that $0 \leq r_o \leq 1/r_f$. (For example, if 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.)

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**:

$$f_{sh,A} = \frac{f_A - f_{AB}}{n_A} + \frac{f_{AB}}{n_A + n_B}. \tag{1}$$

Similarly for rule **B**,

$$f_{sh,B} = \frac{f_B - f_{AB}}{n_B} + \frac{f_{AB}}{n_A + n_B}. \tag{2}$$

4 Time Scale Analysis of Niching

In this section we look for two bounds on the expected times of niche maintenance for the two niche case. Specifically, we first determine the expected steady-state equilibrium point (population distribution) for two distinct, but possibly overlapping niches (rules) **A** and **B**. We then seek the expected time until *niche extinction*: the time until one or the other species is eliminated from the population (and hence cannot return). We then attempt to determine the *niche convergence time*: the expected number of generations to reach steady-state. 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?

4.1 Steady-states and equilibrium points

For niched GAs that use any kind of *resource sharing*, 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:

$$f_{sh,A} = f_{sh,B}. \tag{3}$$

Substituting the formulae for shared fitness in an LCS, Equations 1 and 2,

$$\frac{f_A - f_{AB}}{n_A} + \frac{f_{AB}}{n_A + n_B} = \frac{f_B - f_{AB}}{n_B} + \frac{f_{AB}}{n_A + n_B}. \tag{4}$$

Solving for $\frac{n_B}{n_A}$,

$$\frac{n_B}{n_A} = \frac{f_B - f_{AB}}{f_A - f_{AB}} = \frac{\frac{f_B}{f_A} - \frac{f_{AB}}{f_A}}{1 - \frac{f_{AB}}{f_A}}. \tag{5}$$

Substituting our ratios r_f and r_o, rearranging, and naming the result $r_{eq,n} \equiv \frac{n_B}{n_A}$ (at equilibrium),

$$r_{eq,n} = \frac{\frac{1}{r_f} - r_o}{1 - r_o}. \tag{6}$$

We see that the equilibrium point changes with overlap. With no overlap, that is when $r_o = 0$, then $r_{eq,n} = 1/r_f$. While at complete overlap ($r_o = 1/r_f$) the equilibrium point is 0 (i.e., the population will converge to a uniform one, consisting of all **A**s).

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

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

Solving for P_A and calling it $P_{A,eq}$:

$$P_{A,eq} = \frac{1 - r_o}{1 - 2r_o + \frac{1}{r_f}}. \tag{7}$$

We note 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., $p_A = P_A$).

4.2 Rule maintenance times

In (Horn, Goldberg, & Deb, 1994), the expected time to loss of one of the two niches (i.e., classes of individuals) was calculated exactly by a Markov chain analysis, just as Horn (1993) calculated for *fitness sharing* in a niched simple GA . This quantity is simply the expected time to absorption by either of the two "converged" states (i.e., uniform populations of one or the other class of individual). In this subsection we review their exact results from Markov chain models, and then propose an approximate closed form expression.

4.2.1 Exact Model

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). A random initial population, which has little chance of being in either absorbing state, is assumed. Horn, et. al. (1994) found that the expected absorption time grows exponentially with population size N. The exponent in the growth decreases as r_f moves away from one, just as in the case of the niched GA (Horn, 1993). Finally, holding r_f constant, they found that the exponent of growth (in N) also decreases with increasing niche overlap, just as in the niched GA (Horn, 1993)[3].

The time to absorption (t_{abs}) is also called *niche* (or *rule*) *maintenance* time or niche *extinction*, *failure* or *loss* time (e.g., Mahfoud, 1995a, 1995b). Here we look at how t_{abs} varies with overlap. That is, we vary $0 < r_o \leq 1/r_f$ while holding r_f constant. In Figure 3 we show the growth in expected absorption time as a function of N for $r_o = 0.0$, $r_o = 0.1$, $r_o = 0.3$, and $r_o = 0.5$, while $r_f = 2$ (the solid dots represent the output of the exact (Markov) models).

This growth (Figure 3) appears to be exponential. Indeed, for the case of "perfect sharing" (Horn, 1993), when $r_o = 0$, we can find the exact exponent.

Let p_A be the probability of selecting a single individual copy of rule **A** for reproduction from a particular population with n_A copies of rule **A** and n_B copies of rule **B** (with $n_A + n_B = N$). Under proportionate selection, $p_A = \frac{n_A f_{sh,A}}{n_A f_{sh,A} + n_B f_{sh,B}}$, where $f_{sh,A}$ and $f_{sh,B}$ are the *shared* fitnesses of rules **A** and **B** respectively, in the current population. Substituting in the shared fitnesses $f_{sh,A}$ and $f_{sh,B}$ from Equations 1 and 2, and rearranging, yields

$$p_A = \frac{f_A - f_{AB} + f_{AB}\frac{n_A}{N}}{f_A - f_{AB} + f_{AB}\frac{n_A}{N} + f_B - f_{AB} + f_{AB}\frac{n_B}{N}}. \tag{8}$$

Dividing numerator and denominator by f_A, the above becomes

$$p_A = \frac{1 - \frac{f_{AB}}{f_A} + \frac{f_{AB}}{f_A}\frac{n_A}{N}}{1 - \frac{f_{AB}}{f_A} + \frac{f_{AB}}{f_A}\frac{n_A}{N} + \frac{f_B}{f_A} - \frac{f_{AB}}{f_A} + \frac{f_{AB}}{f_A}\frac{n_B}{N}}. \tag{9}$$

Remembering our defined ratios $r_f = f_A/f_B$ for fitness, and $r_o = f_{AB}/f_A$ for overlap, we find that

$$p_A = \frac{1 + r_o(\frac{n_A}{N} - 1)}{1 - r_o + r_o\frac{n_A}{N} + \frac{1}{r_f} - r_o + r_o\frac{n_B}{N}}. \tag{10}$$

Finally, simplification yields

$$p_A = \frac{1 - r_o + r_o\frac{n_A}{N}}{1 - r_o + \frac{1}{r_f}}. \tag{11}$$

With no overlap ($r_o = 0$), the probability of choosing **A** is constant: $p_A = \frac{1}{1+1/r_f} = \frac{r_f}{r_f+1}$. Therefore in the Markov chain model for no overlap, the transition probabilities for all transient states are identical. It follows that prior to absorption we always have the same probability of absorption, namely $p_{absorb} = (p_A)^N + (p_B)^N$, where $(p_A)^N$ is simply the probability of choosing N **A**s given that p_A is the probability of choosing a single copy of **A** (under proportionate selection). With no overlap,

[3]Note that the case of no overlap ($r_o = 0$) in implicit niching is exactly the same as for the niched GA with no overlap (i.e., "perfect sharing" (Horn, 1993)).

$(p_A)^N = (\frac{r_f}{r_f+1})^N$. Similarly, $(p_B)^N = (\frac{1}{r_f+1})^N$, when $r_o = 0$. The expected time to absorption, assuming that the chain is started in a transient state, is the inverse of the probability of absorption, $\frac{1}{(p_A)^N+(p_B)^N}$:

$$E[t_{abs}] = \frac{1}{(\frac{1}{r_f+1})^N + (\frac{r_f}{r_f+1})^N}. \quad (12)$$

After some rearranging:

$$E[t_{abs}] = \frac{(r_f+1)^N}{r_f^N+1}. \quad (13)$$

The expected absorption time above grows exponentially in N for any $r_f > 0$. In particular, when $r_f = 1$, the growth is $O(2^N)$.

When $r_o \neq 0$ (i.e., niche overlap), it might be impossible to calculate a closed form expression for the *exact* expected absorption time. However, in Figure 3 it certainly appears that all such growth is exponential, at least up until $r_o = 1/r_f$, at which time the growth should become logarithmic (linear) in N, since this is the case of selection (drift)[4].

4.2.2 Closed Form Approximation

A closed form expression for $E[t_{abs}]$ with overlap ($r_o > 0$) is desirable for our timing analysis however, and we define an approximate one here. We use the approach above (for $r_o = 0$) 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

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

where p_A and p_B are the probabilities of selecting **A** and **B** respectively, from the current population under proportionate selection. The above is simply the expected waiting time until selection chooses N copies of **A** or N copies of **B**. Our assumption above translates here into the assumption that p_A is constant, at or near equilibrium, so that we can just substitute $p_{A,eq}$ from Equation 7 above:

$$E[t_{abs}] = \frac{1}{(\frac{1-r_o}{1-2r_o+\frac{1}{r_f}})^N + (1-\frac{1-r_o}{1-2r_o+\frac{1}{r_f}})^N}, \quad (14)$$

which does not simplify much.

[4] It has been suggested (Horn, 1993) that exponential growth in niche maintenance 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 LCS implicit niching clearly qualifies.

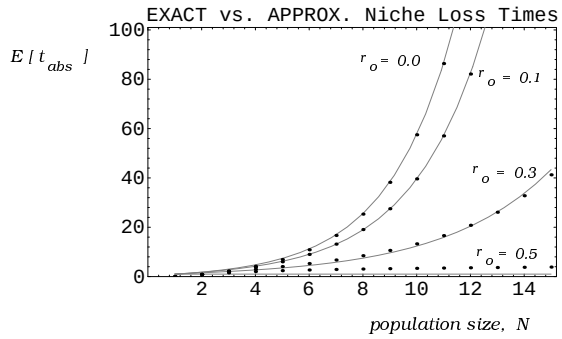

Figure 3: A comparison of *exact* expected niche loss times to the approximated times, as a function of population size. The exact results (from the Markov models) are shown as solid dots. The approximations, from our closed form expressed model, are shown as dashed lines. The plots indicate general agreement for small niche overlap (r_o). For all plots shown $r_f = 2$.

To test our closed form approximation, we plot both the exact results (from the Markov chain model) and the approximations in Figure 3. In Figure 3, the solid dots are the plotted points for the exact (Markov) model, while the dashed lines are defined by Equation 14. In general, we see close agreement over the range plotted, except for high overlap (e.g., complete overlap when $r_o = 1/r_f = 0.5$). As expected, our simple closed form expression modeling absorption time loses accuracy as for high overlap cases. Intuitively, with high overlap the variance of the steady-state distribution will be high, and the population will not spend as much near the steady-state. Since the GA is spending less time at steady-state, the use of the steady-state absorption probability as an estimate of the overall absorption probability introduces increasing error, as overlap increases.

4.3 Steady-state Convergence Times

Having found an upper bound on expected lifetimes of niches (rules), 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., $r_f = 0$), the expected convergence

time grows linearly in N. Can we expect similarly quick convergence to "niching equilibrium"?

To answer such questions, we use the simple, well-known method of *expected proportion equations* to model niche convergence. This method of tracking only the expected next generation *population*, 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), 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.

4.3.1 Niching Convergence Behavior

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 **A**, then $E[P_{A,t+1}] = p_A$, where p_A is calculated at time (generation) t. Substituting for p_A from Equation 11 above, 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 if we make the major assumption that $E[P_{A,t+1}] \approx P_{A,t+1}$ (which is a strict equality only for deterministic selection operators or for infinite populations (Vose, 1993)) then the above is 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:

$$P_{A,t} = \frac{1 - r_o}{1 - 2r_o + \frac{1}{r_f}} + \left(P_{A,0} - \frac{1 - r_o}{1 - 2r_o + \frac{1}{r_f}}\right)\left(\frac{r_o}{\frac{1}{r_f} - r_o + 1}\right)^t. \quad (15)$$

Remembering that at equilibrium (steady-state) $P_{A,eq} = \frac{1-r_o}{1-2r_o+\frac{1}{r_f}}$ (Equation 7), 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

$$P_{A,t} = P_{A,eq}(1 - \beta^t) + P_{A,0}\beta^t. \quad (16)$$

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 16 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 4 illustrates a typical convergence graph. With fitness ratio $r_f = 2$ and overlap ratio $r_o = 1/3$, the equilibrium proportion $P_{A,eq} = \frac{1-r_o}{1-2r_o+\frac{1}{r_f}} = 4/5 = 0.8$. In Figure 4 we see very fast convergence to this equilibrium.

To see the effect of varying the degree of overlap, Figure 4 shows convergence curves for a number of different r_o values. The lowest curve corresponds to $r_o = 1/10$, very little overlap, in which we expect to see almost instantaneous (i.e., one generation) convergence to steady-state ($P_{A,eq}$). The other curves show how increasing overlap increases the convergence time but also changes the equilibrium point. At maximum overlap, $r_o = 1/r_f = 1/2$, 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 16 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 16) 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.

4.3.2 Convergence Times

Here we derive a useful expression for (two-)niche convergence times by solving Equation 16 for time t (in generations). Rearranging Equation 16, 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:

$$t = \frac{Log[\frac{P_{A,eq}-P_{A,t}}{P_{A,eq}-P_{A,0}}]}{Log[\beta]}. \quad (17)$$

In general $P_{A,t}$ approaches $P_{A,eq}$ asymptotically with t. More practically, we introduce the discrete nature of a

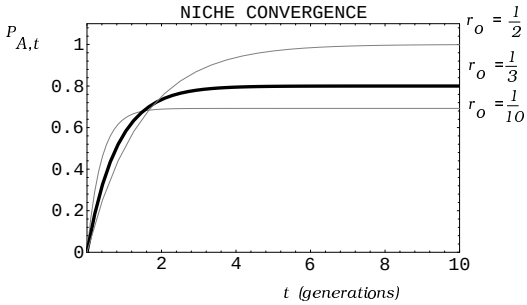

Figure 4: Expected convergence with varying overlap: $r_o = 1/10$, $r_o = 1/3$, and $r_o = 1/2$ (complete overlap). Fitness ($r_f = 2$) is the same for all plots.

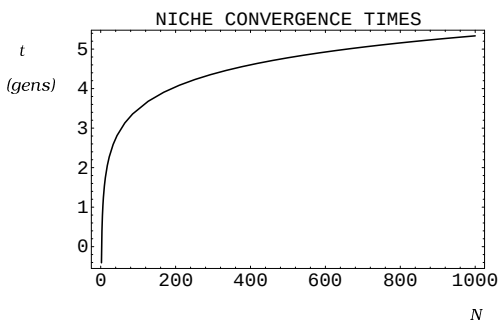

Figure 5: Expected niche convergence time grows logarithmically in population size.

finite population of size N, by asking for the time it takes to go from one copy of **A** ($P_{A,0} = 1/N$) to within at most one individual ($1/N$) of the equilibrium population ($P_{A,t} = P_{A,eq} - 1/N$):

$$t = \frac{Log[\frac{1/N}{P_{A,eq} - 1/N}]}{Log[\beta]}.$$

Simplifying yields:

$$t = \frac{-Log[P_{A,eq}N - 1]}{Log[\beta]}. \quad (18)$$

We can see immediately that expected convergence time grows logarithmically in population size N, as we suspected. Figure 5 illustrates a typical expected growth in convergence time with increasing N. Here $r_f = 2$ and $r_o = 1/3$ (remember that β is a function of r_f and r_o).

4.4 Putting It All Together: Critical Population Sizes

In this subsection we make an initial attempt to bring the above two timing results together in a rather crude fashion. We hope to refine and make rigorous some *niching control maps* in future.

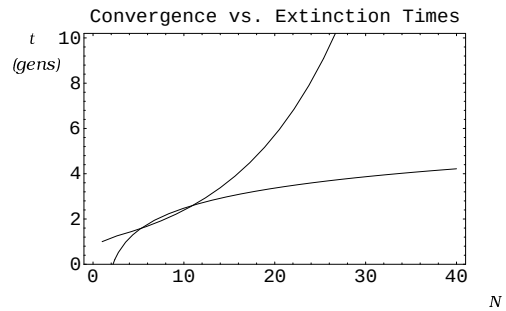

Figure 6: Expected niche extinction times (upper curve) versus "expected" niche convergence times (lower curve). Fitness ratio $r_f = 2$ with very high overlap $r_o = 0.45$ (near maximum).

In Figure 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 implicit niching, in a learning classifier system or in any GA or GP with distinct and finite resources/rewards, works well *within bounds*. And apparently we have some hope of defining those boundaries.

5 Limitations

Before discussing the implications of the above results, we address the limitations on our analysis imposed by our simplifications and assumptions described in Section 3.1. Again, briefly, those assumptions are (1) a ternary alphabet $\{0, 1, \#\}$, (2) stimulus-response (S-R) classifiers, (3) binary classification task, and (4) equal specificity of all rules.

Ternary alphabets are common in both practical and theoretical LCS work, and are equal in expressive power to higher cardinality alphabets. So the assumption of a ternary alphabet is not a simplification and hence does

not limit the applicability of our results.

The restriction to S-R classifiers is a real simplification, however, and does place some limits on the extension of our results. Still, the S-R LCS is quite powerful (Wilson, 1994). We could argue that before adding the complexities of message-passing, we should first understand and master the power of the S-R LCS. We could also look at the additional rule interactions possible with message-passing in terms of their effects on the steady-states of the S-R LCS, thus extending our results to include non-S-R classifiers and strong cooperation.

Our assumption of binary classification is a simplification that allowed us to graphically depict rule coverage and overlap, but which actually has no effect on the mathematical analysis in this paper. The concepts of rule coverage and overlap are just as valid in the case of k-ary classification, but they can no longer be measured and ordered simply by numbers of examples.

The fourth assumption is the only one that places any significant limitations on the applicability of our analytical results. Specifically, LCS implicit fitness sharing describes the effects of rules on the "coverage" component of each other's fitness function (where coverage is based on the number of correct classifications). Rules of the same order of specificity can vary only in this component, and so implicit fitness sharing captures the entirety of rule interaction within a single level of generality. Rules of different orders, however, can vary in a second component of fitness: accuracy (which is based on the number of errors). But coverage and accuracy are conflicting objectives, with no general way to combine them into a single scalar fitness measure. So the extension of our results to include rules of all orders of specificity will depend on the user-specified combination of the two fitness components. However, the effect of a more general rule on a more specific rule, with overlapping coverage, will probably be some kind of sharing of the overlapped resources.

6 Conclusions

The specific focus of this paper was on niching in the LCS. But the results are generally applicable to any evolutionary algorithm maintaining useful diversity under selective pressure on a task with separate rewards/resources.

It has long been known that some kind of natural and implicit speciation takes place in the LCS if we break tied competitions for resources arbitrarily. In (Horn, Goldberg, & Deb, 1994), our goal was to demonstrate a meaningful LCS *dynamic equilibrium point*. By meaningful we mean one that will be reached quickly (i.e., fast convergence rate and high *restorative pressure*), and last long (i.e., long niche extinction times). In 1994, we showed the ability of the LCS to balance selection pressure with restorative force, and thus maintain high quality, diverse niches virtually indefinitely. This balance in turn allows us to apply the GA more vigorously in the LCS. We can then hope to improve LCS rule discovery. But for successful exploration and exploitation of niches, the LCS must have a large enough population to maintain subpopulations at all desirable niches.

In this study we extended our model of implicit niching by finding a closed form *approximation* for expected absorption times in the Markov chain models, and also by deriving a closed form estimation of niching convergence rates using the expected proportion models. We now have upper and lower bounds, however crude, on expected "niche existence" times, as functions of population size, fitness, and niche overlap. These are descriptive models for the moment, but by carefully adjusting population size, scaling fitness, and by introducing a tunable parameter for tolerating niche overlap, we can eventually use the models to *prescribe* proper parameter settings to maintain exactly the cooperative population distributions we desire. That is, we should be able to control selection's distinction between cooperation and competition, and the inherent tradeoffs of accuracy (objective fitness) versus generality versus coverage (overlap), etc.

We hope we have demonstrated that niching in an LCS is absolutely necessary to any kind of cooperation. By definition a cooperative group of rules is a set of *diverse* rules, and without some kind of niching the GA in the LCS will not maintain diversity. If we want serious GA search in our LCS, we must have significant selection pressure. If we want to maintain cooperative rules, we must balance that selection pressure with a restorative pressure. If we can't do that, we won't be able to induce any type of *strong cooperation*, such as that necessary for temporal rule chains and default hierarchies. All such strong cooperation implies successful weak cooperation (i.e., covering). Because of such entailment, niching is fundamental to the success of rule search *and* maintenance in the LCS, and perhaps to any cooperative evolutionary algorithm.

The most important point of this paper, however, is the demonstration that the dynamics of an interacting, coevolving, cooperative/competive population *can* be analyzed, described, predicted, and controlled.

Acknowledgments

The first author is grateful for early support provided by NASA under contract number NGT-50873. Both authors acknowledge the support of the Air Force Office of Scientific Research (AFOSR), Air Force Materiel Command, USAF, under grants numbered F4960-94-1-0103 and F49620-95-1-0338. 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.

Bibliography

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. Pages 265–274.

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)

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 (ICGA 3).* San Mateo, CA: Morgan Kaufmann. Pages 42–50.

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. 1989. *Genetic algorithms in search, optimization, and machine learning.* Reading, MA: Addison-Wesley.

Goldberg, D. E., & Deb, K. 1991. A comparative analysis of selection schemes used in genetic algorithms. In G. J. E. Rawlins, (Ed.), *Foundations of Genetic Algorithms (FOGA).* San Mateo, CA: Morgan Kaufmann. Pages 69–93.

Goldberg, D. E., Deb, K., & Horn, J. 1992. Massive multimodality, deception, and genetic algorithms. In R. Männer & B. Manderick, (Eds.), *Parallel Problem Solving From Nature, 2.* North-Holland. Pages 37–46.

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 (ICGA 2).* Hillsdale, NJ: Lawrence Erlbaum Associates. Pages 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 (ICGA 2).* Hillsdale, NJ: Lawrence Erlbaum Associates. Pages 1–8.

Holland, J. H. 1971. Processing and processors for schemata. *Proceedings of the Symposium on Associative Information Techniques.* Pages 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. Pages 1–7.

Holland, J. H. 1992. *Adaptation in natural and artificial systems.* Cambridge, MA: MIT Press.

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 (ICGA 5).* San Mateo, CA: Morgan Kaufmann. Pages 110–117.

Horn, J., Goldberg, D. E., & Deb, K. 1994. Implicit Niching in a Learning Classifier System: Nature's Way. *Evolutionary Computation, 2*(1). Pages 37–66.

Mahfoud, S. W. 1992. Crowding and preselection revisited. In R. Männer & B. Manderick (Eds.), *Parallel Problem Solving From Nature, 2.* North-Holland. Pages 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 (ICGA 5). San Mateo, CA: Morgan Kaufmann, Page 643. Full paper available as IlliGAL Report No. 93001. Urbana: Department of General Engineering, University of Illinois at Urbana-Champaign.

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. Pages 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.

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. Pages 149–152.

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. Pages 32–39.

Smith, R. E., Forrest, S., & Perelson, A. S. 1993. Searching for diverse, cooperative subpopulations with genetic algorithms. *Evolutionary Computation, 1*(2). Pages 127–149.

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 (ICGA 3).* San Mateo: Morgan Kaufmann. Pages 340–346.

Vose, M. D. 1993. Modeling simple genetic algorithms. In L. D. Whitley, (Ed.), *Foundations of Genetic Algorithms, 2 (FOGA 2).* San Mateo, CA: Morgan Kaufmann. Pages 63–73.

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.* Pages 199–228.

Wilson, S. W. 1994. ZCS: A zeroth level classifier system. *Evolutionary Computation, 2*(1). Pages 1–18.

Wilson, S. W. 1995. A classifier system based on accuracy. *Evolutionary Computation, 3*(2). Pages 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. Pages 244–255.