Implicit Niching in a Learning Classifier System: Nature's Way

Jeffrey Horn, David E. Goldberg, and Kalyanmoy Deb
Illinois Genetic Algorithms Laboratory
University of Illinois at Urbana-Champaign
Urbana, IL 61801-2996

IlliGAL Report No. 94001
March 1994

To appear in *Evolutionary Computation, 2(1)*

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

Implicit Niching in a Learning Classifier System: Nature's Way

Jeffrey Horn
Department of Computer Science
University of Illinois at Urbana-Champaign
117 Transportation Building
104 South Mathews Avenue
Urbana, IL 61801-2996
Phone: 217/333-2346
jeffhorn@uiuc.edu

David E. Goldberg
Department of General Engineering
University of Illinois at Urbana-Champaign
117 Transportation Building
104 South Mathews Avenue
Urbana, IL 61801-2996
Phone: 217/333-0897
deg@uiuc.edu

Kalyanmoy Deb
Department of Mechanical Engineering
Indian Institute of Technology
Kanpur, UP, PIN 208016, India
Phone: 91-512-250-151 x 2302
deb@iitk.ernet.in

Abstract

We approach the difficult task of analyzing the complex behavior of even the simplest learning classifier system (LCS) by isolating one crucial subfunction in the LCS learning algorithm: covering through niching. The LCS must maintain a population of diverse rules that together solve a problem (e.g., classify examples). To maintain a diverse population while applying the GA's selection operator, the LCS 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). This *implicit LCS fitness sharing* is similar to the explicit fitness sharing used in many niched GAs. Indeed, the LCS implicit sharing algorithm can be mapped onto explicit fitness sharing with a one-to-one correspondence between algorithm components. This mapping is important because several studies of explicit fitness sharing, and of niching in GAs generally, have produced key insights and analytical tools for understanding the interaction of the niching and selection forces. We can now bring those results to bear in understanding the fundamental type of cooperation (a.k.a. *weak* cooperation) that an LCS must promote.

Keywords: genetic algorithms, classifier systems, cooperation, weak cooperation, diversity, coadaptive systems, co-evolution, niching, speciation, fitness sharing, resource sharing, set covering, Markov chain, functional decomposition, restorative pressure, equilibrium.

1 Introduction

The learning classifier system[1] (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 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 *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.

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 first 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).

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 *explicit 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 the *niched GA* to the LCS and come to a better understanding of this one critical subfunction of the overall LCS.

The major goal of this paper is to establish a direct correspondence between the natural, intuitive niching in the LCS and the more explicit, but better understood, niching in a GA with fitness sharing. This link facilitates a two-way flow of analytical tools and results. Niched GAs can benefit from knowledge of the shape of the LCS sharing function, and from insights into how the LCS "naturally" handles niche

[1] See (Holland, 1971, 1975, 1985, 1992; Booker, 1982; Goldberg, 1983, 1989; Wilson, 1987; Wilson & Goldberg, 1989).

boundaries. For the LCS our understanding of a niched GA's steady-state behavior, including measures of niche maintenance times, niche subpopulation sizes, and equilibrium points and their stability, all carry over immediately. Together, the niche maintenance results mean that the LCS, with appropriate population sizing for niching, can maintain a stable, diverse population of interacting rules essentially forever. This stability under constant GA selection pressure allows the LCS, via the GA, to continue searching for new and better rules while preserving the cooperative ecology of rules already discovered. Thus we can with confidence apply the GA to LCS exploration as vigorously as we apply it to simple GA search. Greater use of the GA in the LCS might help solve the well-known problem of *rule discovery*.

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 Niched GAs

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. DeJong's *crowding* (DeJong, 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; Deb & Goldberg, 1989; Horn, 1993;Mahfoud, 1993), and challenged by a massively multimodal problem in (Goldberg, Deb, & Horn, 1992).

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 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* $m_i = \sum Sh()$ 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

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

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 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

$$m_i = \sum_{j=1}^{N} Sh(d_{i,j}), \quad (2)$$

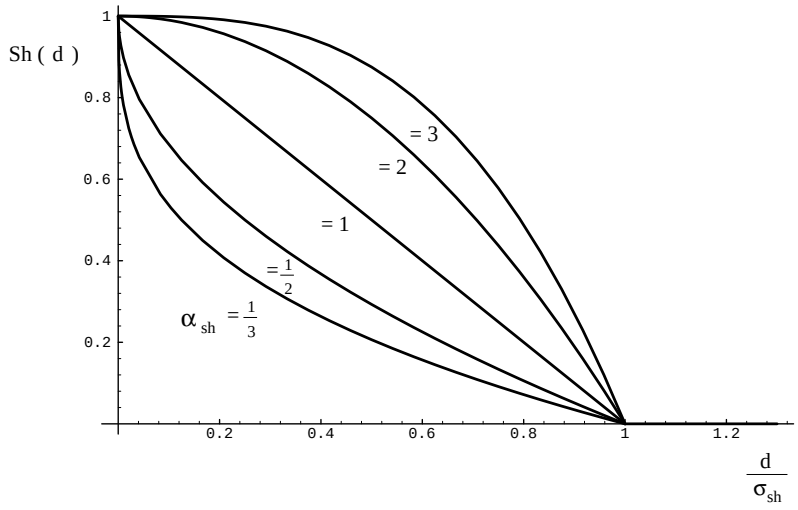

Figure 1: The sharing function, a function of distance d, as α_{sh} varies.

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

$$f_{sh,i} = \frac{f_i}{m_i}. \tag{3}$$

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 on a variety of multimodal functions (Deb, 1989; Deb & Goldberg, 1989).

2.2 The Need for Niching in the LCS

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 the LCS, uniform convergence is undesirable. In multiobjective 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. In a regular GA (i.e., not the GA in the LCS), many 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.

Similarly, 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).

2.3 Previous work

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 LCS (Wilson, 1987). 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. 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[2].

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.

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

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 rules
- 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 rules. The output of each rule 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 rule 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[3], 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: 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. We are trying

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

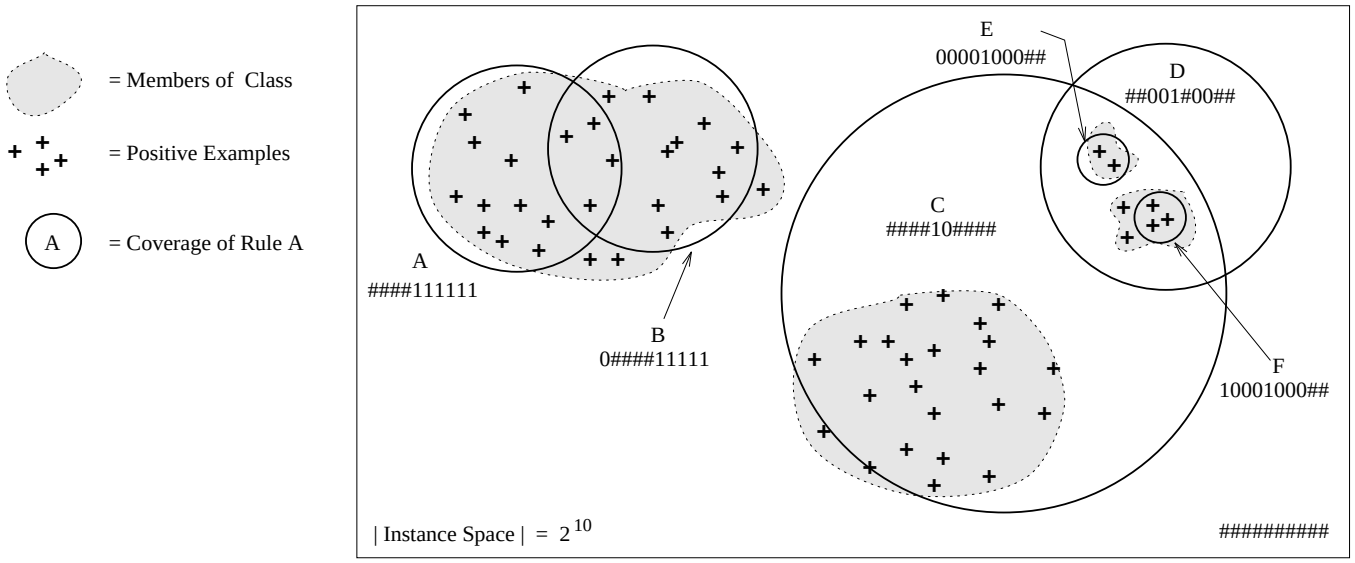

Figure 2: Binary classification as a set covering problem. The task of the LCS is to find classifiers (rules, represented as circles) to cover the examples (positive only) of the unknown concept (single class, represented as shaded regions).

to cover the positive examples with a small set of accurate rules. Figure 2 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 2, 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 2 shows, rules can vary in specificity (coverage of instance space) as well as in number of examples[4] 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 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 2, 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, the large rectangle represents the space of all *examples* given to

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

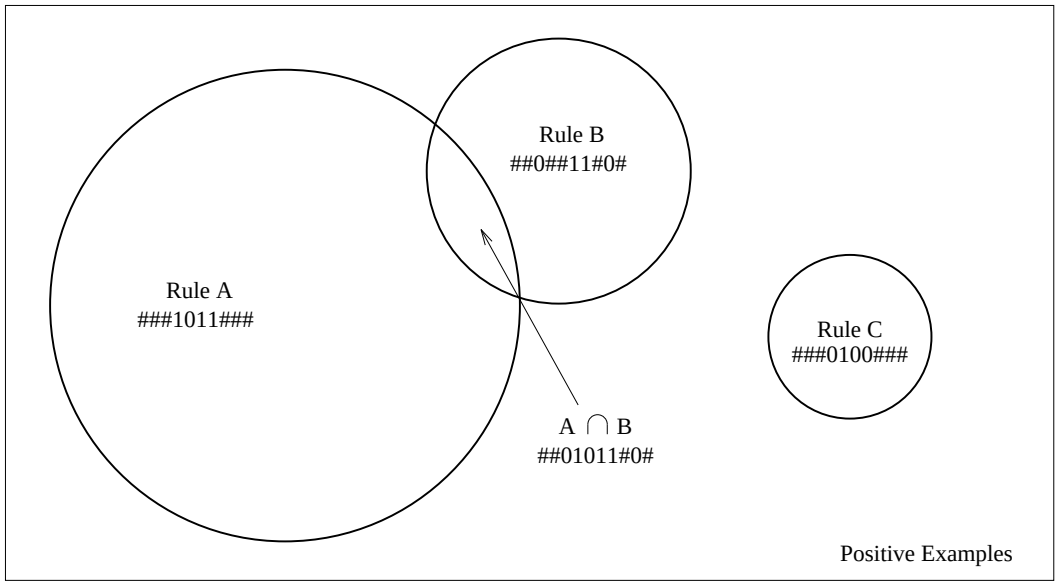

Figure 3: By assuming rules of equal specificity (number of #'s), we can use diameter in the space of examples to indicate a rule's accuracy as "coverage of examples".

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 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 five bit problem would cover all examples with zeroes for the third and fourth attributes and a one for the last attribute, just as the schema ##00#1 "contains" all strings with zeroes in bit positions three and four, and a one in position five. We are choosing, from the set of 3^ℓ possible schemata, a minimal set of highly fit schemata. Figure 4 illustrates a portion of an example search space.

In Figure 4, 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 instances 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

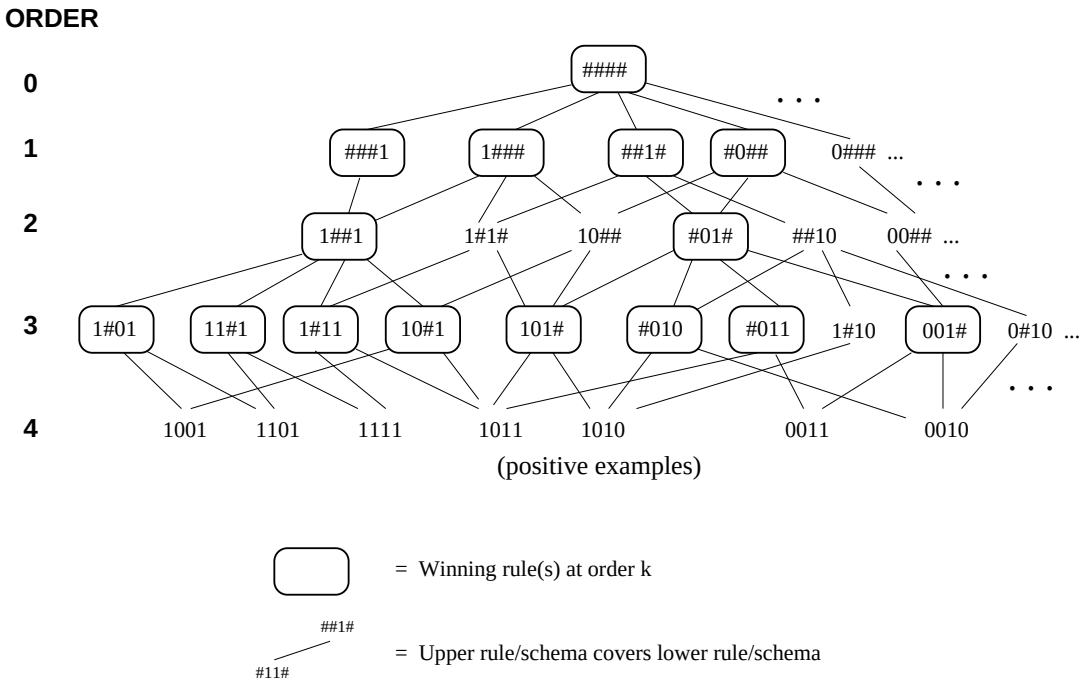

Figure 4: The LCS binary classification problem as a search through schema space. Each rule is a schema, or hyperplane in the instance space. Instances, such as examples, are schemata of order ℓ. Here, $\ell = 4$.

coverage (of instances) of the rule can be calculated from the 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 a schema 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[5]. A GA is thought to search a schema space implicitly, while an LCS clearly searches its schema 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 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 4 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[6]. Note that these two winners at order two overlap in coverage, competing for the example 1011.

[5]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.

[6]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.

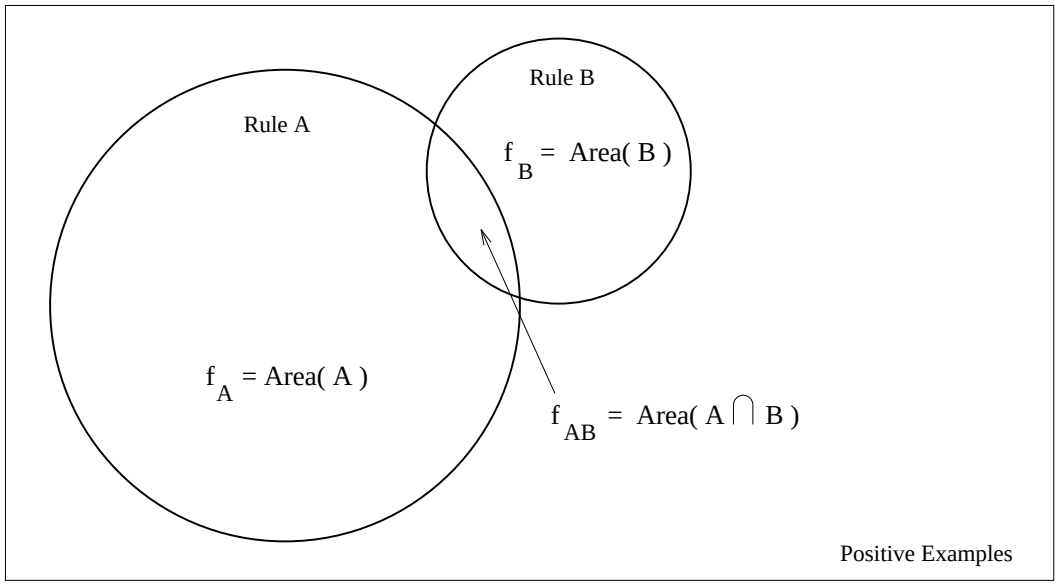

Figure 5: Our definitions for the objective fitness of rules, and the "fitness" of the overlap in rule coverage. Fitness might be measured in number of examples covered. Here area is proportionate to fitness.

3.4 Learning DNF

Finally, we cast the S-R LCS binary classification task as a well-known problem in machine learning: 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 (Wilson, 1987). Each rule represents a *monomial*, or conjunction of attribute settings (e.g., 1##0 maps to $a_1\overline{a_4}$). The population of rules represents a disjunction of such conjunctions (e.g., $a_1\overline{a_4} + \overline{a_2}a_3a_4 + \overline{a_3} + ...$). 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 4 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).

4 Relationship to Fitness Sharing

We now consider the relationship between LCS implicit niching and explicit fitness sharing. We focus on the effect two rules have on each other. In explicit fitness sharing, two nearby individuals degrade each other's fitness according to the distance between them, as we described in the background section. How do two similar rules (i.e., rules with overlapping coverage) affect each other's fitness in the LCS?

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 5 illustrates some of these definitions.

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**. 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**:

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

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

$$f_{sh,A} = \frac{f_A}{m_A}. \tag{5}$$

We can force our expression for shared fitness with LCS sharing, in Equation 4, into a form similar to the right hand side of Equation 5:

$$f_{sh,A} = \frac{f_A}{\frac{f_A n_A (n_A + n_B)}{f_A n_A + f_A n_B - f_{AB} n_B}}. \tag{6}$$

Now we have expressed LCS niching as a degradation by division of the objective fitness, as in explicit fitness sharing. But in its current form, the denominator in Equation 6, call it D, is difficult to relate to the niche count m_A in explicit 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**.

In explicit 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**:

$$\frac{\partial m_A}{\partial n_B} = Sh(d_{A,B}). \tag{7}$$

Turning to LCS implicit niching, we can also take the partial derivative of the denominator D from Equation 6 with respect to n_B:

$$\frac{\partial D}{\partial n_B} = \frac{f_A f_{AB} n_A^2}{(f_{AB} n_B - f_A n_A - f_A n_B)^2}. \tag{8}$$

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

$$\frac{\partial D}{\partial n_B} = \frac{\frac{f_{AB}}{f_A}}{(\frac{f_{AB}}{f_A}\frac{n_B}{n_A} - \frac{n_B}{n_A} - 1)^2}. \tag{9}$$

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

$$\frac{\partial D}{\partial n_B} = \frac{r_o}{(r_o r_n - r_n - 1)^2}. \tag{10}$$

Equation 10 can be considered the contribution to the "niche count" of **A** under LCS niching. This quantity corresponds to the sharing function $Sh(d)$ of explicit fitness sharing. Comparing the expression in Equation 10 to that in Equation 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 explicit 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 explicit fitness sharing, namely an indicator of similarity of purpose in solving the problem at hand.

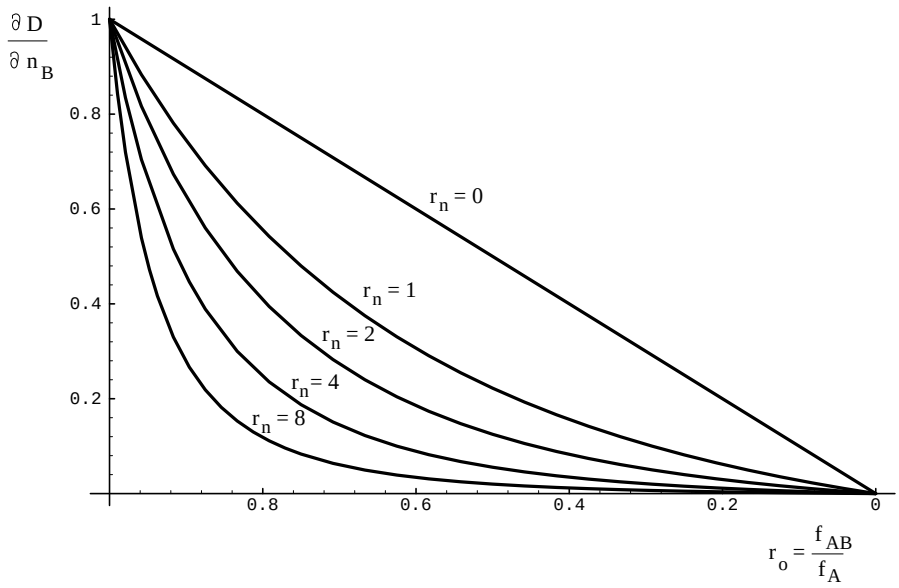

Figure 6: The LCS analog of the sharing function $Sh(d)$, as a function of "distance" r_o, for different population ratios $r_n = \frac{n_B}{n_A}$ With more copies of **A** than of **B** (smaller r_n), the LCS "sharing" function becomes increasingly triangular.

In Figure 6, we plot $\frac{\partial D}{\partial n_B}$ as a function of "distance" r_o for various values of r_n. The curves of Figure 6 are strikingly similar to those of the sharing function in Figure 1, with the sharing exponent $\alpha_{sh} \leq 1$. Specifically, both $Sh(d)$ 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 explicit fitness sharing).

In explicit fitness sharing, however, $Sh(d)$ 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 6 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 $Sh(d)$ 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 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 $Sh(d)$ with increasingly small α_{sh}, where $0 < \alpha_{sh} < 1$.

Figure 7 plots $\frac{\partial D}{\partial n_B}$ as a function of both r_n and r_o. Like explicit fitness sharing, LCS sharing degrades objective fitness according to how much 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 LCS 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 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 explicit 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

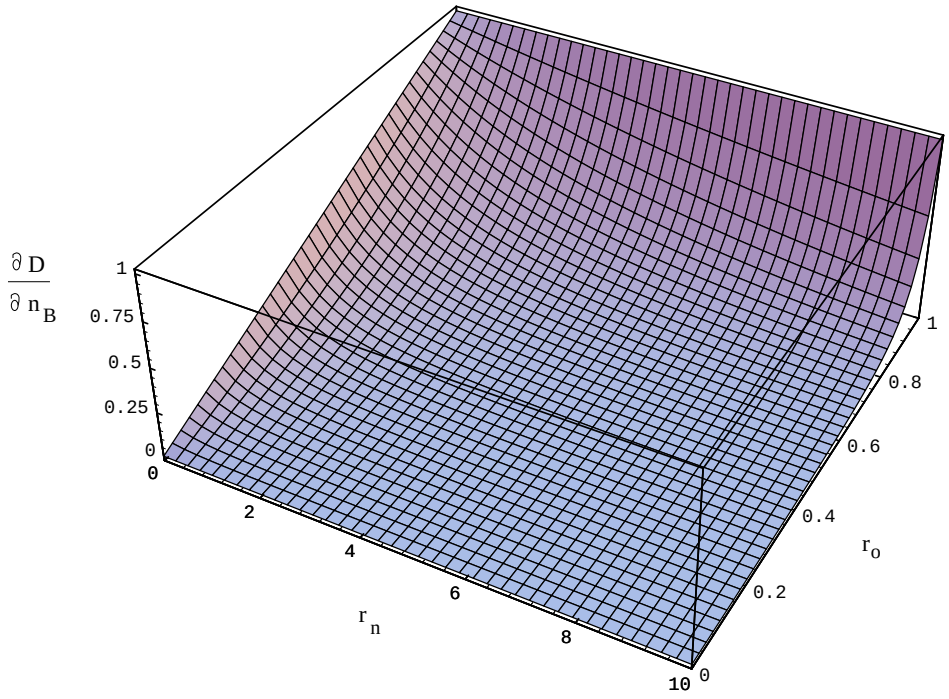

Figure 7: The behavior of the implicit sharing function is summed up in this plot of population ratio $r_n = \frac{n_B}{n_A}$ versus fitness overlap ratio r_o.

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 10, the surface in Figure 7 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 $r_o = 1$.

We can further generalize our results for the behavior of the implicit 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 implicit 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 implicit 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 7, we see that the implicit 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 (Goldberg & Richardson, 1987; Deb, 1989; Deb & Goldberg, 1989; Horn, 1993), the authors have proposed and investigated such stable points, where the "niching force" and selection force are in balance. Explicit 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, is the subject of the next section.

5 Markov Chain Analysis

In this section we bring to bear another analytical approach from the study of simple and niched GAs. Modeling the GA as a Markov chain has been successful in understanding several basic behaviors (De Jong, 1975; Goldberg & Segrest, 1987; Vose, 1992; Nix & Vose, 1992; Davis & Principe, 1991, 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. The work of Goldberg and Segrest was extended separately by Horn (1993) and Mahfoud (1993) to the analysis of niched GAs. Whereas Horn focused on the problem of niche overlap in the case of two niches, Mahfoud looked at multiple, non-overlapping niches. In both of these studies, the primary focus 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.

In this section we review some of the results and techniques from (Goldberg & Segrest, 1987) and (Horn 1993), while adapting and applying them to LCS niching. We can then demonstrate the presence of a similar but unique niching force in LCS implicit niching. We also show that a unique steady-state is induced by LCS niching, allowing us to calculate an equilibrium point as a function of r_f and r_o. This equilibrium point in turn enables us to look at the niching force and the sharing function at or near the point of equilibrium, which by definition is where we *expect* the LCS population to be *most* of the time.

5.1 The model: Markov chain analysis of the simple GA

Goldberg and Segrest (1987) established the intuitive and easily visualized Markov chain model used by them and later by Horn (1993). The original model assumed that there were only two possible kinds or classes of individuals, say **A** and **B**, in a population of size N. Such a limitation allows an intuitive numbering of states. There are $N+1$ possible states i, where i is the population with exactly i copies of **A** and $N-i$ copies of **B**. Goldberg and Segrest defined an $(N+1) \times (N+1)$ transition matrix $P(i,j)$ 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 two-class, simple GA using proportionate (a.k.a. roulette wheel) selection with no mutation. 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 total fitness of the population. Thus, with probability $f_k / \sum f$, we choose individual k (where f_k is the fitness of k and $\sum f$ is the sum of the fitnesses of all individuals in the current population).

In the *generational* GA, we replace the entire population each generation, thus making N selections per generation. In our two-class model, we can write the denominator $\sum f$ as simply $f_A i + f_B(N-i)$, where i is the number of copies of **A** in the current population, f_A is the fitness of **A**, and f_B is the fitness of **B**. Then the probability of choosing an **A** for the next generation's population is $p_A = \frac{f_A i}{f_A i + f_B(N-i)}$. Letting r_f be the fitness ratio $\frac{f_A}{f_B}$, we rewrite p_A as $\frac{r_f i}{r_f i + (N-i)}$.

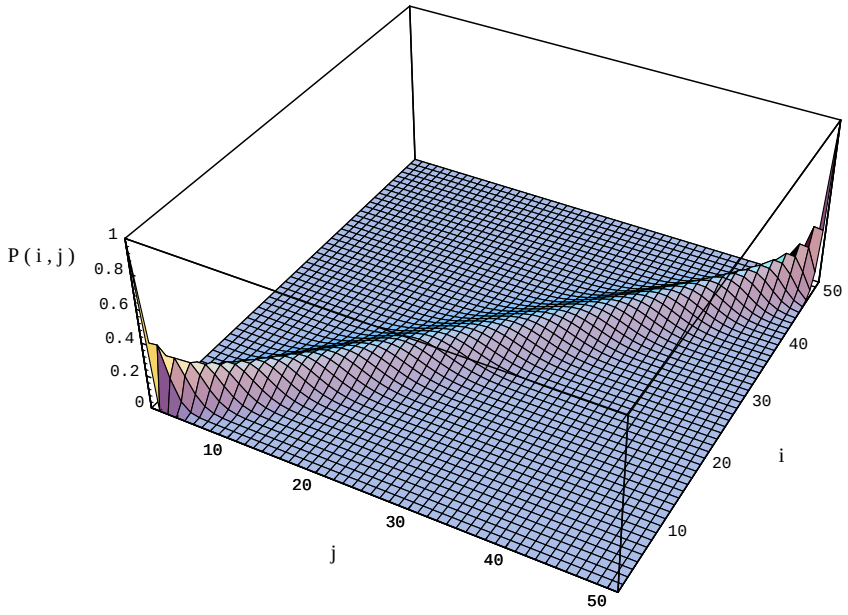

Figure 8: The transition matrix for a simple GA with $N = 50$ and $r_f = 1$ (genetic drift). $P(i, j)$ is the probability of transiting from current state i to next state j in a single generation. Note the absorbing states $i = 0$, 50.

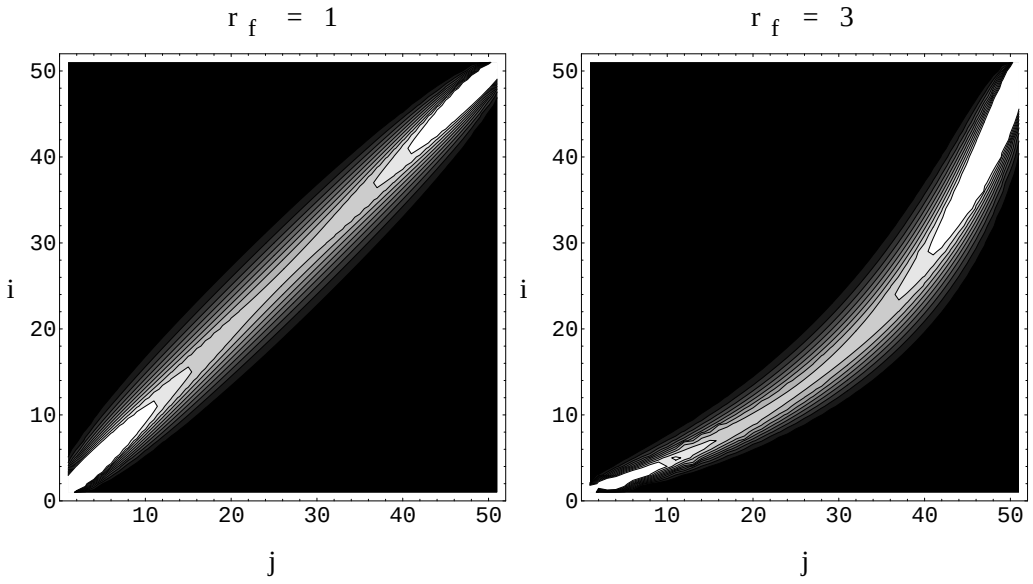

Figure 9: Contour plots of transition matrices for the simple GA with $N = 50$ and $r_f = 1, 3$. On the left is a contour plot of the surface in Figure 8, showing the case of genetic drift. On the right is a matrix illustrating selection pressure.

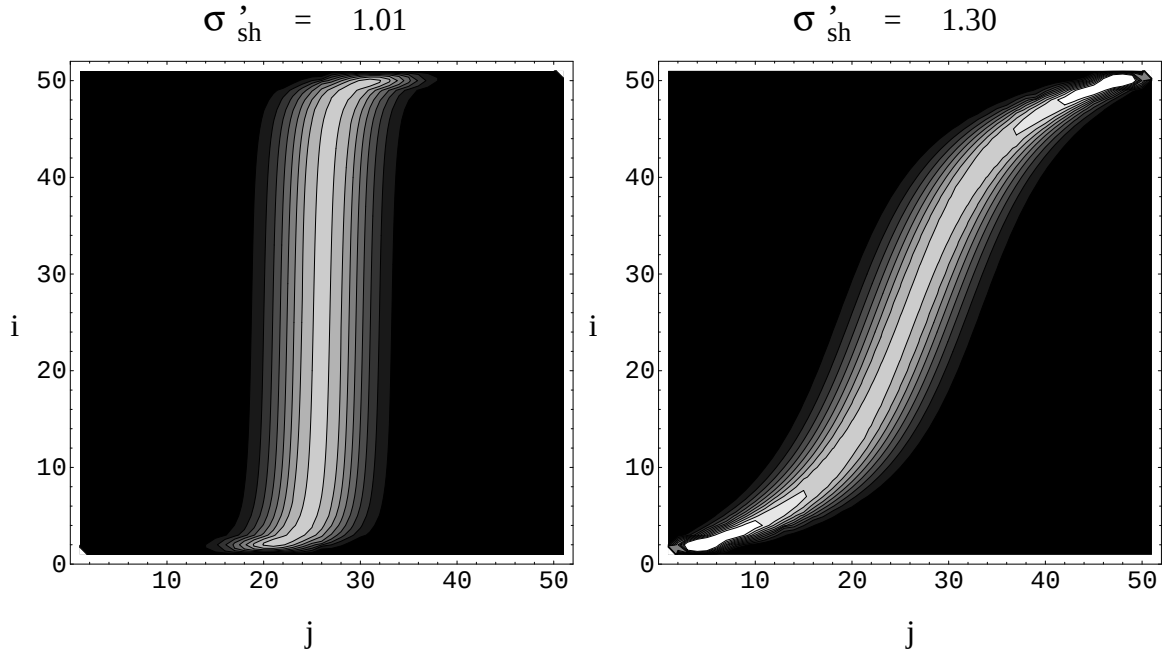

Figure 10: Transition matrices for a GA with fitness sharing, assuming overlapping niches. Here $N = 50$, $r_f = 1$, and $\sigma'_{sh} = 1.01$, 1.30. As the ratio of niche overlap, σ'_{sh}, increases, the "restorative pressure" decreases. The transition matrix changes from the row constant matrix of "perfect sharing" at $\sigma'_{sh} = 1$, approaching the matrix for genetic drift (for $r_f = 1$) or selection pressure (for $r_f \neq 1$) as $\sigma'_{sh} \to \infty$.

The probability of choosing a **B** is $p_B = 1 - p_A$. The probability of going from a state with i **A**s to a state with j **A**s is $P(i,j) = \binom{N}{j}(p_A)^j(p_B)^{N-j}$. Substituting for p_A and p_B:

$$P(i,j) = \binom{N}{j}\left(\frac{r_f i}{r_f i + (N-i)}\right)^j \left(\frac{N-i}{r_f i + (N-i)}\right)^{N-j}. \tag{11}$$

Equation 11 defines a complete transition matrix for any population size N and fitness ratio r_f. In Figure 8 we plot the transition probabilities for a population of size 50, and a fitness ratio of $r_f = 1$. On the left of Figure 9 is a contour plot of the surface plot in Figure 8. With $r_f = 1$, Equation 11 reduces to the equation for pure *genetic drift*. Note that the two states $i = 0$ and $i = N$ are absorbing states, with probability rows consisting of a single spike of probability one ($P(0,0) = P(N,N) = 1$). Goldberg and Segrest used Equation 11 to investigate expected times to absorption for the drift case ($r_f = 1$).

When there is a preference ($r_f \neq 1$), selection pressure can be visualized in the transition matrix. Figure 9, 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 more fit individual. The presence of the ridge in the lower or upper triangles of the matrix indicates a pressure toward more or less copies of **A**, respectively.

5.2 Markov chain analysis of the niched GA

Horn (1993) modified the above model to account for niching by simply substituting the *shared* fitnesses of **A** and **B** for their objective fitnesses. Specifically, he replaced f_A by f_A/m_A and f_B by f_B/m_B, calculating the niche counts m_A and m_B assuming some niche overlap (here normalized by the distance between niches) $\sigma'_{sh} \equiv \sigma_{sh}/d_{A,B}$, and an $\alpha_{sh} = 1$. In general, assuming some degree of overlap ($\sigma'_{sh} > 1$), the transition matrix for a two-class niched GA looks like those in Figure 10.

The niched GA exhibits a restorative pressure toward an equilibrium state. In Figure 10, this state is at the population of half **A**s and half **B**s. At states with more than half **A**s, there is a likelihood of losing **A**s. Conversely, when the number of **A**s is less than $N/2$, we are likely to gain **B**s at the expense of **A**s. As niche overlap increases (larger σ'_{sh}), the restorative pressure decreases and we approach the matrix for genetic drift. When niche overlap is minimal (at or near zero), we get "perfect sharing", in which the row of transition probabilities, and hence the restorative pressure, is constant over all the transient (i.e., non-absorbing) states.

5.3 Markov chain analysis of LCS implicit niching

We now make use of the Markov chain models described above to visualize niching in the LCS. We need only calculate the shared fitnesses for rules **A** and **B** and substitute them for the objective fitnesses f_A and f_B in the derivation of Equation 11. Let n_A be 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 LCS sharing, which is the equal division of a resource among all individuals competing for it, we can calculate the *expected* shared fitnesses of **A** and **B**:

$$f_{sh,A} = \frac{f_A - f_{AB}}{n_A} + \frac{f_{AB}}{N} \qquad\qquad f_{sh,B} = \frac{f_B - f_{AB}}{n_B} + \frac{f_{AB}}{N}, \tag{12}$$

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

$$p_A = \frac{f_A - f_{AB} + f_{AB}\frac{n_A}{n_A+n_B}}{f_A - f_{AB} + f_{AB}\frac{n_A}{n_A+n_B} + f_B - f_{AB} + f_{AB}\frac{n_B}{n_A+n_B}}. \tag{13}$$

Again equating state i to the population with $n_A = i$ copies of **A**, and $N - i = n_B$ copies of **B**, and dividing numerator and denominator by f_A, the above reduces to

$$p_A = \frac{1 - \frac{f_{AB}}{f_A} + \frac{f_{AB}}{f_A}\frac{i}{N}}{1 - \frac{f_{AB}}{f_A} + \frac{f_{AB}}{f_A}\frac{i}{N} + \frac{f_B}{f_A} - \frac{f_{AB}}{f_A} + \frac{f_{AB}}{f_A}\frac{N-i}{N}}. \tag{14}$$

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

$$p_A = \frac{1 + r_o(\frac{i}{N} - 1)}{1 - r_o + r_o\frac{i}{N} + \frac{1}{r_f} - r_o + r_o\frac{N-i}{N}}. \tag{15}$$

Finally, simplification yields

$$p_A = \frac{1 - r_o + r_o\frac{i}{N}}{1 - r_o + \frac{1}{r_f}}. \tag{16}$$

Similarly, for **B**,

$$p_B = \frac{\frac{1}{r_f} - r_o\frac{i}{N}}{1 - r_o + \frac{1}{r_f}}. \tag{17}$$

Inserting these probabilities into our transition function, $P(i,j) = \binom{N}{j}(p_A)^j(p_B)^{N-j}$, we can compute the complete transition matrix:

$$P(i,j) = \binom{N}{j}\left(\frac{1 - r_o + r_o\frac{i}{N}}{1 - r_o + \frac{1}{r_f}}\right)^j\left(\frac{\frac{1}{r_f} - r_o\frac{i}{N}}{1 - r_o + \frac{1}{r_f}}\right)^{N-j}. \tag{18}$$

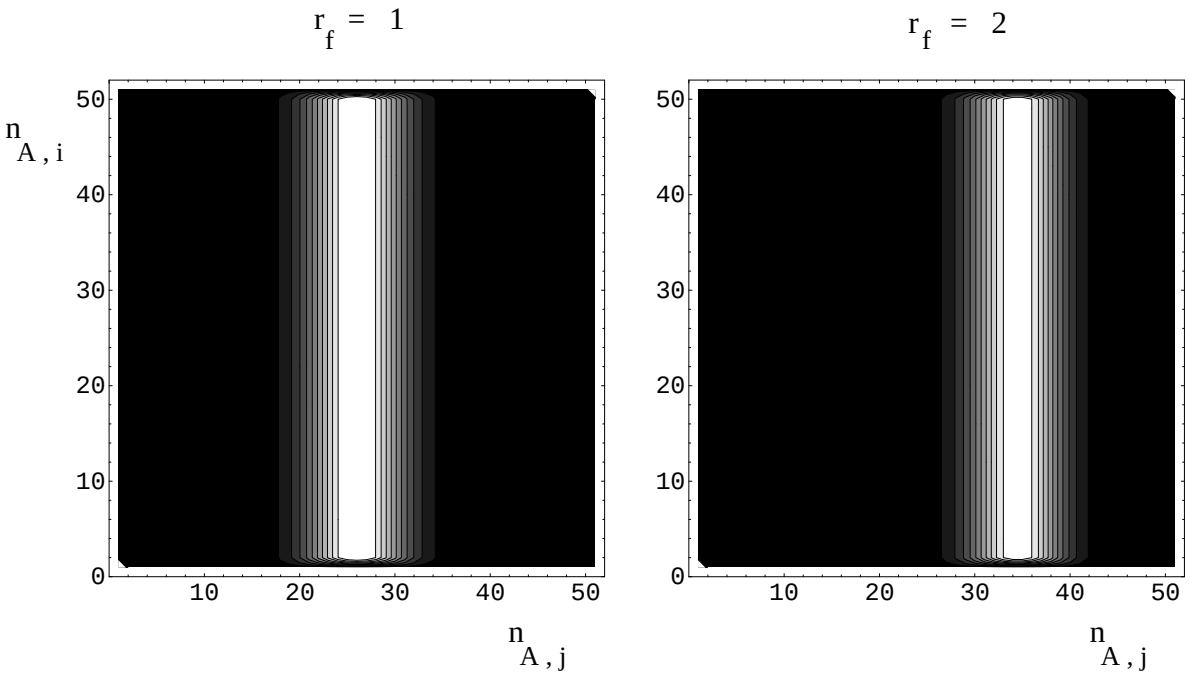

Figure 11: The transition probabilities for implicit niching with a population limited to two rules which do not overlap in coverage ($r_o = 0$). On the left the rules are equally fit, ($r_f = 1$). On the right, the apparent equilibrium point is a population with twice as many copies of **A** than of **B**. Here $r_f = 2$.

5.3.1 Visualizing restorative pressure

Now we can begin to visualize the LCS niching force. Let us first assume equally fit rules ($r_f = 1$) and no overlap ($r_o = 0$). In Figure 11, we show a contour plot of the transition probability matrix for a population size $N = 50$. This matrix closely resembles that of perfect sharing for the niched GA (Horn, 1993). Indeed, when there is no overlap, the matrices for LCS niching and the niched GA are exactly the same. Both algorithms are simply dividing up the objective fitness of each individual among all the copies of that individual[7]. These algorithms differ only in how they handle the case of overlapping niches.

In Figure 12 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 at $r_o = 0$ to pure selection at $r_o = 1/r_f$, 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. The same phenomenon is observed in niched GAs (Horn, 1993). Note that at $r_f = 1$ and $r_o = 1/r_f = 1$, we have the case of pure genetic drift.

5.3.2 Rule maintenance times

To further establish that LCS niching generates a restorative pressure akin to that of niched GAs, we also look at expected niche maintenance times. In (Horn, 1993) the expected time to loss of one of the two niches (i.e., classes of individuals) was calculated. 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). Calculating the absorption time for an absorbing Markov chain is straightforward, although

[7]In the LCS, however, this fitness division is 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 in an LCS. But this is not a qualitative difference in the Markov model.

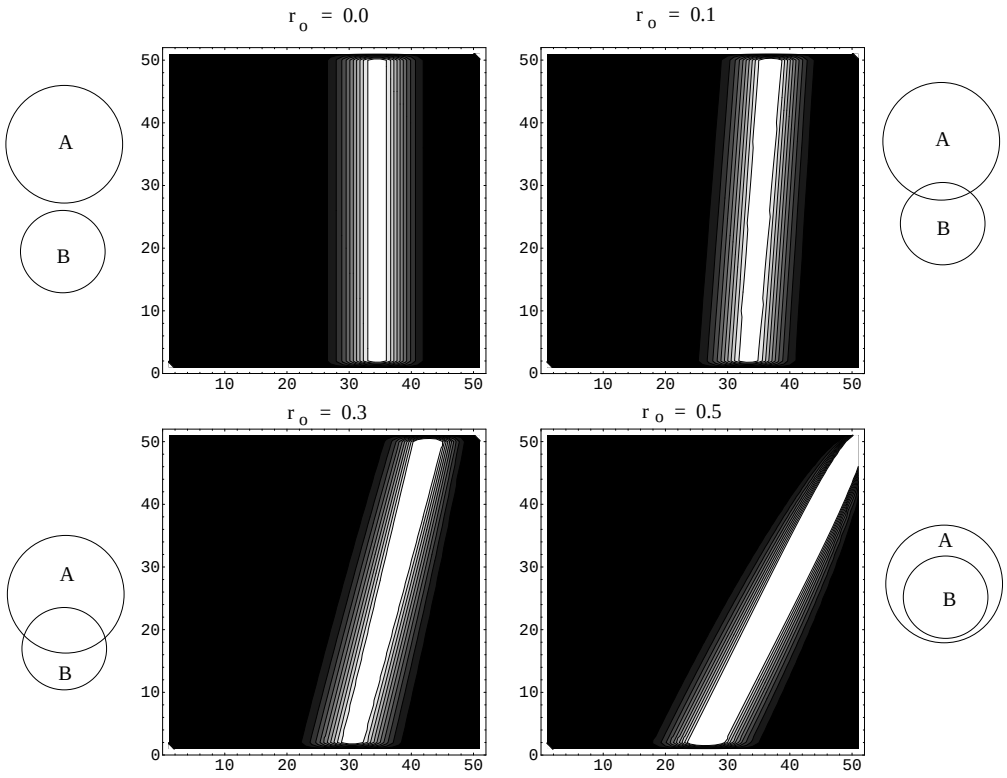

Figure 12: The LCS niching pressure decreases with increasing overlap r_o.

computationally expensive for large matrices (i.e., large population size N). We assume a random initial population, which has little chance of being in either absorbing state. We find that the expected absorption time grows exponentially with population size N, just as with the niched GA. The exponent in the growth decreases as r_f moves away from one, just as in the case of the niched GA. Finally, holding r_f constant, we find that the exponent of growth (in N) also decreases with increasing niche overlap, just as in the niched GA.

Because the case of no overlap ($r_o = 0$) is exactly the same as for the niched GA with $\sigma_{sh} \leq d_{A,B}$, we do not show here a plot of absorption time versus N for $r_o = 0$ and various values of r_f. Such a graph is shown in (Horn, 1993). However, since the two niching algorithms handle overlap differently, we do include here a plot of absorption time versus N for overlapping niches. That is, we vary $r_o > 0$ while holding r_f constant. In Figure 13 we show the growth in expected absorption (or niche loss) time as a function of N for $r_o = 0.0, 0.1, 0.3, 0.5$, and $r_f = 2$.

This growth (Figure 13) appears to be exponential. Indeed, for the case of perfect sharing ($r_o = 0$), we can find the exact exponent. Since the transition probabilities for all transient states are identical, it follows that prior to absorption we always have the same probability of absorption, namely $P_{absorb} = p(i,0) + p(i,N)$, $\forall i : (0 < i < N)$. Now, $p(i,0)$ is simply the probability of choosing N **B**s: $p(i,0) = (p_B)^N = (\frac{1}{r_f+1})^N$, when $r_o = 0$. Similarly, $p(i,N) = (p_A)^N = (\frac{r_f}{r_f+1})^N$. The expected time to absorption, assuming that the chain is started in a transient state, is the inverse of the probability of absorption: $\frac{1}{p(i,0)+p(i,1)}$:

$$E[t_{abs}] = \frac{1}{(\frac{1}{r_f+1})^N + (\frac{r_f}{r_f+1})^N}. \tag{19}$$

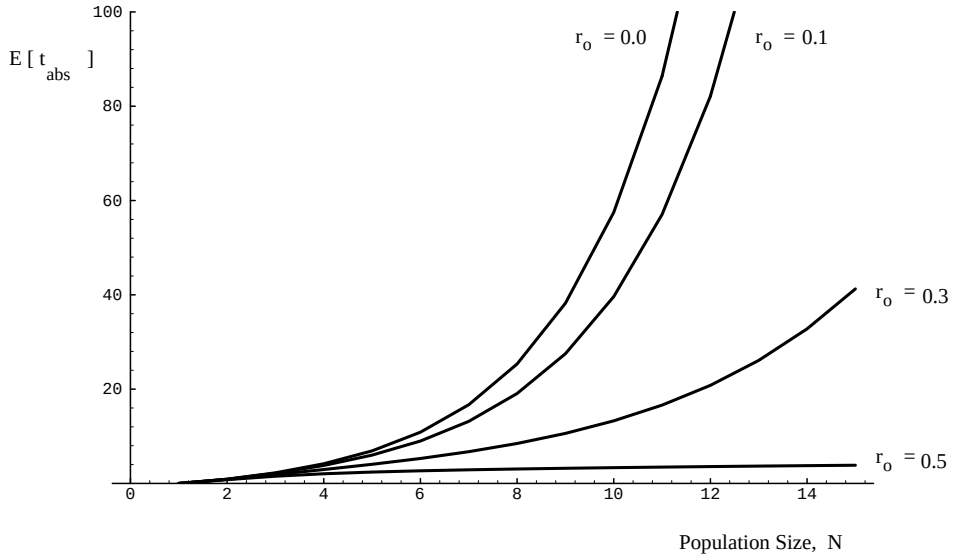

Figure 13: The expected time to niching failure in our two-rule LCS is the absorption time of the Markov chain. This time grows exponentially with population size N (provably at least for $r_o = 0$), a characteristic we propose as fundamental to niching. The exponent of growth appears to decay with r_o, representing a degradation of niching pressure. When $r_o = \frac{1}{r_f}$, selective 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). Here $r_f = 2$.

After some rearranging:

$$E[t_{abs}] = \frac{(r_f + 1)^N}{r_f^N + 1}. \tag{20}$$

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 $r_o \neq 0$, it might be impossible to calculate a closed form expression for absorption time, thus making it difficult to prove exponential growth. However, in Figure 13 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). 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.

5.3.3 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 the LCS to maintain a diverse set of good rules for a very long time (subject to the effects of other sources of noise). We might want to know what kind of distribution the LCS will maintain during that time. Goldberg and Richardson (1987) and Deb (1989) predict that a niched GA, under perfect sharing conditions (i.e., no overlap of niches) should reach an equilibrium when the shared fitnesses of all individuals are equal:

$$\frac{f_A}{m_A} = \frac{f_B}{m_B}, \tag{21}$$

for all pairs of distinct niches A and B. If we assume non-overlapping niches centered on local optima, a population size N, and knowledge of the number and objective fitnesses of the local optima f_X, the above

equations allow us to calculate the expected distribution of the population (i.e., the m_X) when the niched GA is at equilibrium. For the two-class model, with $\sigma_{sh} \leq d_{A,B}$, $m_A = i$, and $m_B = N - i$, the steady-state equation, Equation 21, reduces to

$$\frac{f_A}{i_{eq}} = \frac{f_B}{N - i_{eq}} \quad \Rightarrow \quad \frac{N - i_{eq}}{i_{eq}} = \frac{f_B}{f_A} = \frac{1}{r_f} \quad \Rightarrow \quad i_{eq} = \frac{r_f N}{1 + r_f}. \tag{22}$$

But how well does the niched GA actually satisfy this equilibrium condition? Put another way, does the population remain at or near this equilibrium point over the long term?

Horn (1993) calculated an approximation to a steady-state distribution for the Markov chain model of the niched GA. He then used this to show that the expected distribution is approximately that predicted by the equilibrium condition in Equation 22, although the small discrepancy grows with increasing overlap.

Here we perform the steady-state analysis on the model of LCS niching. We refer the reader to (Horn, 1993) for a description of the technique itself. The steady-state distribution calculation is a straightforward, well-known procedure in Markov chain analysis, but only for ergodic Markov chains. Absorbing chains, like the ones above, are not ergodic. However, if the absorption times are much longer than the "restorative time" (that is, restoration to equilibrium), the chain can be considered *quasi-ergodic*. Horn (1993) contains a brief discussion of this issue, while Darroch and Seneta (1965) treat the issue more thoroughly. Suffice to say here that we can approximate the absorbing chain, for the purpose of calculating the steady-state distribution, by an ergodic chain obtained by simply "chopping off" the rows and columns corresponding to the absorbing states. We then have to normalize the resulting submatrix (often called the Q matrix in the Markov chain literature) so that all rows sum to one. The normed matrix 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 LCS niching, for a population size $N = 16$, fitness ratio $r_f = 1$, and various degrees of overlap $r_o \geq 0$. Figure 14 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 (Horn, 1993) for a niched GA (with 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 (proportionate selection).

Horn (1993) used the steady-state distribution to calculate the expected number of copies of individual **A** at steady-state. He found that this was in exact agreement with the equilibrium prediction i_{eq} for perfect sharing (no overlap), and in very close agreement when niches do overlap. Although we do not show such comparisons here, we have found the same agreement in the case of LCS niching. This result justifies our use of the equilibrium point in the next section.

6 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 i_{eq} and the derivation of Equation 22 to look at the LCS sharing function at the corresponding value of r_n (remember that $r_n = n_B/n_A$), naming this particular r_n the "equilibrium ratio" $r_{eq,n}$. Since $n_A = i_{eq}$ and $n_B = N - i_{eq}$,

$$r_{eq,n} = \frac{n_B}{n_A} = \frac{N - i_{eq}}{i_{eq}} = \frac{f_B}{f_A} = \frac{1}{r_f}. \tag{23}$$

With this $r_{eq,n}$ in hand, we can go back to our LCS sharing function shown in Figure 7 as a function of r_o and r_n, and look at slices of the surface corresponding to $r_{eq,n}$ 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 LCS to GA fitness sharing more directly.

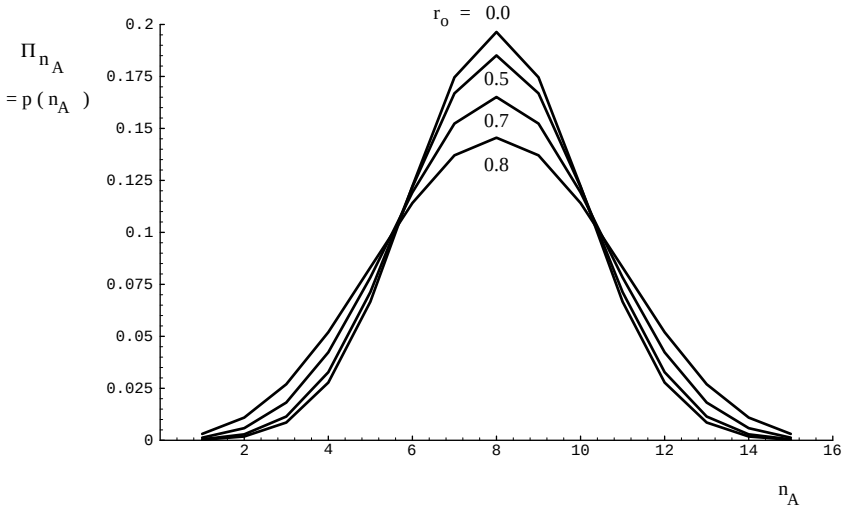

Figure 14: The steady-state probability distribution of the Markov chain model of LCS niching. Here we use population size $N = 16$, equally fit rules $r_f = 1$, and various degrees of overlap $r_o = 0.0, 0.5, 0.7, 0.8$. In all four cases, a distribution symmetric about the equilibrium point of $n_A = 8$ is maintained. This distribution appears to gradually degrade with overlap.

Before taking such slices of the LCS sharing function, however, we note that $r_{eq,n}$ is calculated above under the assumption of perfect sharing. That is, Goldberg and Richardson (1987) assumed no niche overlap. However, since LCS implicit niching is different from GA explicit fitness sharing only when niches do overlap, and because we are interested in cases where two rules overlap almost completely, we modify the equilibrium condition results to account completely and exactly for overlap[8]. The general equilibrium condition is:

$$f_{sh,A} = f_{sh,B}. \tag{24}$$

Substituting the formulae for shared fitness in an LCS, Equation 12,

$$\frac{f_A - f_{AB}}{n_A} + \frac{f_{AB}}{N} = \frac{f_B - f_{AB}}{n_B} + \frac{f_{AB}}{N}. \tag{25}$$

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

$$\frac{n_B}{n_A} = \frac{f_B - f_{AB}}{f_A - f_{AB}} = \frac{\frac{f_B}{f_A} - \frac{f_{AB}}{f_A}}{1 - \frac{f_{AB}}{f_A}}. \tag{26}$$

Substituting our ratios r_f and r_o, rearranging, and naming the result $r'_{eq,n}$,

$$r'_{eq,n} = \frac{\frac{1}{r_f} - r_o}{1 - r_o}. \tag{27}$$

Note that $r'_{eq,n} = r_{eq,n}$ when $r_o = 0$. In Figure 15, 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 ($r_o = 1/r_f$). For $r_f = 1$, there is no selective pressure and hence no preference. These curves

[8] Deb (1989; and with Goldberg, 1989) used the general equilibrium condition (Equation 24) to show that increasing niche overlap, with a fixed population size, leads to niching failure.

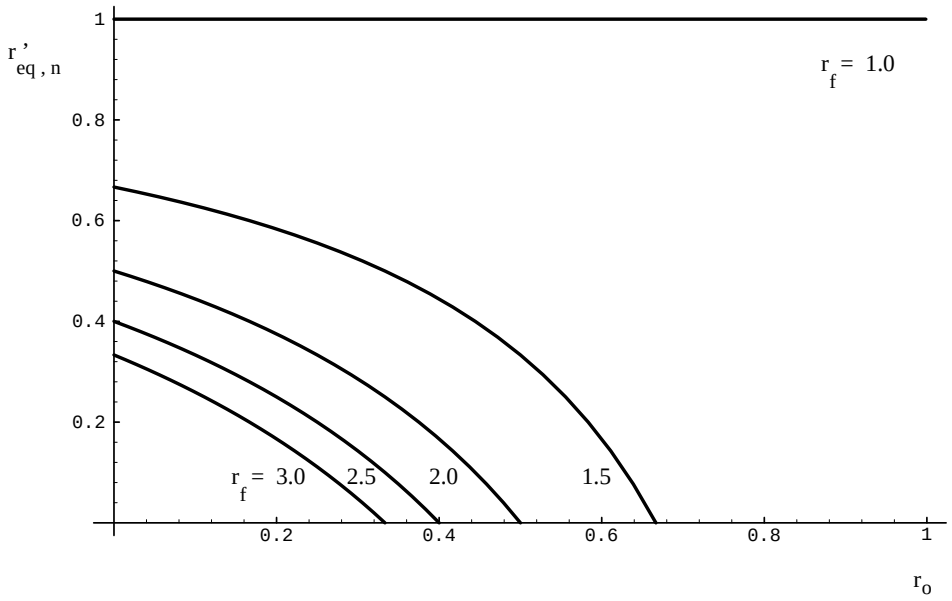

Figure 15: Equilibrium proportions change with varying overlap r_o and fitness ratios r_f.

lie in the (r_o, r_n) plane, and can be used to take slices of the LCS sharing function in Figure 7. We need only substitute $r'_{eq,n}$ for r_n in Equation 10, and simplify:

$$\left[\frac{\partial D}{\partial n_B}\right]_{eq} = \frac{r_o}{(r_o - \frac{1}{r_f} - 1)^2}. \tag{28}$$

We can then plot $\left[\frac{\partial D}{\partial n_B}\right]_{eq}$ as a function of r_o only: Figure 16 is a plot of such slices (Equation 28) through the general sharing function $\frac{\partial D}{\partial n_B}$. That is, Figure 16 gives the one-dimensional LCS sharing function that is arguably of greatest interest. If we assume that the LCS will maintain a population distribution at or near $r'_{eq,n}$ in Equation 27, 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 16.

7 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) rules, (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, and has been used to solve some hard problems. 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.

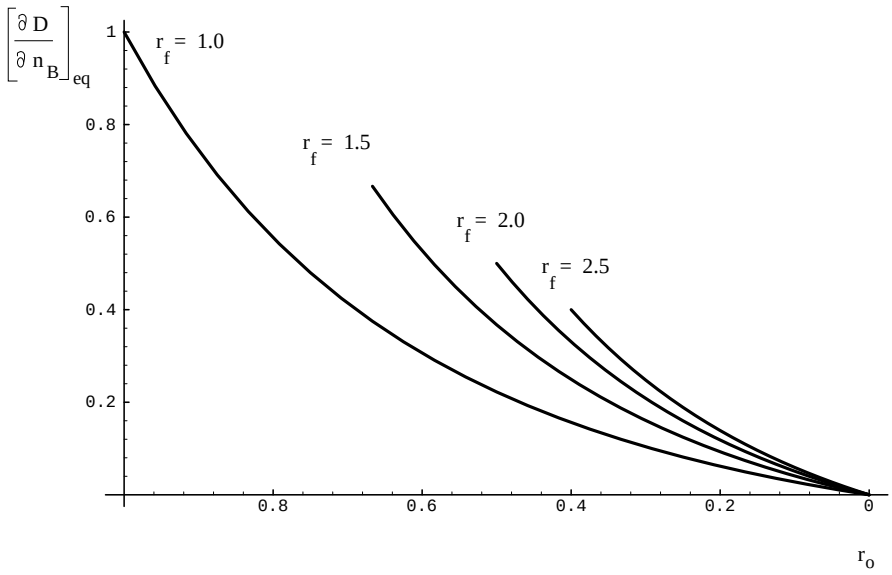

Figure 16: The LCS sharing function along the curves of equilibrium. We note the similarity in shape, but not scale, of the function for different r_f.

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, although 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.

8 Conclusions

To summarize, in this study we applied two general tactics in our analysis of the LCS:

- Functional decomposition of the LCS
- Application of lessons learned from GAs

Functional decomposition is important because of the complexity of the LCS. Once we identify separable, quasi-independent subprocesses in the LCS, we can try to understand the behavior of each in isolation from the others. In addition, each LCS mechanism by itself is a relatively minor modification of the simple GA. By studying each mechanism separately, we can bring to bear the more advanced tools and results

of GA theory. In a sense, GA theory and LCS analysis are converging. As GA theory advances, we are able to understand more sophisticated variants of the simple GA, such as niching. If we can break down the complexity of the LCS, we can start to map components of the LCS onto the advanced GAs. We hope that in this paper we have begun that process.

The specific focus of this paper was on niching in the LCS. 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 (Wilson, 1987). Smith, Forrest, and Perelson (1993) began the process of mapping this *implicit* or *emergent* sharing to explicit fitness sharing. In this study, we have continued in that direction, although we focus on the LCS rather than the immune system model. In addition, our focus has been on demonstrating an LCS *sharing function*, *restorative pressure*, *steady-state distribution*, and a meaningful *equilibrium point*. We have done so by applying tools and techniques from GA niching theory and GA theory in general.

The important lessons from this study include the close correspondence of LCS implicit niching and GA fitness sharing. This should come as no surprise: Goldberg and Richardson were originally motivated by the LCS in their development of fitness sharing. Indeed, they were trying to find some way to share the fitness of a local neighborhood without knowing exactly what the boundaries of the neighborhood are. In the LCS, by contrast, sharing is easy because we know exactly what resources are covered by a rule, so we have no need for some artificial σ_{sh} niche radius. Niched GAs have to make do with much less information about the fitness function. Still, GA fitness sharing might be improved by making it even more closely analogous to LCS niching. For example, we saw how LCS niching seems to use a sharing function that looks like $Sh(d)$ with a sharing exponent $\alpha_{sh} < 1$. Yet the most common setting of α_{sh} in niched GAs is exactly one (the triangular sharing function). We also saw how the LCS analog of the sharing function changes with the changing sizes of niche subpopulations, yet $Sh(d)$ for niched GAs is constant.

The LCS $\leftrightarrow$ niched GA mapping we have established is a two-way street. We can apply to the LCS other lessons learned in our studies of niched GAs, in addition to analytical tools. Goldberg, Deb, and Horn (1992) characterize three fundamental challenges for a niched GA with fitness sharing:

- Deception
- Massive multimodality
- Separation

Even with its extra knowledge of the search space, these three difficulties plague the LCS nicher as well (Goldberg, Horn, & Deb, 1992). Deception[9] can be built into any type of search space, including the S-R LCS binary classification "schema space" introduced earlier. Constructing and overcoming LCS deception is a topic of future work. Massive multimodality becomes a problem when our *desirable* rules have fitnesses within an order of magnitude of the fitnesses of a very large number of undesirable rules. Here "very large" means on the order of population size N. In such a situation just a few copies of a desirable rule, perhaps as few as two, degrade its shared fitness until it is below that of the undesirables. This leads to the large field of undesirables "swallowing up" our entire population, with an occasional appearance of one or two copies of a desirable rule.

Separation also carries over as a problem for LCS niching, but in a more subtle way. Although we no longer have to worry about setting σ_{sh} just right, we do have a problem separating cooperative pairs from competitive pairs. As overlap increases, two rules become less cooperative and more competitive. Because implicit niching has no parameters for us to adjust (it is fixed), the LCS will decide for itself when a pair becomes so competitive that a lesser individual (lower fitness than either of the pair) with no overlap of coverage, is preferable. LCS niching might be in need of the same kind of parameterized tweaking we have in GA niching, so that we can finely adjust the equilibrium distribution of the population. The ability to predict and control the location of the cooperative-competitive boundary in the LCS is critical to users, and is a subject of future work.

[9] Building block misleadingness (Goldberg, 1989).

A fourth type of difficulty for niched GAs also carries over to the LCS via implicit niching. Oei, Goldberg, and Chang (1991) demonstrated the unstable, chaotic behavior of fitness sharing in a generational GA with tournament selection. They showed that even binary tournament selection overcompensates for small deviations from the equilibrium distribution, resulting in large swings in subpopulation sizes and hence early niche loss. This chaotic behavior will also occur in the LCS if tournament selection is substituted directly for proportionate selection. Their suggested solution, continuously updating niche counts using the partially filled new population, should apply to LCS niching as well.

Besides carrying over fundamental niching problems from the niched GA, our LCS $\leftrightarrow$ niched GA mapping also brings to the LCS the demonstration of true niching capability. We have shown the ability of the LCS to balance selective 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. Here again theoretical work on niched GAs carries over. The closed form expressions for expected absorption times in the Markov chain models can be solved for N in terms of desired rule maintenance times. Such a result would yield the first analytical population sizing guideline for the LCS, and for coadaptive systems in general. LCS population sizing is another fruitful area for extending the results of this paper.

The most important point of this paper, however, is the demonstration 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 selective pressure. If we want to maintain cooperative rules, we must balance that selective 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. Fortunately, we already know a fairly successful niching method: fitness sharing. And the inspiration for explicit fitness sharing in GAs came from *implicit* niching in the LCS. Besides, implicit niching is nature's way.

Acknowledgments

This work began as an extended abstract (Goldberg, Horn, & Deb, 1992) submitted to the First International Workshop on Learning Classifier Systems, organized by Robert E. Smith and Manuel Valenzuela-Rendón, and held in October 1992 at the Johnson Space Center in Houston, Texas. The subsequent development of the paper owes much to the criticisms, suggestions, and general discussions provided by the participants of that workshop. The quality of the paper also improved with suggestions from several reviewers, both anonymous and known.

The first author is grateful for support provided by NASA under contract number NGT-50873. The second and third authors acknowledge the support of the U.S. Army under contract number DASG60-90-C-0153 and the National Science Foundation under Grant ECS-9022007.

References

Booker, L. B. (1982). Intelligent behavior as an adaptation to the task environment. *Dissertation Abstracts International, 43*(2), 469B. (University Microfilms No. 8214966)

Cavicchio, D. J. (1970). *Adaptive search using simulated evolution.* Ph.D. thesis, University of Michigan, Ann Arbor, (University Microfilms No. 25-0199).

Collins, R. J., & Jefferson, R. J. (1991). Selection in massively parallel genetic algorithms. In R. K. Belew

& L. B. Booker (Eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms.* San Mateo, CA: Morgan Kaufmann, 249–256.

Darroch, J. N., & Seneta, E. (1965). On quasi-stationary distributions in absorbing discrete-time finite Markov chains. *Journal of Applied Probability, 2*, 88–100.

Davidor, Y. (1991). A naturally occuring niche & species phenomenon: the model and first results. In R. K. Belew & L. B. Booker (Eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms.* San Mateo, CA: Morgan Kaufmann, 257–263.

Davis, T. E., & Principe, J. C. (1991). A simulated annealing-like convergence theory for the simple genetic algorithm. In R. K. Belew & L. B. Booker (Eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms.* San Mateo, CA: Morgan Kaufmann, 174–181.

Davis, T. E., & Principe, J. C. (1993). A Markov chain framework for the simple genetic algorithm. *Evolutionary Computation, 1(3)*, 269–288.

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.* San Mateo, CA: Morgan Kaufmann, 42–50.

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. (1990). A note on Boltzmann tournament selection for genetic algorithms and population-oriented simulated annealing. *Complex Systems, 4,* 445–460.

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. (1994). Genetic and evolutionary algorithms come of age. *Communications of the Association for Computing Machinery, 37(3)*, March 1994, 113–119.

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, 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*, 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.

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.* Pittsburgh, PA: Carnegie-Mellon University, 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.* San Mateo, CA: Morgan Kaufmann, 110–117.

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.

Mahfoud, S. W. (1991). *An analysis of Boltzmann tournament selection* (IlliGAL Report No. 91007). Urbana, IL: Department of General Engineering, University of Illinois at Urbana-Champaign. To appear in *Complex Systems.*

Mahfoud, S. W. (1992). Crowding and preselection revisited. In R. Männer & B. Manderick (Eds.), *Parallel Problem Solving From Nature, 2.* 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.

Michalski, R. S., Carbonell, J. G., & Mitchell, T. M., 1983. *Machine Learning.* Palo Alto, CA: Tioga Press.

Nix, A. E., & Vose, M. D. (1992). Modeling genetic algorithms with Markov chains. *Annals of Mathematics and Artificial Intelligence, 5(1)*, 79–88.

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.

Pitt, L., & Valiant, L. G. (1988). Computational limitations on learning from examples. *Journal of the Association for Computing Machinery, 35(4)*, 965–984.

Smith, R. E., Forrest, S., & Perelson, A. S. (1993). Searching for diverse, cooperative subpopulations with genetic algorithms. *Evolutionary Computation, 1(2)*, 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*, San Mateo: Morgan Kaufmann, 340–346.

Wilson, S. W. (1987). Classifier systems and the animat problem. *Machine Learning, 2(3)*, 199-228.

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.

Valiant, L. G. (1984). A theory of the learnable. *Communications of the Association for Computing Machinery, 27(11)*, 1134–1142.

Vose, M. D. (1992). Modeling simple genetic algorithms. In L. D. Whitley, (Ed.), *Foundations of Genetic Algorithms 2*, San Mateo, CA: Morgan Kaufmann, 63–73.