Spring 1994

Genetic Algorithms and Artificial Life

Melanie Mitchell
Portland State University

Stephanie Forrest

Citation Details

Mitchell, Melanie, and Stephanie Forrest. "Genetic algorithms and artificial life." Artificial Life 1.3 (1994): 267-289. doi:10.1162/artl.1994.1.3.267

Genetic Algorithms and Artificial Life

Melanie Mitchell
Santa Fe Institute
1660 Old Pecos Tr., Suite A
Santa Fe, NM 87501
mm@santafe.edu

Stephanie Forrest
Department of Computer Science
University of New Mexico
Albuquerque, NM 87131-1386
forrest@cs.unm.edu

Abstract Genetic algorithms are computational models of evolution that play a central role in many artificial-life models. We review the history and current scope of research on genetic algorithms in artificial life, giving illustrative examples in which the genetic algorithm is used to study how learning and evolution interact, and to model ecosystems, immune system, cognitive systems, and social systems. We also outline a number of open questions and future directions for genetic algorithms in artificial-life research.

1 Introduction

Evolution by natural selection is a central idea in biology, and the concept of natural selection has influenced our view of biological systems tremendously. Likewise, evolution of artificial systems is an important component of artificial life, providing an important modeling tool and an automated design method. Genetic algorithms (GAs) are currently the most prominent and widely used models of evolution in artificial-life systems. GAs have been used both as tools for solving practical problems and as scientific models of evolutionary processes. The intersection between GAs and artificial life includes both, although in this article we focus primarily on GAs as models of natural phenomena. For example, we do not discuss topics such as "evolutionary robotics" in which the GA is used as a black box to design or control a system with lifelike properties, even though this is certainly an important role for GAs in artificial life. In the following, we provide a brief overview of GAs, describe some particularly interesting examples of the overlap between GAs and artificial life, and give our view of some of the most pressing research questions in this field.

2 Overview of Genetic Algorithms

In the 1950s and 1960s several computer scientists independently studied evolutionary systems with the idea that evolution could be used as an optimization tool for engineering problems. In Goldberg's short history of evolutionary computation ([42], chap. 4), the names of Box [21], Fraser [39, 40], Friedman [41], Bledsoe [18], and Bremermann [22] are associated with a variety of work in the late 1950s and early 1960s, some of which presages the later development of GAs. These early systems contained the rudiments of evolution in various forms—all had some kind of "selection of the fittest," some had population-based schemes for selection and variation, and some, like many GAs, had binary strings as abstractions of biological chromosomes.

In the later 1960s, Rechenberg [89] introduced "evolution strategies," a method first designed to optimize real-valued parameters. This idea was further developed by Schwefel [96, 97], and the field of evolution strategies has remained an active area

of research, developing in parallel to GA research, until recently when the two communities have begun to interact. For a review of evolution strategies, see [9]. Also in the 1960s Fogel, Owens, and Walsh [36] developed "evolutionary programming." Candidate solutions to given tasks are represented as finite-state machines, and the evolutionary operators are selection and mutation. Evolutionary programming also remains an area of active research. For a recent description of the work of Fogel et al., see [34].

GAs as they are known today were first described by John Holland in the 1960s and further developed by Holland and his students and colleagues at the University of Michigan in the 1960s and 1970s. Holland's 1975 book, *Adaptation in Natural and Artificial Systems* [55], presents the GA as an abstraction of biological evolution and gives a theoretical framework for adaptation under the GA. Holland's GA is a method for moving from one population of "chromosomes" (e.g., bit strings representing organisms or candidate solutions to a problem) to a new population, using selection together with the genetic operators of crossover, mutation, and inversion. Each chromosome consists of "genes" (e.g., bits), with each gene being an instance of a particular "allele" (e.g., 0 or 1). Selection chooses those chromosomes in the population that will be allowed to reproduce and decides how many offspring each is likely to have, with the fitter chromosomes producing on average more offspring than less fit ones. Crossover exchanges subparts of two chromosomes (roughly mimicking sexual recombination between two single-chromosome organisms); mutation randomly changes the values of some locations in the chromosome; and inversion reverses the order of a contiguous section of the chromosome, thus rearranging the order in which genes are arrayed in the chromosome. Inversion is rarely used in today's GAs, at least partially because of the implementation expense for most representations. A simple form of the GA (without inversion) works as follows:

1. Start with a randomly generated population of chromosomes (e.g., candidate solutions to a problem).
2. Calculate the fitness of each chromosome in the population.
3. Apply selection and genetic operators (crossover and mutation) to the population to create a new population.
4. Go to step 2.

This process is iterated over many time steps, each of which is called a "generation." After several generations, the result is often one or more highly fit chromosomes in the population. It should be noted that the previous description leaves out many important details. For example, selection can be implemented in different ways—it can eliminate the least fit 50% of the population and replicate each remaining individual once, it can replicate individuals in direct proportion to their fitness (fitness-proportionate selection), or it can scale the fitness and replicate individuals in direct proportion to their scaled fitnesses. For implementation details such as these, see [42].

Introducing a population-based algorithm with crossover and inversion was a major innovation. Just as significant is the theoretical foundation Holland developed based on the notion of "schemata" [42, 55]. Until recently, this theoretical foundation has been the basis of almost all subsequent theoretical work on GAs, although the usefulness of this notion has been debated (see, e.g., [45]). Holland's work was the first attempt to put computational evolution on a firm theoretical footing.

GAs in various forms have been applied to many scientific and engineering problems, including the following:

- *Optimization.* GAs have been used in a wide variety of optimization tasks, including numerical optimization (e.g., [63]), and combinatorial optimization problems such as circuit design and job shop scheduling.
- *Automatic programming.* GAs have been used to evolve computer programs for specific tasks (e.g., [69]) and to design other computational structures, for example, cellular automata [80] and sorting networks [52].
- *Machine and robot learning.* GAs have been used for many machine-learning applications, including classification and prediction tasks such as the prediction of dynamical systems [75], weather prediction [92], and prediction of protein structure (e.g., [95]). GAs have also been used to design neural networks (e.g., [15, 25, 47, 48, 67, 77, 81, 94, 105]) to evolve rules for learning classifier systems (e.g., [54, 57]) or symbolic production systems (e.g., [46]), and to design and control robots (e.g., [29, 31, 50]). For an overview of GAs in machine learning, see De Jong [64, 65].
- *Economic models.* GAs have been used to model processes of innovation, the development of bidding strategies, and the emergence of economic markets (e.g., [3–5, 58]).
- *Immune system models.* GAs have been used to model various aspects of the natural immune system [17, 38], including somatic mutation during an individual's lifetime and the discovery of multi-gene families during evolutionary time.
- *Ecological models.* GAs have been used to model ecological phenomena such as biological arms races, host-parasite coevolution, symbiosis, and resource flow in ecologies (e.g., [11, 12, 26, 28, 52, 56, 61, 70, 71, 83, 87, 88, 101]).
- *Population genetics models.* GAs have been used to study questions in population genetics, such as "under what conditions will a gene for recombination be evolutionarily viable?" (e.g., [16, 35, 74, 93]).
- *Interactions between evolution and learning.* GAs have been used to study how individual learning and species evolution affect one another (e.g., [1, 2, 13, 37, 53, 76, 82, 84, 102, 103]).
- *Models of social systems.* GAs have been used to study evolutionary aspects of social systems, such as the evolution of cooperation [7, 8, 73, 78, 79], the evolution of communication (e.g., [72, 104]), and trail-following behavior in ants (e.g., [27, 68]).

This list is by no means exhaustive, but it gives a flavor of the kinds of things for which GAs have been used, both for problem solving and for modeling. The range of GA applications continues to increase.

In recent years, algorithms that have been termed *genetic algorithms* have taken many forms and in some cases bear little resemblance to Holland's original formulation. Researchers have experimented with different types of representations, crossover and mutation operators, special-purpose operators, and approaches to reproduction and selection. However, all of these methods have a "family resemblance" in that they take some inspiration from biological evolution and from Holland's original GA. A new term,

Evolutionary Computation, has been introduced to cover these various members of the GA family, evolutionary programming, and evolution strategies [66].

In the following sections we describe a number of examples illustrating the use of GAs in Artificial Life. We do not attempt to give an exhaustive review of the entire field of GAs or even that subset relevant to Artificial Life, but rather concentrate on some highlights that we find particularly interesting. We have provided a more complete set of pointers to the GA and Artificial-Life literature in the "Suggested Reading" section at the end of this article.

3 Interactions Between Learning and Evolution

Many people have drawn analogies between learning and evolution as two adaptive processes—one taking place during the lifetime of an organism, and the other taking place over the evolutionary history of life on earth. To what extent do these processes interact? In particular, can learning that occurs over the course of an individual's lifetime guide the evolution of that individual's species to any extent? These are major questions in evolutionary psychology. GAs, often in combination with neural networks, have been used to address these questions. Here we describe two artificial-life systems designed to model interactions between learning and evolution, and in particular the "Baldwin effect."

3.1 The Baldwin effect

Learning during one's lifetime does not directly affect one's genetic makeup; consequently, things learned during an individual's lifetime cannot be transmitted directly to its offspring. However, some evolutionary biologists (e.g., [98]) have discussed an indirect effect of learning on evolution, inspired by ideas about evolution due to Baldwin [10] (among others). The idea behind the so-called Baldwin effect is that if learning helps survival, then organisms best able to learn will have the most offspring and increase the frequency of the genes responsible for learning. If the environment is stable so that the best things to learn remain constant, then this can lead indirectly to a genetic encoding of a trait that originally had to be learned. In short, the capacity to acquire a certain desired trait allows the learning organism to survive preferentially and gives genetic variation the possibility of independently discovering the desired trait. Without such learning, the likelihood of survival—and, thus, the opportunity for genetic discovery—decreases. In this indirect way, learning can affect evolution, even if what is learned cannot be transmitted genetically.

3.2 Capturing the Baldwin Effect in a Simple Model

Hinton and Nowlan [53] used a GA to model the Baldwin effect. Their goal was to demonstrate this effect empirically and to measure its magnitude, using the simplest possible model. A simple neural-network learning algorithm modeled learning, and the GA played the role of evolution, evolving a population of neural networks with varying learning capabilities. In the model, each individual is a neural network with 20 potential connections. A connection can have one of three values: "present," "absent," and "learnable." These are specified by "1," "0," and "?," respectively, where each ? connection can be set during the learning phase to 1 or 0. There is only one correct setting for the connections (i.e., only one correct set of 1s and 0s). The problem is to find this single correct set of connections. This will not be possible for networks that have incorrect fixed connections (e.g., a 1 where there should be a 0), but those networks that have correct settings in all places except where there are ?s have the capacity to learn the correct settings. This is a "needle-in-a-haystack" search problem, because there is only one correct setting in a space of 2^{20} possibilities. However,

allowing learning to take place changes the shape of the fitness landscape, changing the single spike to a smoother "zone of increased fitness," within which it is possible to learn the correct connections.

Hinton and Nowlan used the simplest possible "learning" method: random guessing. On each learning trial, a network guesses a 1 or 0 at random for each of its learnable connections. This method has little to do with the usual notions of neural-network learning. Hinton and Nowlan presented this model in terms of neural networks so as to keep in mind the possibility of extending the example to more standard learning tasks and methods.

In the GA population, each network is represented by a string of length 20 over the alphabet {0, 1, ?}, denoting the settings on the network's connections. Each individual is given 1,000 learning trials. On each learning trial, the individual tries a random combination of settings for the ?s. The fitness is an inverse function of the number of trials needed to find the correct solution. An individual that already has all of its connections set correctly has the highest possible fitness, and an individual that never finds the correct solution has the lowest possible fitness. Hence, a tradeoff exists between efficiency and flexibility: having many ?s means that, on average, many guesses are needed to arrive at the correct answer, but the more connections that are fixed, the more likely it is that one or more of them will be fixed incorrectly, meaning that there is no possibility of finding the correct answer.

Hinton and Nowlan's experiments showed that learning during an individual's "lifetime" does guide evolution by allowing the mean fitness of the population to increase. This increase is due to a Baldwin-like effect: Those individuals that are able to learn the task efficiently tend to be selected to reproduce, and crossovers among these individuals tend to increase the number of correctly fixed alleles, increasing the learning efficiency of the offspring. With this simple form of learning, evolution could discover individuals with all of their connections fixed correctly, and such individuals were discovered in these experiments. Without learning, the evolutionary search never discovered such an individual.

To summarize, learning allows genetically coded partial solutions to get partial credit, rather than the all-or-nothing reward that an organism would get without learning. A common claim for learning is that it allows an organism to respond to unpredictable aspects of an environment—aspects that change too quickly for evolution to track genetically. Although this is clearly one benefit of learning, the Baldwin effect is different: It says that learning helps organisms adapt to genetically predictable, but difficult, aspects of the environment, and that learning indirectly helps these adaptations become genetically fixed. Consequently, the Baldwin effect is important only on fitness landscapes that are hard to search by evolution alone, such as the needle-in-a-haystack example given by Hinton and Nowlan.

As Hinton and Nowlan point out, the "learning" mechanism used in their experiments—random guessing—is completely unrealistic as a model of learning. Hinton and Nowlan point out that "a more sophisticated learning procedure only strengthens the argument for the importance of the Baldwin effect" ([53], p. 500). This is true insofar as a more sophisticated learning procedure would, for example, further smooth the original "needle-in-the-haystack" fitness landscape in Hinton and Nowlan's learning task. However, if the learning procedure were *too* sophisticated—that is, if learning the necessary trait were too easy—then there would be little selection pressure for evolution to move from the ability to learn the trait to a genetic encoding of that trait. Such tradeoffs occur in evolution and can be seen even in Hinton and Nowlan's simple model. Computer simulations such as theirs can help us to understand and to measure such tradeoffs. More detailed analyses of this model were performed by Belew [13] and Harvey [49].

3.3 Evolutionary Reinforcement Learning (ERL)

A second computational demonstration of the Baldwin effect was given by Ackley and Littman [1]. In their Evolutionary Reinforcement Learning (ERL) model, adaptive individuals ("agents") move randomly on a two-dimensional lattice, encountering food, predators, hiding places, and other types of entities. Each agent's state includes the entities in its visual range, the level of its internal energy store, and other parameters. Each agent possesses two feed-forward neural networks: (1) an evaluation network that maps the agent's state at time t to a number representing how good that state is, and (2) an action network that maps the agent's state at time t to the action it is to take on that time step. The only possible actions are moving from the current lattice site to one of the four neighboring sites, but actions can result in eating, being eaten, and other less radical consequences. The architectures of these two networks are the same for all agents, but the weights on the links can vary between agents. The weights on a given agent's evaluation network are fixed from birth—this network represents innate goals and desires inherited from the agent's ancestors (e.g., "being near food is good"). The weights on the action network change over the agent's lifetime according to a reinforcement-learning algorithm.

An agent's genome encodes the weights for the evaluation network and the initial weights for the action network. Agents have an internal energy store (represented by a real number) that must be kept above a certain level to prevent death; this is accomplished by eating food that is encountered as the agent moves from site to site on the lattice. An agent must also avoid predators, or it will be killed. An agent can reproduce once it has enough energy in its internal store. Agents reproduce by cloning their genomes (subject to mutation). In addition to cloning, two spatially nearby agents can together produce offspring via crossover. There is no "exogenous" *a priori* fitness function for evaluating a genome as there was in Hinton and Nowlan's model and in most engineering applications of GAs. Instead, the fitness of an agent (as well as the rate at which a population turns over) is "endogenous": it emerges from many actions and interactions over the course of the agent's lifetime. This feature distinguishes many GAs used in artificial-life models from engineering applications.

At each time step t in an agent's life, the agent evaluates its current state, using its evaluation network. This evaluation is compared with the evaluation it produced at $t - 1$ with respect to the previous action, and the comparison gives a reinforcement signal used in modifying the weights in the action network. The idea here is for agents to learn to act in ways that will improve the current state. After this learning step, the agent's modified action network is used to determine the next action to take.

Ackley and Littman observed many interesting phenomena in their experiments with this model. The main emergent phenomena they describe are a version of the Baldwin effect and an effect they call "shielding." Here we will describe the former; see Ackley and Littman [1] for details on other phenomena. They compared the results of three different experiments: (1) EL: both evolution of populations and learning in individual agents took place, (2) E: evolution of populations took place but there was no individual learning, and (3) L: individual learning took place but there was no evolution. The statistic that Ackley and Littman measured was roughly the average time until the population became extinct, averaged over many separate runs. They found that the best performance (longest average time to extinction) was achieved with EL populations, closely followed by L populations, and with E populations trailing far behind. More detailed analysis of the EL runs revealed that with respect to certain behaviors, the relative importance of learning and evolution changed over the course of a run. In particular, Ackley and Littman looked at the genes related to food-approaching behavior for both the evaluation and action networks. They found that in earlier generations, the genes encoding evaluation of food proximity (e.g., "being near food is good")

remained relatively constant across the population, while the genes encoding initial weights in the action network were more variable. This indicated the importance of maintaining the goals for the learning process and, thus, the importance of learning for survival. However, later in the run the evaluation genes were more variable across the population, whereas the genes encoding the initial weights of the action network remained more constant. This indicated that inherited behaviors were more significant than learning during this phase. Ackley and Littman interpreted this as a version of the Baldwin effect. Initially, it is necessary for agents to *learn* to approach food; thus, maintaining the explicit knowledge that "being near food is good" is essential for the learning process to take place. Later, the genetic knowledge that being near food is good is superseded by the genetically encoded behavior to "approach food if near," so the evaluation knowledge is not as necessary. The initial ability to learn the behavior is what allows it to eventually become genetically coded.

This effect has not been completely analyzed, nor has the strength of the effect been determined. Nevertheless, results such as these, and those of Hinton and Nowlan's experiments, demonstrate the potential of artificial-life modeling: biological phenomena can be studied with controlled computational experiments whose natural equivalent (e.g., running for thousands of generations) is not possible or practical. And when performed correctly, such experiments can produce new evidence for and new insight into these natural phenomena. The potential benefits of such work are not limited to understanding natural phenomena. A growing community of GA researchers is studying ways to apply GAs to optimize neural networks to solve practical problems—a practical application of the interaction between learning and evolution. A survey of this work is given in Schaffer, Whitley, and Eshelman [94]. Other researchers are investigating the benefits of adding "Lamarckian" learning to the GA and have found in some cases that it leads to significant improvements in GA performance [2, 44].

4 Ecosystems and Evolutionary Dynamics

Another major area of artificial-life research is modeling ecosystem behavior and the evolutionary dynamics of populations. (Ackley and Littman's work described earlier could fit into this category as well.) Here we describe two such models that use GAs: Holland's Echo system, meant to allow a large range of ecological interactions to be modeled, and Bedau and Packard's Strategic Bugs system, for which a measure of evolutionary activity is defined and studied. As in the ERL system, both Echo and Strategic Bugs illustrate the use of endogenous fitness.

4.1 Echo

Echo is a model of ecological systems formulated by Holland [55, 56, 62]. Echo models ecologies in the same sense that the GA models population genetics [56]. It abstracts away virtually all of the physical details of real ecological systems and concentrates on a small set of primitive agent–agent and agent–environment interactions. The extent to which Echo captures the essence of real ecological systems is still largely undetermined, yet it is significant because of the generality of the model and its ambitious scope. The goal of Echo is to study how simple interactions among simple agents lead to emergent high-level phenomena such as the flow of resources in a system or cooperation and competition in networks of agents (e.g., communities, trading networks, or arms races). Echo extends the GA in several important ways: resources are modeled explicitly in the system, individuals (called agents) have a geographical location that affects their (implicit) fitness, certain types of interactions between agents are built into the system (e.g., trade, combat, and mating), and fitness is endogenous.

Similar to Ackley and Littman's ERL model, Echo consists of a population of agents distributed on a set of sites on a lattice. Many agents can cohabit the same site, and there is a measure of locality within each site. Also distributed on the lattice are different types of renewable resources; each type of resource is encoded by a letter (e.g., "a," "b," "c," "d"). Different types of agents use different types of resources and can store these resources (the letters) internally.

Agents interact by mating, trading, or fighting. Trading and fighting result in the exchange of internal resources between agents, and mating results in an offspring whose genome is a combination of those of the parents. Agents also self-reproduce (described later), but mating is a process distinct from replication. Each agent has a particular set of rules that determines its interactions with other agents (e.g., which resources it is willing to trade, the conditions under which it will fight, etc.). "External appearance" can also be coded in these rules as a string tag visible to other agents. This allows the possibility of the evolution of social rules and potentially of mimicry, a phenomenon frequently observed in natural ecosystems. The interaction rules use string matching, and it is therefore easy to encode the strings used by the rules onto the genome.

Each agent's genome encodes the details of the rules by which it interacts (e.g., the conditions under which the rules are applied) and the types of resources it requires. As in many other artificial-life models (e.g., ERL and the Strategic Bugs model described below), Echo has no explicit fitness function guiding selection and reproduction. Instead, an agent reproduces when it accumulates sufficient resources to make an exact copy of its genome. For example, if an agent's genome consists of 25 a's, 13 b's, and 50 c's, then it would have to accumulate in its internal storage at least 25 a's, 13 b's, and 50 c's before cloning itself. As is usual in a GA, cloning is subject to a low rate of mutation, and, as was mentioned earlier, genetic material is exchanged through mating.

In preliminary simulations, the Echo system has demonstrated surprisingly complex behavior (including something resembling a biological "arms race" in which two competing species develop progressively more complex offensive and defensive combat strategies), ecological dependencies among different species (e.g., a symbiotic "ant-caterpillar-fly" triangle), and sensitivity (in terms of the number of different phenotypes) to differing levels of renewable resources [55].

Some possible directions for future work on Echo include (1) studying the evolution of external tags as mechanisms for social communication; (2) extending the model to allow the evolution of "metazoans"—connected communities of agents that have internal boundaries and reproduce as a unit; this capacity will allow for the study of individual agent specialization and the evolution of multicellularity; (3) studying the evolutionary dynamics of schemata in the population; and (4) using the results from (3) to formulate a generalization of the well-known Schema Theorem based on endogenous fitness [56]. The last is a particularly important goal, because there has been very little mathematical analysis of artificial-life simulations in which fitness is endogenous.

4.2 Measuring Evolutionary Activity

How can we decide if an observed system is evolving? And how can we measure the rate of evolution in such a system? Bedau and Packard [11] developed an artificial-life model, called "Strategic Bugs," to address these questions. Their model is simpler than both ERL and Echo. The Strategic Bugs world is a two-dimensional lattice, containing only adaptive agents ("bugs") and food. The food supply is renewable; it is refreshed periodically and distributed randomly across the lattice. Bugs survive by finding and eating food, storing it in an internal reservoir until they have enough energy to reproduce. Bugs use energy from their internal reservoir in order to move. A bug dies when

its internal reservoir is empty. Thus, bugs have to find food continually in order to survive.

Each bug's behavior is controlled by an internal look-up table that maps sensory data from the bug's local neighborhood to a vector giving the direction and distance of the bug's next foray. For example, one entry might be, "If more than 10 units of food are two steps to the northeast and the other neighboring sites are empty, move two steps to the northeast." This look-up table is the bug's "genetic material," and each entry is a gene. A bug can reproduce either asexually, in which case it passes on its genetic material to its offspring with some low probability of mutation at each gene, or sexually, in which case it mates with a spatially adjacent bug, producing offspring whose genetic material is a combination of that of the parents, possibly with some small number of mutations.

Bedau and Packard wanted to define and measure the degree of "evolutionary activity" in this system over time, where evolutionary activity is defined informally as "the rate at which useful genetic innovations are absorbed into the population." Bedau and Packard assert that "persistent usage of new genes is what signals genuine evolutionary activity," because evolutionary activity is meant to measure the degree to which useful new genes are discovered and persist in the population.

To measure evolutionary activity, Bedau and Packard began by keeping statistics on gene usage for every gene that appeared in the population. Recall that in the Strategic Bugs model, a bug's genome is represented as a look-up table, and a gene is simply an entry in the table—an input/action pair. Each gene is assigned a counter, initialized to 0, which is incremented every time the gene is used—that is, every time the specified input situation arises and the specified action is taken. When a parent passes on a gene to a child through asexual reproduction or through crossover, the value of the counter is passed on as well and remains with the gene. The only time a counter is initialized to 0 is when a new gene is created through mutation. In this way, a gene's counter value reflects the usage of that gene over many generations. When a bug dies, its genes (and their counters) die with it.

Bedau and Packard [11] plot, for each time step during a run, histograms of the number of genes in the population displaying a given usage value (i.e., a given counter value). These histograms display "waves of activity" over time, showing that clusters of genes are continually being discovered that persist in usage over time—in other words, that the population is continually finding and exploiting new genetic innovations. This is precisely Bedau and Packard's definition of evolution, and according to them, as long as the waves continue to occur, it can be said that the population is continuing to evolve. Bedau and Packard define a single number, the *evolutionary activity* at a given time, $A(t)$, that roughly measures the degree to which the population is acquiring new and useful genetic material at time t—in short, whether or not such activity waves are occurring at time t and what their characteristics are. If $A(t)$ is positive, then evolution is occurring at time t. Claiming that life is a property of populations and not of individual organisms, Bedau and Packard ambitiously propose A(t) as a test for life in a system—if A(t) is positive, then the system is exhibiting life at time t. Bedau, Ronneburg, and Zwick [12] have extended this work to propose several measures of population diversity and to measure them and characterize their dynamics in the context of the Strategic Bugs model.

The important contribution of Bedau and Packard's paper is the attempt to define a macroscopic quantity such as evolutionary activity. It is a first step at such a definition, and the particular definition of gene usage is no doubt too specific to the Strategic Bugs model, in which the relationship between genes and behavior is completely straightforward. In more realistic models it will be considerably harder to define such quantities. However, the formulation of macroscopic measures of evolution and adaptation, as

well as descriptions of the microscopic mechanisms by which the macroscopic quantities emerge, is essential if artificial life is to be made into an explanatory science and if it is to contribute significantly to real evolutionary biology.

5 Learning Classifier Systems

Learning classifier systems [57] are one of the earliest examples of how GAs have been incorporated into models of living systems, in this case cognitive systems. Classifier systems have been used as models of stimulus-response behavior and of more complex cognitive processes. Classifier systems are based on three principles: learning, intermittent feedback from the environment, and hierarchies of internal models that represent the environment. Classifier systems have been used to model a variety of "intelligent" processes, such as how people behave in economic and social situations (playing the stock market, obeying social norms, etc.), maze running by rats, and categorization tasks.

Like neural networks, classifier systems consist of a parallel machine (most often implemented in software) and learning algorithms that adjust the configuration of the underlying machine over time. Classifier systems differ from neural networks in the details of the parallel machine, referred to as the *internal performance system*, and in the details of the learning algorithms. Specifically, the classifier system machine is more complex than most neural networks, computing with quantities called "messages" and controlling its state with if-then rules that specify patterns of messages. The GA is used to discover useful rules, based on intermittent feedback from the environment and an internal credit-assignment algorithm called the *bucket brigade*. Thus, a classifier system consists of three layers, with the performance system forming the lowest level. At the second level, the bucket-brigade learning algorithm manages credit assignment among competing classifiers. It plays a role similar to that of back-propagation in neural networks. Finally, at the highest level are genetic operators that create new classifiers.

Associated with each classifier is a parameter called its strength. This measure reflects the utility of that rule, based on the system's past experience. The bucket-brigade algorithm is the mechanism for altering each rule's strength. The algorithm is based on the metaphor of an economy, with the environment acting both as the producer of raw materials and the ultimate consumer of finished goods, and each classifier acting as an intermediary in an economic chain of production. Using the bucket brigade, a classifier system is able to identify and use the subset of its rule base that has proven useful in the past. However, a classifier system's initial rule base usually will not contain all of the classifiers necessary for good performance. The GA interprets a classifier's strength as a measure of its fitness, and periodically (after the strengths have stabilized under the bucket brigade), the GA deletes rules that have not been useful or relevant in the past (those with low strength) and generates new rules by modifying existing high-strength rules through mutation, crossover, and other special-purpose operators. Similarly to conventional GAs, these deletions and additions are all performed probabilistically. Under the definition of induction as "all inferential processes that expand knowledge in the face of uncertainty" [57, p. 1], the GA plays the role of an inductive mechanism in classifier systems.

An important motivation in the formulation of classifier systems was the principle that inductive systems need the ability to construct internal models. Internal models should allow a system to generate predictions even when its knowledge of the environment is incomplete or incorrect, and further, to refine its internal model as more information about the environment becomes available. This leads naturally to the idea of a default hierarchy in which a system can represent high-level approximations, or defaults, based

on early information, and, over time, refine the defaults with more specific details and exceptions to rules. In classifier systems, default hierarchies are represented using clusters of rules of different specificities. In [57], the concept of a "quasi-morphism" is introduced to describe this modeling process formally.

There have been several modeling efforts based on learning classifier systems, including [19, 20, 32, 90, 91, 106, 107]. Each of these is a variation on the standard classifier system as described earlier, but each of the variations captures the major principles of classifier systems. For example, Riolo [90] used a classifier system to model the kind of latent learning and look-ahead behavior of the type observed in rats. For this work, Riolo designed a simple maze, similar to those in latent-learning experiments on rats. The maze has one start point and several endpoints. At each endpoint there is a box, which may or may not be filled with food, and the various endpoint boxes may or may not be distinguishable (e.g., by color) from one another. In these kinds of experiments, the procedure is roughly as follows: (1) before food is placed in the boxes, nonhungry rats are placed in the maze and allowed to explore; (2) the rats are not fed for 24 hours; (3) the rats are placed in the maze (at one of the endpoints) and allowed to eat from one of the boxes; and (4) the rats are placed at the start location of the maze, and their behavior is observed. If the boxes are distinguishable, then the rats reliably choose the path through the maze leading to the box from which they ate.

Riolo makes several points about these experiments: (1) in the "pre-reward" phase, the rats learn the structure of the maze without explicit rewards; (2) they learn to use an internal model to perform a look-ahead search that allows them to predict which box was in which part of the maze; (3) the rats are able to use this look-ahead search once they associate food with a particular box; and (4) this type of inference cannot be made by a simple reactive (stimulus-response) system. It is commonly believed that the task requires the use of internal models and look-ahead prediction.

To model these experiments using a classifier system, Riolo augmented the basic classifier system model to include a look-ahead component. The extensions included (1) allowing the classifier system to iterate several cycles of its performance system (the rule base) before choosing an action, in effect "running" an internal model before acting; (2) choosing special-purpose genetic operators to coordinate the internal model-building (i.e., to distinguish predictions from suggested actions); and (3) using three different kinds of strength to measure the utility of rules (to measure predictive ability vs. real-time ability, to produce a reward from an action, and to measure long-term vs. short-term utility). With these modifications, the classifier system achieved results comparable with the latent-learning results reported for rats. Further, the classifier system with the look-ahead component outperformed the unmodified version significantly. Riolo's experiment is one of the best demonstrations to date of the necessity of internal models for classifier systems to succeed on some tasks.

6 Immune Systems

Immune systems are adaptive systems in which learning takes place by evolutionary mechanisms similar to biological evolution. Immune systems have been studied by the artificial-life community both because of their intrinsic scientific interest and because of potential applications of ideas from immunology to computational problems (e.g., [17]). The immune system is capable of recognizing virtually any foreign cell or molecule. To do this, it must distinguish the body's own cells and molecules that are created and circulated internally (estimated to consist of on the order of 10^5 different proteins) from those that are foreign. It has been estimated that the immune system is capable of recognizing on the order of 10^{16} different foreign molecules [60]. From a

pattern-recognition perspective, these are staggering numbers, particularly when one considers that the human genome, which encodes the "program" for constructing the immune system, only contains about 10^5 genes, and further, that the immune system is distributed throughout the body with no central organ to control it.

Different approaches to modeling the immune system have included differential-equation-based models (e.g., see [85, 86]), cellular-automata models [24], classifier systems [33], and GAs [38]. In the last, GAs are used to model both somatic mutation (the process by which antibodies are evolved during the lifetime of an individual to match a specific antigen) and the more traditional type of evolution over many individual lifetimes of variable-, or V-, region gene libraries (the genetic material that codes for specific receptors).

The GA models of Forrest, Javornik, Smith, and Perelson [38] are based on a universe in which antigens (foreign material) and antibodies (the cells that perform the recognition) are represented by binary strings. More precisely, the binary strings are used to represent receptors on B cells and T cells and epitopes on antigens, although we refer to these (loosely) as antibodies and antigens. Recognition in the natural immune system is achieved by molecular binding—the extent of the binding being determined by molecular shape and electrostatic charge. The complex chemistry of antigen recognition is highly simplified in the binary immune system and modeled as string matching. The GA is used to evolve populations of strings that match specific antigens well. For strings of any significant length, a perfect match is highly improbable, so a partial matching rule is used that rewards more specific matches (i.e., matches on more bits) over less specific ones. This partial matching rule reflects the fact that the immune system's recognition capabilities need to be fairly specific in order to avoid confusing self molecules with foreign molecules.

In the models of Forrest et al., one population of antibodies and one of antigens is created, each randomly. For most experiments, the antigen population is held constant, and the antibody population is evolved under the GA. However, in some experiments the antigen population is allowed to coevolve with the antibodies (i.e., antigens evolve away from the antibodies while the antibodies are evolving toward the antigens). Antigens are "presented" to the antibody population sequentially (again, by analogy with the natural immune system), and high-affinity antibodies (those that match at many bit positions) have their fitnesses increased.

This binary immune system has been used to study several different aspects of the immune system, including (1) its ability to detect common patterns (schemas) in the noisy environment of randomly presented antigens [38]; (2) its ability to discover and maintain coverage of the diverse antigen population [99]; and (3) its ability to learn effectively, even when not all antibodies are expressed and not all antigens are presented [51]. This last experiment is particularly relevant to the more general question of how selection pressures operating only at the global, phenotypic level can produce appropriate low-level, genetic structures. The question is most interesting when the connection between phenotype and genotype is more than a simple, direct mapping. The multigene families (V-region libraries) of the immune system provide a good subject for experimentation from this point of view—the phenotype is not a direct mapping from the genotype, but the connection is simple enough that it can be studied analytically. In [51], all antigens were exactly 64 bits. The V-region library was modeled as a set of four libraries, each with eight entries of length 16 (producing a genome with 512 bits). Antibodies were expressed by randomly choosing one entry from each library and concatenating them together to form one 64-bit antibody.

Recent work on the kind of genotype–phenotype relations that might be expected between a sequence (e.g., an RNA sequence) and its corresponding higher-order structure (e.g., its secondary structure) may also apply to modeling the immune system

		Player B	
		Cooperate	Defect
Player A	Cooperate	3, 3	0, 5
	Defect	5, 0	1, 1

Figure 1. The payoff matrix for the Prisoner's Dilemma. The pairs of numbers in each cell give the respective payoffs for players A and B in the given situation.

[59]. For example, the interaction between the immune system and a rapidly evolving pathogen can be regarded as a system with rapidly changing fitness criteria at the level of the secondary structure. Yet, the immune system and pathogen are both coevolving through mutations at the genetic level. In a coevolutionary system such as this, the populations evolve toward relatively uncorrelated parts of the phenotype landscape where mutations have a relatively large effect on the secondary structure, thus facilitating the process of continuous adaptation itself. This is a similar point to that raised in [51]. The idea of exploiting variations in the phenotype through mutations at the genetic level is a recurring theme in evolution, and the immune system provides a clear example of where such exploitation might occur.

7 Social Systems

Understanding and modeling social systems, be they insect colonies or human societies, has been a focus of many artificial-life researchers. GAs have played a role in some of these models, particularly those modeling the evolution of cooperation. Here we describe how the GA was used to evolve strategies for interaction in the context of the Prisoner's Dilemma.

The Prisoner's Dilemma (PD) is a simple two-person game that has been studied extensively in game theory, economics, and political science because it can be seen as an idealized model for real-world phenomena such as arms races [6]. On a given turn, each player independently decides whether to "cooperate" or "defect." The game is summarized by the payoff matrix shown in Figure 1. If both players cooperate, they each get three points. If player A defects and player B cooperates, then player A gets five points, and player B gets zero points; vice versa if the situation is reversed. Finally, if both players defect, they each get one point. What is the best strategy to take? If there is only one turn to be played, then clearly the best strategy is to defect: the worst consequence for a defector is to get one point and the best is to get five points, which are better than the worst score and the best score, respectively, for a cooperator. The dilemma is that if the game is iterated, that is, if two players play several turns in a row, the strategy of always defecting will lead to a much lower total payoff than the players would get if they both cooperated. How can reciprocal cooperation be induced? This question takes on special significance when the notions of "cooperating" and "defecting" correspond to actions in the real world, such as a real-world arms race.

Axelrod [6] has studied the PD and related games extensively. Early work, including the results of two tournaments that played pairs of human-designed strategies against each other, suggested that the best strategy for playing the iterated PD is one of the simplest: TIT FOR TAT. TIT FOR TAT cooperates on the first move and then, on subsequent moves, does whatever the other player did last. That is, it offers cooperation

and then reciprocates it, but if the other player defects, TIT FOR TAT will retaliate with a defection.

Axelrod [8] performed a series of experiments to see if a GA could evolve strategies to play this game successfully. Strategies were encoded as look-up tables, with each entry (C or D) being the action to be taken given the outcomes of three previous turns. In Axelrod's first experiment, the evolving strategies were played against eight human-designed strategies, and the fitness of an evolving strategy was a weighted average of the scores against each of the eight fixed strategies. Most of the strategies that evolved were similar to TIT FOR TAT, having many of the properties that make TIT FOR TAT successful. Strikingly, the GA occasionally found strategies that scored substantially higher than TIT FOR TAT.

It is not correct to conclude that the GA evolved strategies that are "better" than any human-designed strategy. The performance of a strategy depends very much on its environment, that is, the other strategies with which it is playing. Here the environment was fixed, and the highest-scoring strategies produced by the GA were ones that discovered how to exploit specific weaknesses of the eight fixed strategies. It is not necessarily true that these high-scoring strategies would also score well in some other environment. TIT FOR TAT is a generalist, whereas the highest-scoring evolved strategies were more specialized to their given environment. Axelrod concluded that the GA is good at doing what evolution often does: developing highly specialized adaptations to specific characteristics of the environment.

To study the effects of a dynamic environment, Axelrod carried out another experiment in which the fitness was determined by allowing the strategies in the population to play with each other rather than with the fixed set of eight strategies. The environment changes from generation to generation because the strategies themselves are evolving. At each generation, each strategy played an iterated PD with the other members of the population, and its fitness was the average score over all these games. In this second set of experiments, Axelrod observed the following phenomenon: the GA initially evolves uncooperative strategies, because strategies that tend to cooperate early on do not find reciprocation among their fellow population members and, thus, tend to die out. But after about 10–20 generations, the trend starts to reverse: the GA discovers strategies that reciprocate cooperation and that punish defection (i.e., variants of TIT FOR TAT). These strategies do well with each other and are not completely defeated by other strategies, as were the initial cooperative strategies. The reciprocators score better than average, so they spread in the population, resulting in more and more cooperation and increasing fitness.

Lindgren [70] performed a series of experiments similar to Axelrod's second experiment but included the possibility of noise, in which players can make mistakes in following their strategies. He also allowed a more open-ended kind of evolution in which a "gene duplication" operator allowed the amount of memory available to a given strategy to increase. He observed some very interesting evolutionary dynamics, including periods of relative stasis with one or two strategies fairly stable in the population, punctuated by mass extinction events. Other work using computational evolution to discover PD strategies in the presence of noise or imperfect information about the past (both making the PD a more realistic model of social or political interactions) has been done by Miller [79] and Marks [73], among others.

8 Open Problems and Future Directions

In the previous sections we have briefly described some representative examples of artificial-life projects that use GAs in a significant way. These examples, and many others that we do not have space to discuss, point the way to several open problems

in GAs. Some of these are quite technical (e.g., questions about genetic operators and representations), and some are more general questions, relevant to many areas of Artificial Life.

It is difficult to distinguish between "yet another cute simulation" and systems that teach us something important and general, either about how to construct artificial life or about the natural phenomena that they model. We suggest that artificial-life research should address at least one of these two criteria and that it is important to be explicit about what any specific system teaches us that was not known before. This is a much more difficult task than may be readily appreciated, so difficult in fact that we consider it an open problem to develop adequate criteria and methods for evaluating artificial-life systems.

On the modeling side it can be very difficult to relate the behavior of a simulation quantitatively to the behavior of the system it is intended to model. This is because the level at which Artificial-Life models are constructed is often so abstract that they are unlikely to make numerical predictions. In GAs, for example, all of the biophysical details of transcription, protein synthesis, gene expression, and meiosis have been stripped away. Useful Artificial-Life models, however, may well reveal general conditions under which certain qualitative behaviors arise, or critical parameters in which a small change can have a drastic effect on the behavior of the system. What is difficult is to distinguish between good qualitative modeling and simulations that are only vaguely suggestive of natural phenomena.

More specific to GAs is the central question of representation. For any given environment or problem domain, the choice of which features to represent on the genotype and how to represent them is crucial to the performance of the GA (or any other learning system). The choice of system primitives (in the case of GAs, the features that comprise the genotype) is a design decision that cannot be automated. GAs typically use low-level primitives such as bits, which can be very far removed from the natural representation of environmental states and control parameters. For this reason, the representation problem is especially important for GAs, both for constructing artificial life and in modeling living systems.

Although the representation problem has been acknowledged for many years, there have been surprisingly few innovative representations, the recent work on genetic programming [69] and messy GAs [43] being notable exceptions. In genetic programming, individuals are represented as S-expressions—small programs written in a subset of LISP. Although S-expressions can be written as linear strings, they are naturally viewed as trees, and the genetic operators operate on trees. Crossover, for example, swaps subtrees between S-expressions. Messy GAs were developed by Goldberg, Korb, and Deb [43] to allow variable-length strings that can be either over- or underspecified with respect to the problem being solved. This allows the GA to manipulate short strings early in a run, and over time, to combine short, well-tested building blocks into longer, more complex strings. New versions of the crossover operator (e.g., uniform crossover [100]) can reduce the inherent bias in standard crossover of breaking up correlated genes that are widely separated on the chromosome (referred to as "positional bias"). These approaches are promising in some cases, especially because the strong positional dependence of most current representations is an artifact introduced by GAs. In natural genetic systems, one gene (approximately) codes for one protein regardless of where it is located, although the expression of a gene (when the protein is synthesized) is indirectly controlled by its location. In spite of the foregoing, the vast majority of current GA implementations use a simple binary alphabet linearly ordered along a single haploid string. It should be noted that researchers interested in engineering applications have long advocated the use of simple "higher-cardinality alphabets," including, for example, real numbers as alleles [30]. Given the fact that GA

performance is heavily dependent on the representation chosen, this lack of diversity is surprising.

The representation issues described earlier primarily address the question of how to engineer GAs. Moving away from this question toward more realistic models of evolution are more extended mappings between the genotypic representation and the phenotype. Buss [23], among others, has pointed out that the principle of evolution by natural selection is applicable at many levels besides that of the individual, and in particular, that natural selection controls development (e.g., embryology) that interacts with selection at the level of the individual. Related to this point, and to the observation that evolution and learning can interact, are several recent studies of GAs that include a "development" cycle, which translates the genotype through a series of steps into the phenotype. The most common example of this is to let the genotype specify a grammar (as in L-systems). The grammar is then used to produce a legal object in the language it specifies (the development step), and this string (the phenotype) is then evaluated by the fitness function. Examples of this exploratory work include Belew [14], Gruau [47], Kitano [67], and Wilson [108]. Although this work is only a crude approximation of development in living systems, it is an important first step and represents a promising avenue for future research.

Related to the question of representation is the choice of genetic operators for introducing variation into a population. One reason that binary linearly ordered representations are so popular is that the standard mutation and crossover operators can be applied in a problem-independent way. Other operators have been experimented with in optimization settings, but no new general-purpose operators have been widely adopted since the advent of GAs. Rather, the inversion operator, included in the original proposals for theoretical reasons, has been largely abandoned. We believe it deserves more study. In addition, during the past several decades, molecular biology has discovered many new mechanisms for rearranging genetic material (e.g., jumping genes, gene deletion and duplication, and introns and exons). It would be interesting to know if any of these is significant algorithmically.

Explicit fitness evaluation is the most biologically unrealistic aspect of GAs. Several of the examples described in the previous sections (e.g., ERL, Echo, Strategic Bugs, and some of the PD work) move away from an external, static fitness measure toward more coevolutionary and endogenous evaluations. Although it is relatively easy to implement endogenous or coevolutionary fitness strategies, there is virtually no theory describing the behavior of GAs under these circumstances. In particular, a theory about how building blocks are processed (cf. [42, 55]) under these circumstances would be helpful.

Perhaps the most obvious area for extending the GA is to the study of evolution itself. Although ideas from evolution have provided inspiration for developing interesting computational techniques, there have been few attempts to use these techniques to understand better the evolutionary systems that inspired them. GAs, and the insights provided by analyzing them carefully, should help us to understand better natural evolutionary systems. This "closing of the modeling loop" is an important area of future research on evolutionary computational methods.

Acknowledgments

The authors gratefully acknowledge the Santa Fe Institute Adaptive Computation Program and the Alfred P. Sloan Foundation (grant B1992-46). Support was also provided to Forrest by the National Science Foundation (grant IRI-9157644). We thank Ron Hightower, Terry Jones, and Chris Langton for suggestions that improved this paper.

Suggested Reading

1. Belew, R. K., & Booker, L. B. (Eds.) (1991) *Proceedings of the Fourth International Conference on Genetic Algorithms.* San Mateo, CA: Morgan Kaufmann.
2. Farmer, J. D., Lapedes, A., Packard, N. H., & Wendroff, B., (Eds.) (1986). *Evolution, games, and learning.* Special issue of *Physica D, 22.*
3. Forrest, S. (Ed.) (1990). *Emergent computation.* Cambridge, MA: The MIT Press. Also published as *Physica D, 42.*
4. Forrest, S. (Ed.) (1993) *Proceedings of the Fifth International Conference on Genetic Algorithms.* San Mateo, CA: Morgan Kaufmann.
5. Forrest, S. (1993). Genetic algorithms: Principles of natural selection applied to computation. *Science, 261,* 872–878.
6. Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning.* Reading, MA: Addison-Wesley.
7. Grefenstette, J. J. (Ed.) (1985). *Proceedings of an International Conference on Genetic Algorithms and Their Applications.* Hillsdale, NJ: Lawrence Erlbaum Associates.
8. Grefenstette, J. J. (Ed.) (1987). *Proceedings of the Second International Conference on Genetic Algorithms.* Hillsdale, NJ: Lawrence Erlbaum Associates.
9. Holland, J. H. (1992). *Adaptation in natural and artificial systems.* 2nd ed. Cambridge, MA: The MIT Press. (1st ed., 1975).
10. Holland, J. H. (1992). Genetic algorithms. *Scientific American,* July, pp. 114–116.
11. Holland, J. H., Holyoak, K. J., Nisbett, R. E., & Thagard, P. (1986). *Induction: Processes of inference, learning, and discovery.* Cambridge, MA: The MIT Press.
12. Langton, C. G. (Ed.) (1989). *Artificial life.* Reading, MA: Addison-Wesley.
13. Langton, C. G. (Ed.) (1993). *Artificial life III.* Reading, MA: Addison-Wesley.
14. Langton, C. G., Taylor, C., Farmer, J. D., & Rasmussen, S. (Eds.) (1992). *Artificial life II.* Reading, MA: Addison-Wesley.
15. Männer, R., & Manderick, B. (Eds.) (1992). *Parallel problem solving from nature 2.* Amsterdam: North Holland.
16. Meyer, J.-A., Roitblatt, H. L., and Wilson, S. W. (Eds.) (1993). *From animals to animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior.* Cambridge, MA: The MIT Press
17. Meyer, J.-A., & Wilson, S. W. (Eds.) (1991). *From animals to animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior.* Cambridge, MA: The MIT Press
18. Mitchell, M. (1993). Genetic algorithms. In L. Nadel and D. L. Stein (Eds.), *1992 lectures in complex systems.* Reading, MA: Addison-Wesley.
19. Schaffer, J. D. (ed.) (1989). *Proceedings of the Third International Conference on Genetic Algorithms.* Los Altos, CA: Morgan-Kaufmann.
20. Schwefel, H.-P., & Männer, R. (Eds.) (1990). *Parallel problem solving from nature.* Berlin: Springer-Verlag (Lecture Notes in Computer Science, Vol. 496).
21. Varela, F. J., and Bourgine, P. (Eds.) (1992). *Toward a practice of autonomous systems: Proceedings of the First European Conference on Artificial Life.* Cambridge, MA: The MIT Press.

References

1. Ackley, D. H., & Littman, M. L. (1992). Interactions between learning and evolution. In C. G. Langton, C. Taylor, J. D. Farmer, & S. Rasmussen (Eds.), *Artificial life II,* (pp. 487–507). Reading, MA: Addison-Wesley.

2. Ackley, D. H., & Littman, M. L. (1993). A case for Lamarckian evolution. In C. G. Langton (Ed.), *Artificial life III.* Reading, MA: Addison-Wesley.

3. Andreoni, J., & Miller, J. H. (1991). *Auctions with adaptive artificial agents* (Working Paper 91-01-004). Santa Fe, NM: Santa Fe Institute.

4. Andreoni, J., & Miller, J. H. (in press). Auction experiments in artificial worlds. *Cuadernos.*

5. Arthur, W. Brian (1993). On designing economic agents that behave like human agents. *Evolutionary Economics, 3,* 1–22.

6. Axelrod, R. (1984). *The evolution of cooperation.* New York, NY: Basic Books.

7. Axelrod, R. (1986). An evolutionary approach to norms. *The American Political Science Review, 80.*

8. Axelrod, R. (1987). The evolution of strategies in the iterated Prisoner's Dilemma. In L. D. Davis (Ed.), *Genetic algorithms and simulated annealing.* Research Notes in Artificial Intelligence. Los Altos, CA: Morgan Kaufmann.

9. Bäck, T., Hoffmeister, F., & Schwefel, H.-P. (1991). A survey of evolution strategies. In R. K. Belew & L. B. Booker (Eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms* (pp. 2–9). San Mateo, CA: Morgan Kaufmann.

10. Baldwin, J. M. (1896). A new factor in evolution. *American Naturalist, 30,* 441–451.

11. Bedau, M. A., & Packard, N. H. (1992). Measurement of evolutionary activity, teleology, and life. In C. G. Langton, C. Taylor, J. D. Farmer, & S. Rasmussen (Eds.), *Artificial life II* (pp. 431–461). Reading, MA: Addison-Wesley.

12. Bedau, M. A., Ronneburg, F., & Zwick, M. (1992). Dynamics of diversity in an evolving population. In R. Männer & B. Manderick (Eds.), *Parallel problem solving from nature 2* (pp. 95–104). Amsterdam: North Holland.

13. Belew, R. K. (1990). Evolution, learning, and culture: Computational metaphors for adaptive algorithms. *Complex Systems, 4,* 11–49.

14. Belew, R. K. (1993). Interposing an ontogenic model between genetic algorithms and neural networks. In J. Cowan (Ed.), *Advances in neural information processing (NIPS5).* San Mateo, CA: Morgan Kaufmann.

15. Belew, R. K., McInerney, J., & Schraudolph, N. N. Evolving networks: Using the genetic algorithm with connectionist learning. In C. G. Langton, C. Taylor, J. D. Farmer, & S. Rasmussen (Eds.), *Artificial life II* (pp. 511–547). Reading, MA: Addison-Wesley.

16. Bergman, A., & Feldman, M. W. (1992). Recombination dynamics and the fitness landscape. *Physica D, 56,* 57–67.

17. Bersini, H., & Varela, F. J. (1991). The immune recruitment mechanism: A selective evolutionary strategy. In R. K. Belew & L. B. Booker (Eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms* (pp. 520–526). San Mateo, CA: Morgan Kaufmann.

18. Bledsoe, W. W. (1961). The use of biological concepts in the analytical study of systems. Paper presented at the ORSA-TIMS National Meeting, San Francisco, CA, November 1961.

19. Booker, L. (1991). Instinct as an inductive bias for learning behavioral sequences. In J.-A. Meyer & S. W. Wilson (Eds.), *From animals to animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior* (pp. 230–237). Cambridge, MA: MIT Press.

20. Booker, L. B. (1982). *Intelligent behavior as an adaptation to the task environment.* Ph.D. thesis, The University of Michigan, Ann Arbor, MI, 1982.

21. Box, G. E. P. (1957). Evolutionary operation: A method for increasing industrial productivity. *Journal of the Royal Statistical Society C, 6,* 81–101.

22. Bremermann, H. J. (1962). Optimization through evolution and recombination. In M. C. Yovits, G. T. Jacobi, & G. D. Goldstein (Eds.), *Self-organizing systems* (pp. 93–106). Washington, DC: Spartan Books.

23. Buss, L. W. (1987). *The evolution of individuality.* Princeton, NJ: Princeton University Press.

24. Celada, F., & Seiden, P. E. (1992). A computer model of cellular interactions in the immune system. *Immunology Today, 13,* 56–62.

25. Chalmers, D. J. (1990). The evolution of learning: An experiment in genetic connectionism. In D. S. Touretzky et al. (Eds.), *Proceedings of the 1990 Connecionist Models Summer School.* San Mateo, CA: Morgan Kaufmann.

26. Collins, R. J., & Jefferson, D. R. (1991). Selection in massively parallel genetic algorithms. In R. K. Belew & L. B. Booker (Eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms* (pp. 249–256). San Mateo, CA: Morgan Kaufmann.

27. Collins, R. J., & Jefferson, D. R. (1992). AntFarm: Towards simulated evolution. In C. G. Langton, C. Taylor, J. D. Farmer, & S. Rasmussen (Eds.), *Artificial life II* (pp. 579–601). Reading, MA: Addison-Wesley.

28. Collins, R. J., & Jefferson, D. R. (1992). The evolution of sexual selection and female choice. In F. J. Varela & P. Bourgine (Eds.), *Toward a practice of autonomous systems: Proceedings of the First European Conference on Artificial Life* (pp. 327–336). Cambridge, MA: The MIT Press.

29. Davidor, Y. (1992). *Genetic algorithms and robotics.* Robotics and Automated Systems. Singapore: World Scientific.

30. Davis, L. D. (Ed.) (1991). *Handbook of genetic algorithms.* New York: Van Nostrand Reinhold.

31. Dorigo, M. & Sirtori, E. (1991). Alecsys: A parallel laboratory for learning classifier systems. In R. K. Belew & L. B. Booker (Eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms* (pp. 296–302). San Mateo, CA: Morgan Kaufmann.

32. Dumeur, R. (1991). Extended classifiers for simulation of adaptive behavior. In J. A. Meyer & S. W. Wilson (Eds.), *From animals to animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior* (pp. 58–65). Cambridge, MA: The MIT Press.

33. Farmer, J. D., Packard, N. H., & Perelson, A. S. (1986). The immune system, adaptation, and machine learning. *Physica D, 22,* 187–204.

34. Fogel, D. B. (1992). *Evolving artificial intelligence.* Ph.D. thesis, University of California, San Diego, CA.

35. Fogel, D. B., & Atmar, J. W. (1990). Comparing genetic operators with Gaussian mutations in simulated evolutionary processes using linear search. *Biological Cybernetics, 63,* 111–114.

36. Fogel, L. J., Owens, A. J., & Walsh, M. J. (1966). *Artificial intelligence through simulated evolution.* New York: John Wiley.

37. Fontanari, J. F., & Meir, R. (1990). The effect of learning on the evolution of asexual populations. *Complex Systems, 4,* 401–414.

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

39. Fraser, A. S. (1957). Simulation of genetic systems by automatic digital computers: I. introduction. *Australian Journal of Biological Science, 10,* 484–491.

40. Fraser, A. S. (1957). Simulation of genetic systems by automatic digital computers: II. effects of linkage on rates of advance under selection. *Australian Journal of Biological Science, 10,* 492–499.

41. Friedman, G. J. (1959). Digital simulation of an evolutionary process. *General Systems Yearbook, 4,* 171–184.

42. Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning.* Reading, MA: Addison-Wesley.

43. Goldberg, D. E., Korb, B., & Deb, K. (1990). Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems, 3*, 493–530.

44. Grefenstette, J. J. (1991). Lamarckian learning in multi-agent environments. In R. K. Belew & L. B. Booker (Eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms* (pp. 303–310). San Mateo, CA: Morgan Kaufmann.

45. Grefenstette, J. J., & Baker, J. E. (1989). How genetic algorithms work: A critical look at implicit parallelism. In J. D. Schaffer (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms.* San Mateo, CA: Morgan Kaufmann.

46. Grefenstette, J. J., Ramsey, C. L., & Schultz, A. C. (1990). Learning sequential decision rules using simulation models and competition. *Machine Learning, 5*, 355–381.

47. Gruau, F. (1992). Genetic synthesis of Boolean neural networks with a cell rewriting developmental process. In L. D. Whitley & J. D. Schaffer (Eds.), *International Workshop on Combinations of Genetic Algorithms and Neural Networks* (pp. 55–72). Los Alamitos, CA: IEEE Computer Society Press.

48. Harp, S. A., & Samad, T. (1991). Genetic synthesis of neural network architecture. In L. D. Davis (Ed.), *Handbook of genetic algorithms* (pp. 202–221). New York: Van Nostrand Reinhold.

49. Harvey, I. (1993). The puzzle of the persistent question marks: A case study of genetic drift. In S. Forrest (Ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms* (pp. 15–22). San Mateo, CA: Morgan Kaufmann.

50. Harvey, I., Husbands, P., & Cliff, D. (1993). Issues in evolutionary robotics. In J.-A. Meyer, H. L. Roitblat, & S. W. Wilson (Eds.), *From animals to animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior* (pp. 364–373). Cambridge, MA: The MIT Press.

51. Hightower, R., Forrest, S., & Perelson, A. (in press). The evolution of secondary organization in immune system gene libraries. In *Proceedings of the Second European Conference on Artificial Life.* (Working Paper 92-11-054). Santa Fe, NM: Santa Fe Institute.

52. Hillis, W. D. (1990). Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D, 42*, 228–234.

53. Hinton, G. E., & Nowlan, S. J. (1987). How learning can guide evolution. *Complex Systems, 1*, 495–502.

54. Holland, J. H. (1986). Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning II* (pp. 593–623). San Mateo, CA: Morgan Kaufmann.

55. Holland, J. H. (1992). *Adaptation in natural and artificial systems* (2nd ed.). Cambridge, MA: The MIT Press (1st ed., 1975).

56. Holland, J. H. (1993). *Echoing emergence: Objectives, rough definitions, and speculations for Echo-class models* (Working Paper 93-04-023). Santa Fe Institute. To appear in G. Cowan, D. Pines, & D. Melzner (Eds.), *Complexity: Metaphors, models, and reality.* Reading, MA: Addison-Wesley.

57. Holland, J. H., Holyoak, K. J., Nisbett, R. E., & Thagard, P. (1986). *Induction: Processes of inference, learning, and discovery.* Cambridge, MA: The MIT Press.

58. Holland, J. H., & Miller, J. H. (1991). *Artificial adaptive agents in economic theory* (Working Paper 91-05-025). Santa Fe, NM: Santa Fe Institute.

59. Huynen, M. (1993). *Evolutionary dynamics and pattern generalization in the sequence and secondary structure of RNA.* Ph.D. thesis, Universiteit Utrecht, The Netherlands.

60. Inman, J. K. (1978). The antibody combining region: Speculations on the hypothesis of general multispecificity. In G. I. Bell, A. S. Perelson, & G. H. Pimbley, Jr. (Eds.), *Theoretical immunology* (pp. 243–278). New York: Marcel Dekker.

61. Jefferson, D., Collins, R., Cooper, C., Dyer, M., Flowers, M., Korf, R., Taylor, C., & Wang,

A. (1992). Evolution as a theme in artificial life: The Genesys/Tracker system. In C. G. Langton, C. Taylor, J. D. Farmer, & S. Rasmussen (Eds.), *Artificial life II* (pp. 549–577). Reading, MA: Addison-Wesley.

62. Jones, T., & Forrest, S. (1993). *An introduction to SFI Echo* (Working Paper 93-12-074). Santa Fe, NM: Santa Fe Institute.

63. De Jong, K. A. (1975). *An analysis of the behavior of a class of genetic adaptive systems.* Ph.D. thesis, The University of Michigan, Ann Arbor, MI.

64. De Jong, K. A. (1990). Genetic-algorithm-based learning. In Y. Kodratoff & R. Michalski (Eds.), *Machine learning* (vol. 3, pp. 611–638). San Mateo, CA: Morgan Kaufmann.

65. De Jong, K. A. (1990). Introduction to second special issue on genetic algorithms. *Machine Learning, 5,* 351–353.

66. De Jong, K. A. (1993). Editorial introduction. *Evolutionary Computation, 1,* iii–v.

67. Kitano, H. (1990). Designing neural networks using genetic algorithms with graph generation system. *Complex Systems, 4,* 461–476.

68. Koza, J. R. (1992). Genetic evolution and co-evolution of computer programs. In C. G. Langton, C. Taylor, J. D. Farmer, & S. Rasmussen (Eds.), *Artificial life II* (pp. 603–629). Reading, MA: Addison-Wesley.

69. Koza, J. R. (1993). *Genetic programming: On the programming of computers by means of natural selection.* Cambridge, MA: MIT Press.

70. Lindgren, K. (1992). Evolutionary phenomena in simple dynamics. In C. G. Langton, C. Taylor, J. D. Farmer, & S. Rasmussen (Eds.), *Artificial life II* (pp. 295–312). Reading, MA: Addison-Wesley.

71. Lindgren, K., & Nordhal, M. G. (1993). Artificial food webs. In C. G. Langton (Eds.), *Artificial life III.* Reading, MA: Addison-Wesley.

72. MacLennan, B. (1992). Synthetic ethology: An approach to the study of communication. In C. G. Langton, C. Taylor, J. D. Farmer, & S. Rasmussen (Eds.), *Artificial life II* (pp. 631–655). Reading, MA: Addison-Wesley.

73. Marks, R. E. (1992). Breeding hybrid strategies: Optimal behavior for oligopolists. *Journal of Evolutionary Economics, 2,* 17–38.

74. Menczer, F., & Parisi, D. (1992). A model for the emergence of sex in evolving networks: Adaptive advantage or random drift? In F. J. Varela & P. Bourgine (Eds.), *Toward a practice of autonomous systems: Proceedings of the First European Conference on Artificial Life.* Cambridge, MA: MIT Press/Bradford Books.

75. Meyer, T. P., & Packard, N. H. (1991). Local forecasting of high dimensional chaotic dynamics (Tech. Rep. CCSR-91-1). Center for Complex Systems Research, Beckman Institute, University of Illinois at Urbana Champaign.

76. Miller, G. F., & Todd, P. M. (1990). Exploring adaptive agency: I: Theory and methods for simulating the evolution of learning. In D. S. Touretzky et al. (Eds.), *Proceedings of the 1990 Connectionist Models Summer School.* San Mateo, CA: Morgan Kaufmann.

77. Miller, G. F., Todd, P. M., & Hegde, S. U. (1989). Designing neural networks using genetic algorithms. In J. D. Schaffer (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 379–384). San Mateo, CA: Morgan Kaufmann.

78. Miller, J. H. (1988). *Two essays on the economics of imperfect information.* Ph.D. thesis, The University of Michigan, Ann Arbor, MI.

79. Miller, J. H. (1989). The coevolution of automata in the repeated prisoner's dilemma (Tech. Rep. 89-003). Santa Fe, NM: Santa Fe Institute.

80. Mitchell, M., Crutchfield, J. P., & Hraber, P. T. (in press). Evolving cellular automata to perform computations: Mechanisms and impediments. *Physica D.*

81. Montana, D. J., & Davis, L. D. (1989). Training feedforward networks using genetic algorithms. In *Proceedings of the International Joint Conference on Artificial Intelligence.*

San Mateo, CA: Morgan Kaufmann.

82. Nolfi, S., Elman, J. L., & Parisi, D. (1990). Learning and evolution in neural networks (Tech. Rep. CRL 9019). San Diego: Center for Research in Language, University of California.

83. Packard, N. H. (1989). Intrinsic adaptation in a simple model for evolution. In C. G. Langton (Ed.), *Artificial life* (pp. 141–155). Reading, MA: Addison-Wesley.

84. Parisi, D., Nolfi, S., & Cecconi, F. (1992). Learning, behavior, and evolution. In F. J. Varela & P. Bourgine (Eds.), *Toward a practice of autonomous systems: Proceedings of the First European Conference on Artificial Life.* Cambridge, MA: The MIT Press.

85. Perelson, A. S. (1989). Immune network theory. *Immunological Review, 110,* 5–36.

86. Perelson, A. S., Weisbuch, G., & Coutinho, A. (1992). *Theoretical and experimental insights into immunology.* New York: Springer-Verlag.

87. Ray, T. S. (1991). Is it alive, or is it GA? In R. K. Belew & L. B. Booker (Eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms* (pp. 527–534). San Mateo, CA: Morgan Kaufmann.

88. Ray, T. S. (1992). An approach to the synthesis of life. In C. G. Langton, C. Taylor, J. D. Farmer, & S. Rasmussen (Eds.), *Artificial life II* (pp. 371–408). Reading, MA: Addison-Wesley.

89. Rechenberg, I. (1973). *Evolutionstrategie: optimierung technischer systeme nach prinzipien der biologischen evolution.* Stuttgart: Frommann-Holzboog.

90. Riolo, R. (1991). Lookahead planning and latent learning in a classifier system. In J. A. Meyer & S. W. Wilson (Eds.), *From animals to animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior* (pp. 316–326). Cambridge, MA: The MIT Press.

91. Riolo, R. (1991). Modeling simple human category learning with a classifier system. In R. K. Belew and L. B. Booker (Eds.), *Proceedings of the Fifth International Conference on Genetic Algorithms* (pp. 324–333). San Mateo, CA: Morgan Kaufmann.

92. Rogers, D. (1990). Weather prediction using a genetic memory (Tech. Rep. 90.6). Moffett Field, CA: Research Institute for Advanced Computer Science, NASA Ames Research Center.

93. Schaffer, J. D., & Eshelman, L. J. (1991). On crossover as an evolutionarily viable strategy. In R. K. Belew & L. B. Booker (Eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms* (pp. 61–68). San Mateo, CA: Morgan Kaufmann.

94. Schaffer, J. D., & Whitley, D., & Eshelman, L. J. (1992). Combinations of genetic algorithms and neural networks: A survey of the state of the art. In L. D. Whitley & J. D. Schaffer (Eds.), *International Workshop on Combinations of Genetic Algorithms and Neural Networks* (pp. 1–37). Los Alamitos, CA: IEEE Computer Society Press.

95. Schulze-Kremer, S. (1992). Genetic algorithms for protein tertiary structure prediction. In R. Männer & B. Manderick (Eds.), *Parallel problem solving from nature 2* (pp. 391–400). Amsterdam: North Holland.

96. Schwefel, H.-P. (1975). *Evolutionsstrategie und numerische Optimierung.* Ph.D. thesis, Technische Universität Berlin, Berlin, 1975.

97. Schwefel, H.-P. (1977). *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie* (vol. 26 of *Interdisciplinary Systems Research*). Basel: Birkhäuser.

98. Maynard Smith, J. (1987). When learning guides evolution. *Nature, 329,* 761–762.

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

100. Syswerda, G. (1989). Uniform crossover in genetic algorithms. In J. D. Schaffer (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 2–9). San Mateo, CA: Morgan Kaufmann.

101. Taylor, C. E., Jefferson, D. R., Turner, S. R., & Goldman, S. R. (1989). RAM: Artificial life

for the exploration of complex biological systems. In C. G. Langton (Ed.), *Artificial life* (pp. 275–295). Reading, MA: Addison-Wesley.

102. Todd, P. M., & Miller, G. F. (1990). Exploring adaptive agency III: Simulating the evolution of habituation and sensitization. In H.-P. Schwefel & R. Männer (Eds.), *Parallel problem solving from nature* (Lecture Notes in Computer Science). Berlin: Springer-Verlag.

103. Todd, P. M., & Miller, G. F. (1991). Exploring adaptive agency II: Simulating the evolution of associative learning. In J.-A. Meyer & S. W. Wilson (Eds.), *From animals to animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior* (pp. 306–315). Cambridge, MA: The MIT Press.

104. Werner, G. M., & Dyer, M. G. (1992). Evolution of communication in artificial organisms. In C. G. Langton, C. Taylor, J. D. Farmer, & S. Rasmussen (Eds.), *Artificial life II* (pp. 659–687). Reading, MA: Addison-Wesley.

105. Whitley, L. D., Dominic, S., & Das, R. (1991). Genetic reinforcement learning with multilayer neural networks. In R. K. Belew & L. B. Booker (Eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms* (pp. 562–569). San Mateo, CA: Morgan Kaufmann.

106. Wilson, S. W. (1985). Knowledge growth in an artificial animal. In J. Grefenstette (Ed.), *Proceedings of the First International Conference on Genetic Algorithms and Their Applciations.* Hillsdale, NJ: Lawrence Erlbaum Associates.

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

108. Wilson, S. W. (1989). The genetic algorithm and simulated evolution. In C. G. Langton (Ed.), *Artificial life* (pp. 157–165). Reading, MA: Addison-Wesley.