An Introduction to Simulated Evolutionary Optimization

David B. Fogel, *Member, IEEE*

***Abstract*—Natural evolution is a population-based optimization process. Simulating this process on a computer results in stochastic optimization techniques that can often outperform classical methods of optimization when applied to difficult real-world problems. There are currently three main avenues of research in simulated evolution: genetic algorithms, evolution strategies, and evolutionary programming. Each method emphasizes a different facet of natural evolution. Genetic algorithms stress chromosomal operators. Evolution strategies emphasize behavioral changes at the level of the individual. Evolutionary programming stresses behavioral change at the level of the species. The development of each of these procedures over the past 35 years is described. Some recent efforts in these areas are reviewed.**

I. Introduction

THE fundamental approach to optimization is to formulate a single standard of measurement—a cost function—that summarizes the performance or value of a decision and iteratively improve this performance by selecting from among the available alternatives. Most classical methods of optimization generate a deterministic sequence of trial solutions based on the gradient or higher-order statistics of the cost function [1, chaps. 8–10]. Under regularity conditions on this function, these techniques can be shown to generate sequences that asymptotically converge to locally optimal solutions, and in certain cases they converge exponentially fast [2, pp. 12–15]. Variations on these procedures are often applied to training neural networks (backpropagation) [3], [4], or estimating parameters in system identification and adaptive control applications (recursive prediction error methods, Newton-Gauss) [2, pp. 22–23], [5]. But the methods often fail to perform adequately when random perturbations are imposed on the cost function. Further, locally optimal solutions often prove insufficient for real-world engineering problems.

Darwinian evolution is intrinsically a robust search and optimization mechanism. Evolved biota demonstrate optimized complex behavior at every level: the cell, the organ, the individual, and the population. The problems that biological species have solved are typified by chaos, chance, temporality, and nonlinear interactivity. These are also characteristics of problems that have proved to be especially intractable to classic methods of optimization. The evolutionary process can be applied to problems where heuristic solutions are not available or generally lead to unsatisfactory results.

The most widely accepted collection of evolutionary theories is the neo-Darwinian paradigm. These arguments assert that the history of life can be fully accounted for by physical processes operating on and within populations and species [6, p. 39]. These processes are reproduction, mutation, competition, and selection. Reproduction is an obvious property of extant species. Further, species have such great reproductive potential that their population size would increase at an exponential rate if all individuals of the species were to reproduce successfully [7], [8, p. 479]. Reproduction is accomplished through the transfer of an individual's genetic program (either asexually or sexually) to progeny. Mutation, in a positively entropic system, is guaranteed, in that replication errors during information transfer will necessarily occur. Competition is a consequence of expanding populations in a finite resource space. Selection is the inevitable result of competitive replication as species fill the available space. Evolution becomes the inescapable result of interacting basic physical statistical processes ([9], [10, p. 25], [11] and others).

Individuals and species can be viewed as a duality of their genetic program, the genotype, and their expressed behavioral traits, the phenotype. The genotype provides a mechanism for the storage of experiential evidence, of historically acquired information. Unfortunately, the results of genetic variations are generally unpredictable due to the universal effects of pleiotropy and polygeny (Fig. 1) [8], [12], [13], [14, p. 224], [15]–[19], [20, p. 296]. *Pleiotropy* is the effect that a single gene may simultaneously affect several phenotypic traits. *Polygeny* is the effect that a single phenotypic characteristic may be determined by the simultaneous interaction of many genes. There are no one-gene, one-trait relationships in natural evolved systems. The phenotype varies as a complex, nonlinear function of the interaction between underlying genetic structures and current environmental conditions. Very different genetic structures may code for equivalent behaviors, just as diverse computer programs can generate similar functions.

Selection directly acts only on the expressed behaviors of individuals and species [19, pp. 477–478]. Wright [21] offered the concept of adaptive topography to describe the fitness of individuals and species (minimally, isolated reproductive populations termed *demes*). A population of genotypes maps to respective phenotypes (*sensu* Lewontin [22]), which are in turn mapped onto the adaptive topography (Fig. 2). Each peak corresponds to an optimized collection of phenotypes, and thus one or more sets of optimized genotypes. Evolution probabilistically proceeds up the slopes of the topography toward peaks as selection culls inappropriate phenotypic variants.

Others [11], [23, pp. 400–401] have suggested that it is more appropriate to view the adaptive landscape from an inverted

Manuscript received April 15, 1993; revised August 2, 1993.
The author is with Natural Selection, Inc., La Jolla, CA 92037.
IEEE Log Number 9213549.

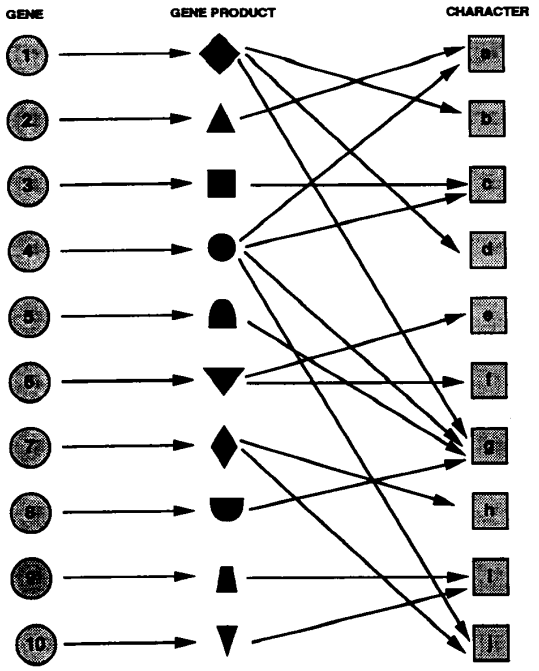


Fig. 1. *Pleiotropy* is the effect that a single gene may simultaneously affect several phenotypic traits. *Polygeny* is the effect that a single phenotypic characteristic may be determined by the simultaneous interaction of many genes. These one-to-many and many-to-one mappings are pervasive in natural systems. As a result, even small changes to a single gene may induce a raft of behavioral changes in the individual (after [18]).

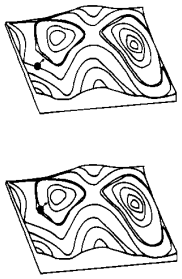


Fig. 2. Wright's adaptive topology, inverted. An adaptive topography, or adaptive landscape, is defined to represent the fitness of all possible phenotypes. Wright [21] proposed that as selection culls the least appropriate existing behaviors relative to others in the population, the population advances to areas of higher fitness on the landscape. Atmar [11] and others have suggested viewing the topography from an inverted perspective. Populations then advance to areas of lower behavioral error.

position. The peaks become troughs, "minimized prediction error entropy wells" [11]. Such a viewpoint is intuitively appealing. Searching for peaks depicts evolution as a slowly advancing, tedious, uncertain process. Moreover, there appears to be a certain fragility to an evolving phyletic line; an optimized population might be expected to quickly fall off the peak under slight perturbations. The inverted topography leaves an altogether different impression. Populations advance rapidly, falling down the walls of the error troughs until their cohesive set of interrelated behaviors are optimized, at which point stagnation occurs. If the topography is generally static, rapid descents will be followed by long periods of stasis. If, however, the topography is in continual flux, stagnation may never set in.

Viewed in this manner, evolution is an obvious optimizing problem-solving process. Selection drives phenotypes as close to the optimum as possible, given initial conditions and environmental constraints. But the environment is continually changing. Species lag behind, constantly evolving toward a new optimum. No organism should be viewed as being perfectly adapted to its environment. The suboptimality of behavior is to be expected in any dynamic environment that mandates trade-offs between behavioral requirements. But selection never ceases to operate, regardless of the population's position on the topography.

Mayr [19, p. 532] has summarized some of the more salient characteristics of the neo-Darwinian paradigm. These include:

1) The individual is the primary target of selection.
2) Genetic variation is largely a chance phenomenon. Stochastic processes play a significant role in evolution.
3) Genotypic variation is largely a product of recombination and "only ultimately of mutation."
4) "Gradual" evolution may incorporate phenotypic discontinuities.
5) Not all phenotypic changes are necessarily consequences of *ad hoc* natural selection.
6) Evolution is a change in adaptation and diversity, not merely a change in gene frequencies.
7) Selection is probabilistic, not deterministic.

Simulations of evolution should rely on these foundations.

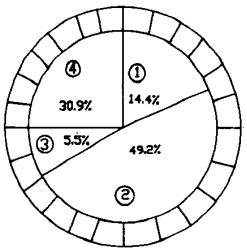
II. Genetic Algorithms

Fraser [24]–[28], Bremermann *et al.* [29]–[36], Reed *et al.* [37], and Holland [38], [39] proposed similar algorithms that simulate genetic systems. These procedures are now described by the term *genetic algorithms* and are typically implemented as follows:

1) The problem to be addressed is defined and captured in an objective function that indicates the fitness of any potential solution.
2) A population of candidate solutions is initialized subject to certain constraints. Typically, each trial solution is coded as a vector $\boldsymbol{x}$, termed a *chromosome,* with elements being described as *genes* and varying values at specific positions called *alleles.* Holland [39, pp. 70–72] suggested that all solutions should be represented by binary strings. For example, if it were desired to find the scalar value x that maximizes:

$$F(x) = -x^2,$$

then a finite range of values for x would be selected and the minimum possible value in the range would be represented by the string $\{0 \ldots 0\}$, with the maximum value being represented by the string $\{1 \ldots 1\}$. The de-



No.	String	Fitness	% of Total
1	01101	169	14.4
2	11000	576	49.2
3	01000	64	5.5
4	10011	361	30.9
Total		1170	100.0

Fig. 3. Roulette wheel selection in genetic algorithms. Selection in genetic algorithms is often accomplished via differential reproduction according to fitness. In the typical approach, each chromosome is given a probability of being copied into the next generation that is proportional to its fitness relative to all other chromosomes in the population. Successive trials are conducted in which a chromosome is selected, until all available positions are filled. Those chromosomes with above-average fitness will tend to generate more copies than those with below-average fitness. The figure is adapted from [143].

sired degree of precision would indicate the appropriate length of the binary coding.

3) Each chromosome, $\boldsymbol{x}_i$, $i = 1, \ldots, P$, in the population is decoded into a form appropriate for evaluation and is then assigned a fitness score, $\mu(\boldsymbol{x}_i)$ according to the objective.
4) Each chromosome is assigned a probability of reproduction, p_i, $i = 1, \ldots, P$, so that its likelihood of being selected is proportional to its fitness relative to the other chromosomes in the population. If the fitness of each chromosome is a strictly positive number to be maximized, this is often accomplished using *roulette wheel selection* (see Fig. 3).
5) According to the assigned probabilities of reproduction, p_i, $i = 1, \ldots, P$, a new population of chromosomes is generated by probabilistically selecting strings from the current population. The selected chromosomes generate "offspring" via the use of specific genetic operators, such as crossover and bit mutation. Crossover is applied to two chromosomes *(parents)* and creates two new chromosomes *(offspring)* by selecting a random position along the coding and splicing the section that appears before the selected position in the first string with the section that appears after the selected position in the second string, and vice versa (see Fig. 4). Other, more sophisticated, crossover operators have been introduced and will be discussed later. Bit mutation simply offers the chance to flip each bit in the coding of a new solution. Typical values for the probabilities of crossover and bit mutation range from 0.6 to 0.95 and 0.001 to 0.01, respectively [40], [41].
6) The process is halted if a suitable solution has been found, or if the available computing time has expired; otherwise the process proceeds to step (3) where the new chromosomes are scored and the cycle is repeated.

For example, suppose the task is to find a vector of 100 bits {0,1} such that the sum of all of the bits in the vector is maximized. The objective function could be written as:

$$\mu(\boldsymbol{x}) = \sum_{i=1}^{100} x_i,$$

where $\boldsymbol{x}$ is a vector of 100 symbols from {0,1}. Any such vector $\boldsymbol{x}$ could be scored with respect to $\mu(\boldsymbol{x})$ and would receive a fitness rating ranging from zero to 100. Let an initial population of 100 parents be selected completely at random and subjected to roulette wheel selection in light of $\mu(\boldsymbol{x})$, with the probabilities of crossover and bit mutation being 0.8 and 0.01, respectively. Fig. 5 shows the rate of improvement of the best vector in the population, and the average of all parents, at each generation (one complete iteration of steps 3–6) under such conditions. The process rapidly converges on vectors of all 1's.

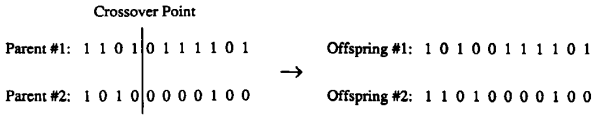


Fig. 4. The one-point crossover operator. A typical method of recombination in genetic algorithms is to select two parents and randomly choose a splicing point along the chromosomes. The segments from the two parents are exchanged and two new offspring are created.

There are a number of issues that must be addressed when using a genetic algorithm. For example, the necessity for binary codings has received considerable criticism [42]–[44]. To understand the motivation for using bit strings, the notion of a schema must be introduced. Consider a string of symbols from an alphabet $\boldsymbol{A}$. Suppose that some of the components of the string are held fixed while others are free to vary. Define a wild card symbol, #, that matches any symbol from $\boldsymbol{A}$. A string with fixed and variable symbols defines a schema. Consider the string {01##}, defined over the union of {#} and the alphabet $\boldsymbol{A}$ = {0,1}. This set includes {0100}, {0101}, {0110} and {0111}. Holland [39, pp. 66–74] recognized that every string that is evaluated actually offers partial information about the expected fitness of all possible schemata in which that string resides. That is, if the string {0000} is evaluated to have some fitness, then partial information is also received about the worth of sampling from variations in {0###}, {#0##}, {#00#}, {#0#0}, and so forth. This characteristic is termed *implicit parallelism,* as it is through a single sample that information is gained with respect to many schemata. Holland [39, p. 71] speculated that it would be beneficial to maximize the number of schemata being sampled, thus providing maximum implicit parallelism, and proved that this is achieved for

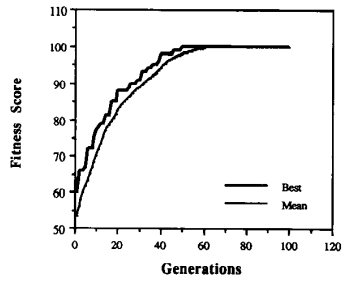


Fig. 5. The rate of optimization in a simple binary coding problem using a standard genetic algorithm. The curves indicate the fitness of the best chromosomes in the population and the mean fitness of all parents at each generation. The optimum fitness is 100 units.

$|A| = 2$. Binary strings were therefore suggested as a universal representation.

The use of binary strings is not universally accepted in genetic algorithm literature, however. Michalewicz [44, p. 82] indicates that for real-valued numerical optimization problems, floating-point representations outperform binary representations because they are more consistent, more precise, and lead to faster execution. But Michalewicz [44, p. 75] also claims that genetic algorithms perform poorly when the state space of possible solutions is extremely large, as would be required for high-precision numerical optimization of many variables that could take on real-values in a large range. This claim is perhaps too broad. The size of the state space alone does not determine the efficiency of the genetic algorithm, regardless of the choice of representation. Very large state spaces can sometimes be searched quite efficiently, and relatively small state spaces sometimes provide significant difficulties. But it is fair to say that maximizing implicit parallelism will not always provide for optimum performance. Many researchers in genetic algorithms have foregone the bit strings suggested by Holland [39, pp. 70–72] and have achieved reasonable results to difficult problems [44]–[47].

Selection in proportion to fitness can be problematic. There are two practical considerations: 1) roulette wheel selection depends upon positive values, and 2) simply adding a large constant value to the objective function can eliminate selection, with the algorithm then proceeding as a purely random walk. There are several heuristics that have been devised to compensate for these issues. For example, the fitness of all parents can be scaled relative to the lowest fitness in the population, or proportional selection can be based on ranking by fitness. Selection based on ranking also eliminates problems with functions that have large offsets.

One mathematical problem with selecting parents to reproduce in proportion to their relative fitness is that this procedure cannot ensure asymptotic convergence to a global optimum [48]. The best chromosome in the population may be lost at any generation, and there is no assurance that any gains made up to a given generation will be retained in future generations. This can be overcome by employing a heuristic termed *elitist selection* [49], which simply always retains the best chromosome in the population. This procedure guarantees asymptotic convergence [48], [50], [51], but the specific rates of convergence vary by problem and are generally unknown.

The crossover operator has been termed the distinguishing feature of genetic algorithms [52, pp. 17–18]. Holland [39, pp. 110–111] indicates that crossover provides the main search operator while bit mutation simply serves as a background operator to ensure that all possible alleles can enter the population. The probabilities commonly assigned to crossover and bit mutation reflect this philosophical view. But the choice of crossover operator is not straightforward.

Holland [39, p. 160], and others [53], [54], propose that genetic algorithms work by identifying good "building blocks" and eventually combining these to get larger building blocks. This idea has become known as the *building block hypothesis*. The hypothesis suggests that a one-point crossover operator would perform better than an operator that, say, took one bit from either parent with equal probability *(uniform crossover)*, because it could maintain sequences (blocks) of "good code" that are associated with above-average performance and not disrupt their linkage. But this has not been clearly demonstrated in the literature. Syswerda [55] conducted function optimization experiments with uniform crossover, two-point crossover and one-point crossover. Uniform crossover provided generally better solutions with less computational effort. Moreover, it has been noted that sections of code that reside at opposite ends of a chromosome are more likely to be disrupted under one-point crossover than are sections that are near the middle of the chromosome. Holland [39, pp. 106–109] proposed an inversion operator that would reverse the index position for a section of the chromosome, so that linkages could be constructed between arbitrary genes. But inversion has not been found to be useful in practice [52, p. 21]. The relevance of the building block hypothesis is presently unclear, but its value is likely to vary significantly by problem.

Premature convergence is another important concern in genetic algorithms. This occurs when the population of chromosomes reaches a configuration such that crossover no longer produces offspring that can outperform their parents, as must be the case in a homogeneous population. Under such circumstances, all standard forms of crossover simply regenerate the current parents. Any further optimization relies solely on bit mutation and can be quite slow. Premature convergence is often observed in genetic algorithm research ([40], [52, pp. 25, 26], [56], [57], and others) because of the exponential reproduction of the best observed chromosomes coupled with the strong emphasis on crossover. Davis [52, pp. 26, 27]

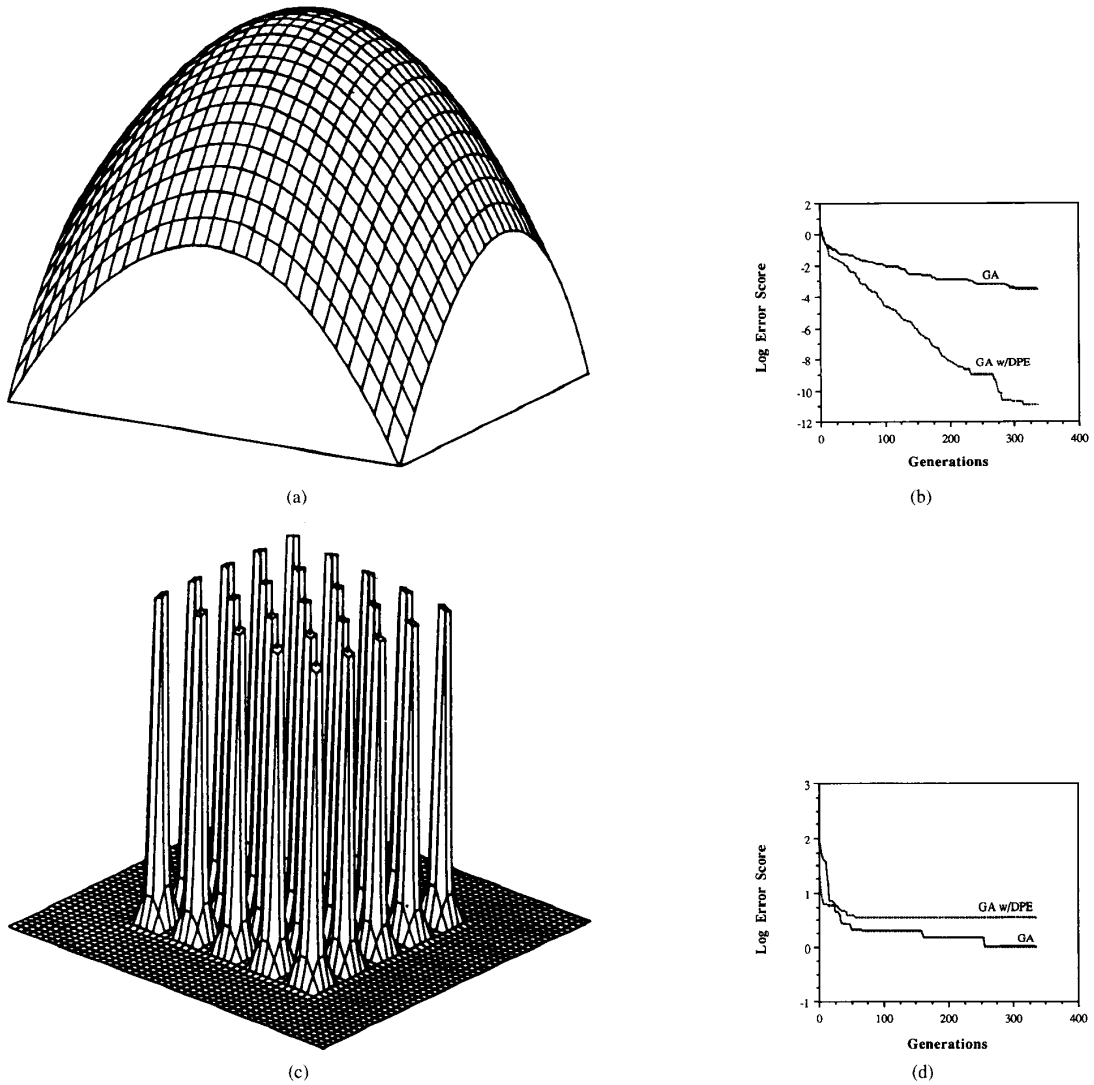


Fig. 6. Comparing dynamic parameter encoding to more standard genetic algorithm coding techniques. (a) A two-dimensional, inverted illustration of a quadratic bowl. (b) Optimization on a three-dimensional quadratic bowl. (c) An inverted illustration of the Shekel's foxholes problem. (d) Optimization on the Shekel's foxholes problem. Dynamic parameter encoding offers the possibility of increasing the precision of a solution on-line, but may also encounter problems with premature convergence.

recommends that when the population converges on a chromosome that would require the simultaneous mutation of many bits in order to improve it, the run is practically completed and it should either be restarted using a different random seed, or hill-climbing heuristics should be employed to search for improvements.

One recent proposal for alleviating the problems associated with premature convergence was offered in [41]. The method, termed *dynamic parameter encoding* (DPE), dynamically resizes the available range of each parameter. Broadly, when a heuristic suggests that the population has converged, the minimum and maximum values for the range are resized to a smaller window and the process is iterated. In this manner, DPE can zoom in on solutions that are closer to the global optimum than provided by the initial precision. Schraudolph [58] has kindly provided results from experiments with DPE presented in [41]. As indicated in Fig. 6, DPE clearly outperforms the standard genetic algorithm when searching a quadratic bowl, but actually performs worse on a multimodal function (Shekel's foxholes). The effectiveness of DPE is an open, promising area of research. DPE only zooms in, so the initial range of parameters must be set to include the global optimum or it will not be found. But it would be relatively straightforward to include a mechanism in DPE to expand the search window, as well as reduce it.

Although many open questions remain, genetic algorithms have been used to successfully address diverse practical optimization problems [59]. While some researchers do not view

genetic algorithms as function optimization procedures *per se* (e.g., [60]), they are commonly used for precisely that purpose. Current research efforts include: 1) developing a stronger mathematical foundation for the genetic algorithm as an optimization technique [41], [48], [61], [62], including analysis of classes of problems that are difficult for genetic algorithms [63]–[66] as well as the sensitivity to performance of the general technique to various operator and parameter settings [42], [44], [67]–[70]; 2) comparing genetic algorithms to other optimization methods and examining the manner in which they can be enhanced by incorporating other procedures such as simulated annealing [71]–[73]; 3) using genetic algorithms for computer programming and engineering problems [74]–[79]; 4) applying genetic algorithms to machine learning rule-based classifier systems [80]–[84]; 5) using genetic algorithms as a basis for artificial life simulations [85], [86, pp. 186–195]; and 6) implementing genetic algorithms on parallel machines [87]–[89]. The most recent investigations can be found in [90].

III. Evolution Strategies and Evolutionary Programming

An alternative approach to simulating evolution was independently adopted by Schwefel [91] and Rechenberg [92] collaborating in Germany, and L. Fogel [93], [94] in the United States, and later pursued by [95]–[99], among others. These models, commonly described by the terms *evolution strategies* or *evolutionary programming*, or more broadly as *evolutionary algorithms* [87], [100] (although many authors use this term to describe the entire field of simulated evolution), emphasize the behavioral link between parents and offspring, or between reproductive populations, rather than the genetic link. When applied to real-valued function optimization, the most simple method is implemented as follows:

1) The problem is defined as finding the real-valued n-dimensional vector $\boldsymbol{x}$ that is associated with the extremum of a functional $F(\boldsymbol{x}) : \boldsymbol{R}^n \rightarrow \boldsymbol{R}$. Without loss of generality, let the procedure be implemented as a minimization process.
2) An initial population of parent vectors, $\boldsymbol{x}_i$, $i = 1, \ldots, P$, is selected at random from a feasible range in each dimension. The distribution of initial trials is typically uniform.
3) An offspring vector, $\boldsymbol{x}_i'$, $i = 1, \ldots, P$, is created from each parent $\boldsymbol{x}_i$ by adding a Gaussian random variable with zero mean and preselected standard deviation to each component of $\boldsymbol{x}_i$.
4) Selection then determines which of these vectors to maintain by comparing the errors $F(\boldsymbol{x}_i)$ and $F(\boldsymbol{x}_i')$, $i = 1, \ldots, P$. The P vectors that possess the least error become the new parents for the next generation.
5) The process of generating new trials and selecting those with least error continues until a sufficient solution is reached or the available computation is exhausted.

In this model, each component of a trial solution is viewed as a behavioral trait, not as a gene. A genetic source for these phenotypic traits is presumed, but the nature of the linkage is not detailed. It is assumed that whatever genetic transformations occur, the resulting change in each behavioral trait will follow a Gaussian distribution with zero mean difference and some standard deviation. Specific genetic alterations can affect many phenotypic characteristics due to pleiotropy and polygeny (Fig. 1). It is therefore appropriate to simultaneously vary all of the components of a parent in the creation of a new offspring.

The original efforts in evolution strategies [91], [92] examined the preceding algorithm but focused on a single parent-single offspring search. This was termed a $(1+1)-ES$ in that a single offspring is created from a single parent and both are placed in competition for survival, with selection eliminating the poorer solution. There were two main drawbacks to this approach when viewed as a practical optimization algorithm: 1) the constant standard deviation (step size) in each dimension made the procedure slow to converge on optimal solutions, and 2) the brittle nature of a point-to-point search made the procedure susceptible to stagnation at local minima (although the procedure can be shown to asymptotically converge to the global optimum vector $\boldsymbol{x}$) [101].

Rechenberg [92] defined the expected convergence rate of the algorithm as the ratio of the average distance covered toward the optimum and the number of trials required to achieve this improvement. For a quadratic function

$$F(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2, \tag{1}$$

where $\boldsymbol{x}$ is an n-dimensional vector of reals, and x_i denotes the ith component of $\boldsymbol{x}$, Rechenberg [92] demonstrated that the optimum expected convergence rate is given when $\sigma \approx 1.224r/n$, where σ is the standard deviation of the zero mean Gaussian perturbation, r denotes the current Euclidean distance from the optimumn and there are n dimensions. Thus, for this simple function the optimum convergence rate is obtained when the average step size is proportional to the square root of the error function and inversely proportional to the number of variables. Additional analyses have been conducted on other functions and the results have yielded similar forms for setting the standard deviation [102].

The use of multiple parents and offspring in evolution strategies was developed by Schwefel [103], [104]. Two approaches are currently explored, denoted by $(\mu + \lambda) - ES$ and $(\mu, \lambda) - ES$. In the former, μ parents are used to create λ offspring and all solutions compete for survival, with the best being selected as parents of the next generation. In the latter, only the λ offspring compete for survival, and the parents are completely replaced each generation. That is, the lifespan of every solution is limited to a single generation. Increasing the population size increases the rate of optimization over a fixed number of generations.

To provide a very simple example, suppose it is desired to find the minimum of the function in (1) for $n = 3$. Let the original population consist of 30 parents, with each component initialized in accordance with a uniform distribution over $[-5.12, 5.12]$ (after [40]). Let one offspring be created from each parent by adding a Gaussian random variable with mean zero and variance equal to the error score of the parent divided

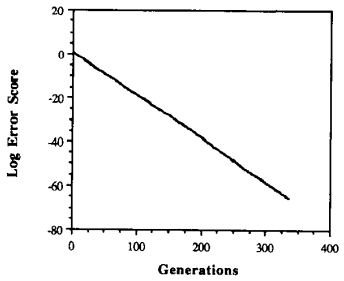


Fig. 7. The rate of optimization using a primitive version of evolution strategies on the three-dimensional quadratic bowl. Thirty parents are maintained at each generation. Offspring are created by adding a Gaussian random variable to each component.

by the square of the number of dimensions ($3^2 = 9$) to each component. Let selection simply retain the best 30 vectors in the population of parents and offspring. Fig. 7 indicates the rate of optimization of the best vector in the population as a function of the number of generations. The process rapidly converges close to the unique global optimum.

Rather than using a heuristic schedule for reducing the step size over time, Schwefel [104] developed the idea of making the distribution of new trials from each parent an additional adaptive parameter (Rechenberg, personal communication, indicates that he introduced the idea in 1967). In this procedure, each solution vector comprises not only the trial vector $\boldsymbol{x}$ of n dimensions, but a perturbation vector $\boldsymbol{\sigma}$ which provides instructions on how to mutate $\boldsymbol{x}$ and is itself subject to mutation. For example, if $\boldsymbol{x}$ is the current position vector and $\boldsymbol{\sigma}$ is a vector of variances corresponding to each dimension of $\boldsymbol{x}$, then a new solution vector ($\boldsymbol{x}'$,$\boldsymbol{\sigma}'$) could be created as:

$$\begin{aligned} \boldsymbol{\sigma}_i' &= \boldsymbol{\sigma}_i \exp(\tau' \cdot N(0,1) + \tau \cdot N_i(0,1)) \\ \boldsymbol{x}_i' &= \boldsymbol{x}_i + N(0, \boldsymbol{\sigma}_i') \end{aligned}$$

where $i = 1, \ldots, n$, and $N(0,1)$ represents a single standard Gaussian random variable, $N_i(0,1)$ represents the ith independent identically distributed standard Gaussian, and τ and τ' are operator set parameters which define global and individual step-sizes [102]. In this manner, the evolution strategy can self-adapt to the width of the error surface and more appropriately distribute trials. This method was extended again [104] to incorporate correlated mutations so that the distribution of new trials could adapt to contours on the error surface (Fig. 8).

Finally, additional extensions were made to evolution strategies to include methods for recombining individual solutions in the creation of new offspring. There are many proposed procedures. These include selecting individual components from either of two parents at random, averaging individual components from two parents with a given weighting, and so forth [102].

The original evolutionary programming approach was similar to that of Schwefel and Rechenberg but involved a more complex problem, that of creating artificial intelligence. Fogel [94] proposed that intelligent behavior requires the composite ability to predict one's environment coupled with a translation of the predictions into a suitable response in light of the given goal. To provide maximum generality, in a series

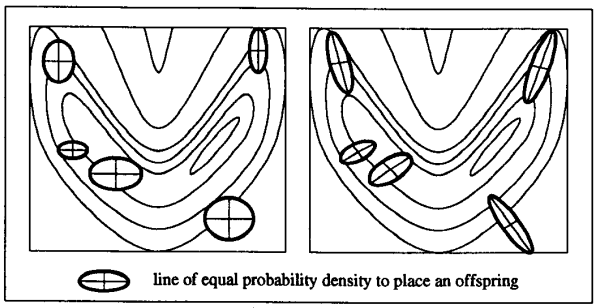


Fig. 8. Under independent Gaussian perturbations to each component of every parent, new trials are are distributed such that the contours of equal probability are aligned with the coordinate axes (left picture). This will not be optimal in general because the contours of the response are rarely similarly aligned. Schwefel [104] suggests a mechanism for incorporating self-adaptive covariance terms. Under this procedure, new trials can be distributed in any orientation (right picture). The evolutionary process adapts to the contours of the response surface, distributing trials so as to maximaize the probability of discovering improved solutions.

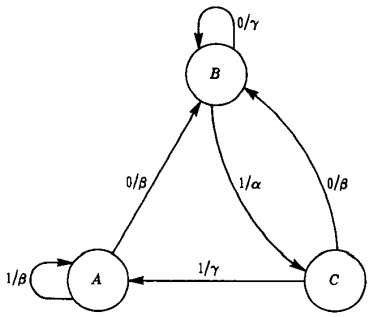


Fig. 9. A finite state machine (FSM) consists of a finite number of states. For each state, for every possible input symbol, there is an associated output symbol and next-state transition. In the figure, input symbols are shown to the left of the virgule, output symbols are shown to the right. The input alphabet is {0, 1} and the output alphabet is {α, β, γ}. The machine is presumed to start in state A. The figure is taken from [144].

of experiments, a simulated environment was described as sequence of symbols taken from a finite alphabet. The problem was then defined to evolve an algorithm that would operate on the sequence of symbols thus far observed in such a manner as to produce an output symbol that is likely to maximize the benefit to the algorithm in light of the next symbol to appear in the environment and a well-defined payoff function. Finite state machines (FSM's) [105] provided a useful representation for the required behavior (Fig. 9).

Evolutionary programming operated on FSM's as follows:

1) Initially, a population of parent FSM's is randomly constructed.
2) The parents are exposed to the environment; that is, the sequence of symbols that have been observed up to the current time. For each parent machine, as each input symbol is offered to the machine, each output symbol is compared to the next input symbol. The worth of this prediction is then measured with respect to the given payoff function (e.g., all–none, absolute error, squared error, or any other expression of the meaning of the symbols). After the last prediction is made, a function

		Player B: C	Player B: D
Player A	C	(3,3)	(0,5)
	D	(5,0)	(1,1)

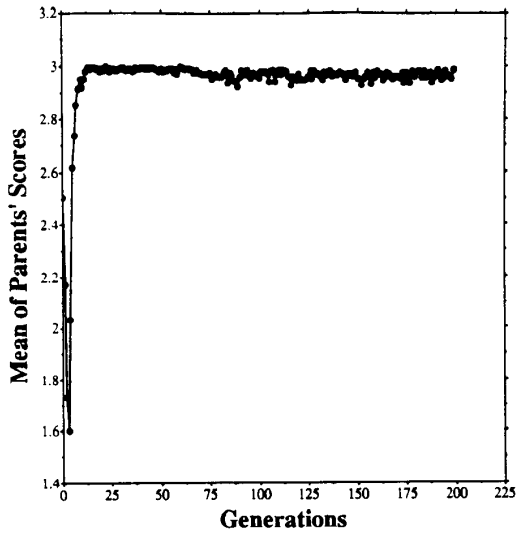
Fig. 10. A payoff matrix for the prisoner's dilemma. Each of two players must either cooperate (C) or defect (D). The entries in the matrix, (a,b), indicate the gain to players A and B, respectively. This payoff matrix was used in simulations in [106]–[108].

of the payoff for each symbol (e.g., average payoff per symbol) indicates the fitness of the machine.

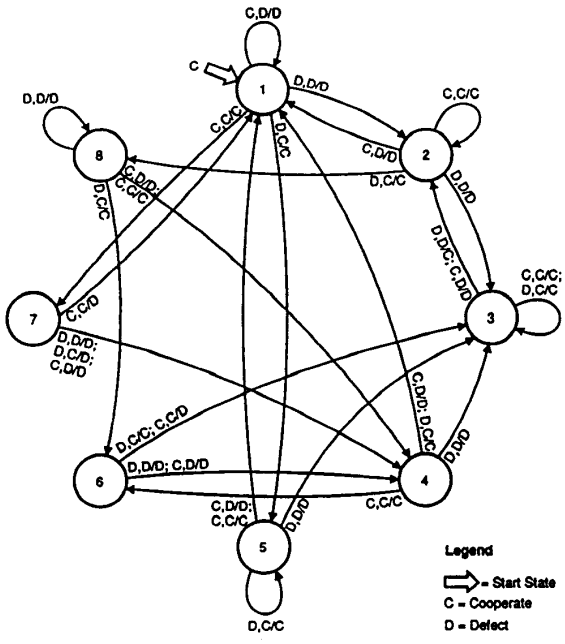
3) Offspring machines are created by randomly mutating each parent machine. There are five possible modes of random mutation that naturally result from the description of the machine: change an output symbol, change a state transition, add a state, delete a state, or change the initial state. The deletion of a state and the changing of the start state are only allowed when the parent machine has more than one state. Mutations are chosen with respect to a probability distribution, which is typically uniform. The number of mutations per offspring is also chosen with respect to a probability distribution (e.g., Poisson) or may be fixed *a priori*.
4) The offspring are evaluated over the existing environment in the same manner as their parents.
5) Those machines that provide the greatest payoff are retained to become parents of the next generation. Typically, the parent population remains the same size, simply for convenience.
6) Steps 3)–5) are iterated until it is required to make an actual prediction of the next symbol (not yet experienced) from the environment. The best machine is selected to generate this prediction, the new symbol is added to the experienced environment, and the process reverts to step 2).

The prediction problem is a sequence of static optimization problems in which the adaptive topography (fitness function) is time-varying. The process can be easily extended to abstract situations in which the payoffs for individual behaviors depend not only on an extrinsic payoff function, but also on the behavior of other individuals in the population. For example, Fogel [106], [107], following previous foundational research by Axelrod using genetic algorithms [108], evolved a population of FSM's in light of the iterated prisoner's dilemma (Fig. 10). Starting with completely random FSM's of one to five states, but ultimately possessing a maximum of eight states, the simulated evolution quickly converged on mutually cooperative behavior (Fig. 11). The evolving FSM's essentially learned to predict the behavior (a sequence of symbols) of other FSM's in the evolving population.

Evolutionary programming has recently been applied to real-valued continuous optimization problems and is virtually



(a)



(b)

Fig. 11. (a) The mean of all parents' scores as a function of the number generations when using evolutionary programming to simulate an iterated prisoner's dilemma incorporating 50 parents coded as finite state machines (FSM's). The input alphabet consists of the previous moves for the current player and the opponent {(C,C), (C,D), (D,C), (D,D)}; the output alphabet consists of the next move {C,D}. Each FSM plays against every other FSM in the population over a long series of moves. The results indicate a propensity to evolve cooperative behavior even though it would appear more beneficial for an individual to defect on any given play. (b) A typical FSM evolved after 200 generations when using 100 parents. The cooperative nature of the machine can be observed by noting that (C,C) typically elicits further cooperation, and in states 2 and 3, such cooperation will be absorbing. Further, (D,D) typically elicits further defection, indicating that the machine will not be taken advantage of during an encounter with a purely selfish machine. These results appear in [107].

equivalent in many cases to the procedures used in evolution strategies. The extension to using self-adapting independent variances was offered in [109] with procedures for optimizing the covariance matrix used in generating new trials offered in [110]. These methods differ from those offered in [104] in that Gaussian perturbations are appl ied to the self-adaptive parameters instead of lognormal perturbations. Initial comparisons [111], [112] indicate that the procedures in [104] appear to be more robust than those in [110]. One possible explanation for this would be that it is easier for variances of individual terms to transition between small and large values under the method of [104]. Theoretical and empirical comparison between these mechanisms is an open area of research.

As currently implemented, there are two essential differences between evolution strategies and evolutionary programming.

1) Evolution strategies rely on strict deterministic selection. Evolutionary programming typically emphasizes the probabilistic nature of selection by conducting a stochastic tournament for survival at each generation. The probability that a particular trial solution will be maintained is made a function of its rank in the population.
2) Evolution strategies typically abstracts coding structures as analogues of individuals. Evolutionary programming typically abstracts coding structures as analogues of distinct species (reproductive populations). Therefore, evolution strategies may use recombination operations to generate new trials [111], but evolutionary programming does not, as there is no sexual communication between species [100].

The current efforts in evolution strategies and evolutionary programming follow lines of investigation similar to those in genetic algorithms: 1) developing mathematical foundations for the procedures [51], [111], [113], investigating their computational complexity theoretically and empirically [114], [115] and combining evolutionary optimization with more traditional search techniques [116]; 2) using evolutionary algorithms to train and design neural networks [117]–[121]; 3) examining evolutionary algorithms for system identification, control, and robotics applications [122]–[127], as well as pattern recognition problems [128]–[130], along with the possibility for synergism between evolutionary and fuzzy systems [131], [132]; 4) applying evolutionary optimization to machine learning [133]; 5) relating evolutionary models to biological observations or applications [107], [134]–[137]; and also 6) designing evolutionary algorithms for implementation on parallel processing machines [138], [139], [140]. The most recent investigations can be found in [141], [142].

IV. Summary

Simulated evolution has a long history. Similar ideas and implementations have been independently invented numerous times. There are currently three main lines of investigation: genetic algorithms, evolution strategies, and evolutionary programming. These methods share many similarities. Each maintains a population of trial solutions, imposes random changes to those solutions, and incorporates the use of selection to determine which solutions to maintain into future generations and which to remove from the pool of trials. But these methods also have important differences. Genetic algorithms emphasize models of genetic operators as observed in nature, such as crossing over, inversion, and point mutation and apply these to abstracted chromosomes. Evolution strategies and evolutionary programming emphasize mutational transformations that maintain behavioral linkage between each parent and its offspring, respectively, at the level of the individual or the species. Recombination may be appropriately applied to individuals, but is not applicable for species.

No model can be a complete description of the true system. Each of the three possible evolutionary approaches described above is incomplete. But each has also been demonstrated to be of practical use when applied to difficult optimization problems. The greatest potential for the application of evolutionary optimization to real-world problems will come from their implementation on parallel machines, for evolution is an inherently parallel process. Recent advances in distributed processing architectures will result in dramatically reduced execution times for simulations that would simply be impractical on current serial computers.

Natural evolution is a robust yet efficient problem-solving technique. Simulated evolution can be made as robust. The same procedures can be applied to diverse problems with relatively little reprogramming. While such efforts will undoubtedly continue to address difficult real-world problems, the ultimate advancement of the field will, as always, rely on the careful observation and abstraction of the natural process of evolution.

Acknowledgment

The author is grateful to W. Atmar, T. Bäck, L. Davis, G. B. Fogel, L. J. Fogel, E. Mayr, Z. Michalewicz, G. Rudolph, H.-P. Schwefel, and the anonymous referees for their helpful comments and criticisms of this review.

References

[1] M. S. Bazaraa and C. M. Shetty, *Nonlinear Programming*, New York: John Wiley, 1979.
[2] B. D. O. Anderson, R. R. Bitmead, C. R. Johnson, P. V. Kokotovic, R. L. Kosut, I. M. Y. Mareels, L. Praly, and B. D. Riedle, *Stability of Adaptive Systems: Passivity and Averaging Analysis*. Cambridge, MA: MIT Press, 1986.
[3] P. Werbos, "Beyond regression: new tools for prediction and analysis in the behavioral sciences," Doctoral dissertation, Harvard University, 1974.
[4] D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*. vol. 1, Cambridge, MA: MIT Press, 1986.
[5] L. Ljung, *System Identification: Theory for the User*, Englewood Cliffs, NJ: Prentice-Hall, 1987.
[6] A. Hoffman, *Arguments on Evolution: A Paleontologist's Perspective*, New York: Oxford University Press, 1988.
[7] T. R. Malthus, *An Essay on the Principle of Population, as it Affects the Future Improvement of Society*, 6th ed., London: Murray, 1826.
[8] E. Mayr, *The Growth of Biological Thought: Diversity, Evolution and Inheritance*, Cambridge, MA: Belknap Press, 1988.
[9] J. Huxley, "The evolutionary process," in *Evolution as a Process*, J. Huxley, A. C. Hardy, and E. B. Ford, Eds. New York: Collier Books, pp. 9–33, 1963.
[10] D. E. Wooldridge, *The Mechanical Man: The Physical Basis of Intelligent Life*. New York: McGraw-Hill, 1968.

[11] W. Atmar, "The inevitability of evolutionary invention," unpublished manuscript, 1979.
[12] S. Wright, "Evolution in Mendelian populations," *Genetics,* vol. 16, pp. 97–159, 1931.
[13] S. Wright, "The evolution of life," Panel discussion in *Evolution After Darwin: Issues in Evolution,* vol. III, S. Tax and C. Callender, Eds. Chicago: Univ. of Chicago Press, 1960.
[14] G. G. Simpson, *The Meaning of Evolution: A Study of the History of Life and Its Significance for Man.* New Haven, CT: Yale Univ. Press, 1949.
[15] T. Dobzhansky, *Genetics of the Evolutionary Processes.* New York: Columbia Univ. Press, 1970.
[16] S. M. Stanley, "A theory of evolution above the species level," *Proc. Nat. Acad. Sci.,* vol. 72, no. 2, pp. 646–650, 1975.
[17] E. Mayr, "Where are we?" *Cold Spring Harbor Symp. Quant. Biol.,* vol. 24, pp. 409–440, 1959.
[18] E. Mayr, *Animal Species and Evolution.* Cambridge, MA: Belknap Press, 1963.
[19] E. Mayr, *Toward a New Philosophy of Biology: Observations of an Evolutionist.* Cambridge, MA: Belknap Press, 1988.
[20] R. Dawkins, *The Blind Watchmaker.* Oxford: Clarendon Press, 1986.
[21] S. Wright, "The roles of mutation, inbreeding, crossbreeding, and selection in evolution," *Proc. 6th Int. Cong. Genetics, Ithaca,* vol. 1, pp. 356–366, 1932.
[22] R. C. Lewontin, *The Genetic Basis of Evolutionary Change.* New York: Columbia University Press, NY, 1974.
[23] P. H. Raven and G. B. Johnson, *Biology,* St. Louis, MO: Times Mirror, 1986.
[24] A. S. Fraser, "Simulation of genetic systems by automatic digital computers. I. Introduction," *Australian J. of Biol. Sci.,* vol. 10, pp. 484–491, 1957.
[25] A. S. Fraser, "Simulation of genetic systems by automatic digital computers. II. Effects of linkage on rates of advance under selection," *Australian J. of Biol. Sci.,* vol. 10, pp. 492–499, 1957.
[26] A. S. Fraser, "Simulation of genetic systems by automatic digital computers. IV. Epistasis," *Australian J. of Biol. Sci.,* vol. 13, pp. 329–346, 1960.
[27] A. S. Fraser, "Simulation of genetic systems," *J. of Theor. Biol.,* vol. 2, pp. 329–346, 1962.
[28] A. S. Fraser, "The evolution of purposive behavior," in *Purposive Systems,* H. von Foerster, J. D. White, L. J. Peterson, and J. K. Russell, Eds. Washington, DC: Spartan Books, pp. 15–23, 1968.
[29] H. J. Bremermann, "The evolution of intelligence. The nervous system as a model of its environment," Technical Report No. 1, Contract No. 477(17), Dept. of Mathematics, Univ. of Washington, Seattle, 1958.
[30] H. J. Bremermann, "Optimization through evolution and recombination," in *Self-Organizing Systems.* M. C. Yovits, G. T. Jacobi, and G. D. Goldstine, Eds. Washington, DC: Spartan Books, pp. 93–106, 1962.
[31] H. J. Bremermann, "Quantitative aspects of goal-seeking self-organizing systems," in *Progress in Theoretical Biology,* vol. 1, New York: Academic Press, pp. 59–77, 1967.
[32] H. J. Bremermann, "Numerical Optimization Procedures Derived from Biological Evolution Processes," in *Cybernetic Problems in Bionics,* H. L. Oestreicher and D. R. Moore, Eds. New York: Gordon & Breach, pp. 543–562, 1968.
[33] H. J. Bremermann, "On the Dynamics and Trajectories of Evolution Processes," in *Biogenesis, Evolution, Homeostasis.* A. Locker, Ed. New York: Springer-Verlag, pp. 29–37, 1973.
[34] H. J. Bremermann and M. Rogson, "An Evolution-Type Search Method for Convex Sets," ONR Technical Report, Contracts 222(85) and 3656(58), UC Berkeley, 1964.
[35] H. J. Bremermann, M. Rogson, and S. Salaff, "Search by Evolution," in *Biophysics and Cybernetic Systems.* M. Maxfield, A. Callahan, and L. J. Fogel, Eds. Washington, DC: Spartan Books, pp. 157–167, 1965.
[36] H. J. Bremermann, M. Rogson, and S. Salaff, "Global Properties of Evolution Processes," in *Natural Automata and Useful Simulations.* H. H. Pattee, E. A. Edlsack, L. Fein, and A. B. Callahan, Eds. Washington, DC: Spartan Books, pp. 3–41, 1966.
[37] J. Reed, R. Toombs, and N. A. Barricelli, "Simulation of biological evolution and machine learning," *Journal of Theoretical Biology,* vol. 17, pp. 319–342, 1967.
[38] J. H. Holland, "Adaptive plans optimal for payoff-only environments," *Proc. of the 2nd Hawaii Int. Conf. on System Sciences,* pp. 917–920, 1969.
[39] J. H. Holland, *Adaptation in Natural and Artificial Systems.* Ann Arbor: Univ. Of Michigan Press, 1975.
[40] K. A. De Jong, "The analysis of the behavior of a class of genetic adaptive systems," Doctoral dissertation, Univ. of Michigan, Ann Arbor, 1975.
[41] N. N. Schraudolph and R. K. Belew, "Dynamic parameter encoding for genetic algorithms," *Machine Learning,* vol. 9, no. 1, pp. 9–21, 1992.
[42] G. A. Vignaux and Z. Michalewicz, "A genetic algorithm for the linear transportation problem," *IEEE Trans. on Systems, Man and Cybernetics,* vol. 21, no. 2, pp. 445–452, 1991.
[43] J. Antonisse, "A new interpretation of schema notation that overturns the binary encoding constraint," *Proc. of the Third International Conf. on Genetic Algorithms,* J. D. Schaffer, Ed. San Mateo, CA: Morgan Kaufmann Publishers, pp. 86–91, 1989.
[44] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs.* New York: Springer-Verlag, 1992.
[45] D. J. Montana, "Automated parameter tuning for interpretation of synthetic images," in *Handbook of Genetic Algorithms.* L. Davis, Ed. New York: Van Nostrand Reinhold, pp. 282–311, 1991.
[46] G. Syswerda, "Schedule optimization using genetic algorithms," in *Handbook of Genetic Algorithms,* L. Davis, Ed. New York: Van Nostrand Reinhold, pp. 332–349, 1991.
[47] A. H. Wright, "Genetic algorithms for real parameter optimization," *Foundations of Genetic Algorithms,* G. J. E. Rawlins, Ed. San Mateo, CA: Morgan Kaufmann Publishers, pp. 205–218, 1991.
[48] G. Rudolph, "Convergence properties of canonical genetic algorithms," *IEEE Trans. on Neural Networks,* vol. 5. no. 1, 1994.
[49] J. J. Grefenstette, "Optimization of control parameters for genetic algorithms," *IEEE Trans. Sys., Man and Cybern.,* vol. 16, no. 1, pp. 122–128, 1986.
[50] A. E. Eiben, E. H. Aarts, and K. M. Van Hee, "Global convergence of genetic algorithms: An infinite Markov chain analysis," *Parallel Problem Solving from Nature,* H.-P. Schwefel and R. Männer, Eds. Heidelberg, Berlin: Springer-Verlag, pp. 4–12, 1991.
[51] D. B. Fogel, "Asymptotic convergence properties of genetic algorithms and evolutionary programming: Analysis and experiments," *Cybernetics and Systems,* in press, 1994.
[52] L. Davis, Ed. *Handbook of Genetic Algorithms,* New York: Van Nostrand Reinhold, 1991.
[53] D. E. Goldberg, "Computer-aided gas pipeline operation using genetic algorithms and rule learning," Doctoral dissertation, Univ. of Michigan, Ann Arbor, 1983.
[54] J. J. Grefenstette, R. Gopal, B. Rosmaita, and D. Van Gucht, "Genetic algorithms for the traveling salesman problem," in *Proc. of an Intern. Conf. on Genetic Algorithms and Their Applications,* J. J. Grefenstette, Ed. Lawrence Earlbaum, pp. 160–168, 1985.
[55] G. Syswerda, "Uniform crossover in genetic algorithms," in *Proc. of the Third Intern. Conf. on Genetic Algorithms,* J. D. Schaffer, Ed. San Mateo, CA: Morgan Kaufmann, pp. 2–9, 1989.
[56] A. S. Bickel and R. W. Bickel, "Determination of near-optimum use of hospital diagnostic resources using the 'GENES' genetic algorithm shell," *Comput. Biol. Med.,* vol. 20, no. 1, pp. 1–13, 1990.
[57] G. Pitney, T. R. Smith, and D. Greenwood, "Genetic design of processing elements for path planning networks," *Proc. of the Int. Joint Conf. on Neural Networks 1990,* vol. III, IEEE, pp. 925–932, 1990.
[58] N. N. Schraudolph, personal communication, UCSD, 1992.
[59] J. H. Holland, "Genetic algorithms," *Scientific American,* pp. 66–72, July, 1992.
[60] K. A. De Jong, "Are genetic algorithms function optimizers?" *Proc. of the Sec. Parallel Problem Solving from Nature Conf.,* R. Männer and B. Manderick, Eds. The Netherlands: Elsevier Science Press, pp. 3–14, 1992.
[61] T. E. Davis and J. C. Principe, "A simulated annealing like convergence theory for the simple genetic algorithm," *Proc. of the Fourth Intern. Conf. on Genetic Algorithms,* R. K. Belew and L. B. Booker, Eds. San Mateo, CA: Morgan Kaufmann, pp. 174–181, 1991.
[62] X. Qi and F. Palmieri, "Adaptive mutation in the genetic algorithm," *Proc. of the Sec. Ann. Conf. on Evolutionary Programming,* D.B. Fogel and W. Atmar, Eds. La Jolla, CA: Evolutionary Programming Society, pp. 192–196, 1993.
[63] G. E. Liepins and M. D. Vose, "Deceptiveness and genetic algorithm dynamics," in *Foundations of Genetic Algorithms,* G. J. E. Rawlins, Ed. San Mateo, CA: Morgan Kaufmann, pp. 36–52, 1991.
[64] L. D. Whitley, "Fundamental principles of deception in genetic search," in *Foundations of Genetic Algorithms,* G. J. E. Rawlins, Ed. San Mateo, CA: Morgan Kaufmann, pp. 221–241, 1991.
[65] W. E. Hart and R. K. Belew, "Optimizing an arbitrary function is hard for the genetic algorithm," *Proc. of the Fourth Intern. Conf. on Genetic Algorithms,* R. K. Belew and L. B. Booker, Eds. San Mateo, CA: Morgan Kaufmann, pp. 190–195, 1991.
[66] S. Forrest and M. Mitchell, "What makes a problem hard for a genetic algorithm?" *Machine Learning.* vol. 13, no. 2–3, pp. 285–319, 1993.
[67] J. D. Schaffer and L. J. Eshelman, "On crossover as an evolutionarily

viable strategy," in *Proc. of the Fourth Intern. Conf. on Genetic Algorithms*, R. K. Belew and L. B. Booker, Eds. San Mateo, CA: Morgan Kaufmann, pp. 61–68, 1991.
[68] W. M. Spears and K. A. De Jong, "On the virtues of parameterized uniform crossover," in *Proc. of the Fourth Intern. Conf. on Genetic Algorithms*, R. K. Belew and L. B. Booker, Eds. San Mateo, CA: Morgan Kaufmann, pp. 230–236, 1991.
[69] D. E. Goldberg, K. Deb, and J. H. Clark, "Genetic algorithms: noise, and the sizing of populations," *Complex Systems*, vol. 6, pp. 333–362, 1992.
[70] V. Kreinovich, C. Quintana, and O. Fuentes, "Genetic algorithms—what fitness scaling is optimal," *Cybernetics and Systems*, vol. 24, no. 1, pp. 9–26, 1993.
[71] S. W. Mahfoud and D. E. Goldberg, "Parallel recombinative simulated annealing: A genetic algorithm," IlliGAL Report No. 92002, Univ. of Illinois, Urbana-Champaign, 1992.
[72] L. Ingber and B. Rosen, "Genetic algorithms and very fast simulated annealing—a comparison," *Math. and Comp. Model.*, vol. 16, no. 11, pp. 87–100, 1992.
[73] D. Adler, "Genetic algorithms and simulated annealing: A marriage proposal," in *IEEE Conference on Neural Networks 1993*, pp. 1104–1109, 1993.
[74] J. R. Koza, "A hierarchical approach to learning the boolean multiplexer function," in *Foundations of Genetic Algorithms*. G. J. E. Rawlins, Ed. San Mateo, CA: Morgan Kaufmann, pp. 171–192, 1991.
[75] J. R. Koza, *Genetic Programming*. Cambridge, MA: MIT Press, 1992.
[76] S. Forrest and G. Mayer-Kress, "Genetic algorithms, nonlinear dynamical systems, and models of international security," *Handbook of Genetic Algorithms*, L. Davis, Ed. New York: Van Nostrand Reinhold, pp. 166–185, 1991.
[77] J. R. Koza, "Hierarchical automatic function definition in genetic programming," *Foundations of Genetic Algorithms 2*, L. D. Whitley, Ed. San Mateo, CA: Morgan Kaufmann, pp. 297–318, 1992.
[78] K. Kristinsson and G. A. Dumont, "System identification and control using genetic algorithms," *IEEE Trans. Sys., Man and Cybern.*, vol. 22, no. 5, pp. 1033–1046, 1992.
[79] K. Krishnakumar and D. E. Goldberg, "Control system optimization using genetic algorithms," *Journ. of Guidance, Control and Dynamics*, vol. 15, no. 3, pp. 735–740, 1992.
[80] J. H. Holland, "Concerning the emergence of tag-mediated lookahead in classifier systems," *Physica D*, vol. 42, pp. 188–201, 1990.
[81] S. Forrest and J. H. Miller, "Emergent behavior in classifier systems," *Physica D*, vol. 42, pp. 213–227, 1990.
[82] R. L. Riolo, "Modeling simple human category learning with a classifier system," in *Proc. of the Fourth Intern. Conf. on Genetic Algorithms*, R. K. Belew and L. B. Booker, Eds. San Mateo, CA: Morgan Kaufmann, pp. 324–333, 1991.
[83] G. E. Liepins, M. R. Hilliard, M. Palmer and G. Rangarajan, "Credit assignment and discovery in classifier systems," *Intern. Journ. of Intelligent Sys.*, vol. 6, no. 1, pp. 55–69, 1991.
[84] S. Tokinaga and A.B. Whinston, "Applying adaptive credit assignment algorithms for the learning classifier system based upon the genetic algorithm," *IEICE Trans. on Fund. Elec. Comm. and Comp. Sci.*, vol. E75A, no. 5, pp. 568–577, 1992.
[85] D. Jefferson, R. Collins, C. Cooper, M. Dyer, M. Flowers, R. Korf, C. Taylor, and A. Wang, "Evolution as a theme in artificial life: The Genesys/Tracker system," in *Artificial Life II*, C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, Eds. Reading, MA: Addison-Wesley, pp. 549–578, 1991.
[86] J. H. Holland, *Adaptation in Natural and Artificial Systems*. 2nd ed., Cambridge, MA: MIT Press, 1992.
[87] H. Mühlenbein, "Evolution in time and space—the parallel genetic algorithm," in *Foundations of Genetic Algorithms*, G. J. E. Rawlins, Ed. San Mateo, CA: Morgan Kaufmann, pp. 316–337, 1991.
[88] P. Spiessens and B. Manderick, "A massively parallel genetic algorithm: implementation and first analysis," in *Proc. of the Fourth Intern. Conf. on Genetic Algorithms*, R. K. Belew and L. B. Booker, Eds. San Mateo, CA: Morgan Kaufmann, pp. 279–286, 1991.
[89] H. Mühlenbein, M. Schomisch and J. Born, "The parallel genetic algorithm as function optimizer," *Parallel Computing*, vol. 17, pp. 619–632, 1991.
[90] S. Forrest, Ed., *Proc. of the Fifth Intern. Conf. on Genetic Algorithms*, San Mateo, CA: Morgan Kaufmann, 1993.
[91] H.-P. Schwefel, "Kybernetische evolution als strategie der experimentellen forschung in der strmungstechnik," Diploma thesis, Technical Univ. of Berlin, 1965.
[92] I. Rechenberg, *Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biolgischen evolution*. Stuttgart: Frommann-Holzboog Verlag, 1973.
[93] L. J. Fogel, "Autonomous automata," *Industrial Research*, vol. 4, pp. 14–19, 1962.
[94] L. J. Fogel, "On the organization of intellect," Doctoral dissertation, UCLA, 1964.
[95] M. Conrad, "Evolutionary learning circuits," *Journ. Theor. Biol.*, vol. 46, pp. 167–188, 1974.
[96] M. Conrad and H. H. Pattee, "Evolution experiments with an artificial ecosystem," *Journ. Theor. Biol.*, vol. 28, pp. 393–409, 1970.
[97] G. H. Burgin, "On playing two-person zero-sum games against nonminimax players," *IEEE Trans. on Systems Science and Cybernetics*, vol. SSC-5, no. 4, pp. 369–370, 1969.
[98] G. H. Burgin, "System identification by quasilinearization and evolutionary programming," *Journal of Cybernetics*, vol. 2, no. 3, pp. 4–23, 1974.
[99] J. W. Atmar, "Speculation on the evolution of intelligence and its possible realization in machine form," Doctoral dissertation, New Mexico State University, Las Cruces, 1976.
[100] D. B. Fogel, "On the philosophical differences between genetic algorithms and evolutionary algorithms," in *Proc. of the Sec. Ann. Conf. on Evolutionary Programming*, D. B. Fogel and W. Atmar, Eds. La Jolla, CA: Evolutionary Programming Society, pp. 23–29, 1993.
[101] F. J. Solis and R. J.-B. Wets, "Minimization by random search techniques," *Math. Operations Research*, vol. 6, pp. 19–30, 1981.
[102] T. Bäck and H.-P. Schwefel, "An Overview of Evolutionary Algorithms for Parameter Optimization," *Evolutionary Computation*, vol. 1, no. 1, pp. 1–24, 1993..
[103] H.-P. Schwefel, "Numerische optimierung von computer-modellen mittels der evoluionsstrategie," *Interdisciplinary systems research*, vol. 26, Basel: Birkhuser, 1977.
[104] H.-P. Schwefel, *Numerical Optimization of Computer Models*. Chichester, UK: John Wiley, 1981.
[105] G. H. Mealy, "A method of synthesizing sequential circuits," *Bell Sys. Tech. Journ.*, vol. 34, pp. 1054–1079, 1955.
[106] D. B. Fogel, "The evolution of intelligent decision making in gaming," *Cybernetics and Systems*, vol. 22, pp. 223–236, 1991.
[107] D. B. Fogel, "Evolving behaviors in the iterated prisoner's dilemma," *Evolutionary Computation*, vol. 1, no. 1, pp. 77–97, 1993.
[108] R. Axelrod, "The evolution of strategies in the iterated prisoner's dilemma," in *Genetic Algorithms and Simulated Annealing*, L. Davis, Ed. London: Pitman Publishing, pp. 32–41, 1987.
[109] D. B. Fogel, L. J. Fogel, and W. Atmar, "Meta-evolutionary programming," in *Proc. of the 25th Asilomar Conf. on Signals, Systems and Computers*, R. R. Chen, Ed. IEEE Computer Society, pp. 540–545, 1991.
[110] D. B. Fogel, L. J. Fogel, W. Atmar, and G. B. Fogel, "Hierarchic methods of evolutionary programming," in *Proc. of the First Ann. Conf. on Evolutionary Programming*, D. B. Fogel and W. Atmar, Eds. La Jolla, CA: Evolutionary Programming Society, pp. 175–182, 1992.
[111] T. Bäck, G. Rudolph, and H.-P. Schwefel, "Evolutionary programming and evolution strategies: similarities and differences," in *Proc. of the Second Ann. Conf. on Evolutionary Programming*, D. B. Fogel and W. Atmar, Eds. La Jolla, CA: Evolutionary Programming Society, pp. 11–22, 1993.
[112] N. Saravanan, "Learning of Strategy Parameters in Evolutionary Programming," *Proc. of Third Annual Conference on Evolutionary Programming*, A. V. Sebald and L. J. Fogel, Eds. RiverEdge, NJ: World Scientific, to appear, 1994.
[113] G. Rudolph, "On correlated mutations in evolution strategies," in *Parallel Problem Solving from Nature 2*, R. Männer and B. Manderick, Eds. The Netherlands: Elsevier Science Press, pp. 105–114, 1992.
[114] B. K. Ambati, J. Ambati, and M. M. Mokhtar, "Heuristic combinatorial optimization by simulated darwinian evolution: A polynomial time algorithm for the traveling salesman problem," *Biological Cybernetics*, vol. 65, pp. 31–35, 1991.
[115] D. B. Fogel, "Empirical estimation of the computation required to discover approximate solutions to the traveling salesman problem using evolutionary programming," in *Proc. of the Second Ann. Conf. on Evolutionary Programming*, D. B. Fogel and W. Atmar, Eds. La Jolla, CA: Evolutionary Programming Society, in press, 1993.
[116] D. Waagen, P. Diercks, and J. R. McDonnell, "The stochastic direction set algorithm: A hybrid technique for finding function extrema," in *Proc. of the First Ann. Conf. on Evolutionary Programming*, D. B. Fogel and W. Atmar, Eds. La Jolla, CA: Evolutionary Programming Society, pp. 35–42, 1992.
[117] R. Lohmann, "Structure evolution and incomplete induction," in *Proc. of the Sec. Parallel Problem Solving from Nature Conf.*, R. Männer and B. Manderick, Eds. The Netherlands: Elsevier Science Press, pp. 175–186, 1992.

[118] J. R. McDonnell and D. Waagen, "Evolving neural network connectivity," *Intern. Conf. on Neural Networks 1993,* IEEE, pp. 863–868, 1993.
[119] P. J. Angeline, G. Saunders and J. Pollack, "An evolutionary algorithm that constructs neural networks," *IEEE Trans. Neural Networks,* vol. 5, no 1, 1994.
[120] D. B. Fogel, "Using evolutionary programming to create neural networks that are capable of playing tic-tac-toe," *Inern. Conf. on Neural Networks 1993,* IEEE, pp. 875–880, 1993.
[121] R. Smalz and M. Conrad, "Evolutionary credit apportionment and time-dependent neural processing," in *Proc. of the Second Ann. Conf. on Evolutionary Programming,* D. B. Fogel and W. Atmar, Eds. La Jolla, CA: Evolutionary Programming Society, pp. 119–126, 1993.
[122] W. Kuhn and A. Visser, "Identification of the system parameter of a 6 axis robot with the help of an evolution strategy," *Robotersysteme,* vol. 8, no. 3, pp. 123–133, 1992.
[123] J. R. McDonnell, B. D. Andersen, W. C. Page and F. Pin, "Mobile manipulator configuration optimization using evolutionary programming," in *Proc. of the First Ann. Conf. on Evolutionary Programming,* D. B. Fogel and W. Atmar, Eds. La Jolla, CA: Evolutionary Programming Society, pp. 52–62, 1992.
[124] W. C. Page, B. D. Andersen, and J. R. McDonnell, "An evolutionary programming approach to multi-dimensional path planning," in *Proc. of the First Ann. Conf. on Evolutionary Programming,* D. B. Fogel and W. Atmar, Eds. La Jolla, CA: Evolutionary Programming Society, pp. 63–70, 1992.
[125] A. V. Sebald, J. Schlenzig, and D. B. Fogel, "Minimax design of CMAC encoded neural controllers for systems with variable time delay," in *Proc. of the First Ann. Conf. on Evolutionary Programming,* D. B. Fogel and W. Atmar, Eds. La Jolla, CA: Evolutionary Programming Society, pp. 120–126, 1992.
[126] D. B. Fogel, *System Identification Through Simulated Evolution: A Machine Learning Approach to Modeling.* Needham, MA: Ginn Press, 1991.
[127] D. B. Fogel, "Using evolutionary programming for modeling: An ocean acoustic example," *IEEE Journ. on Oceanic Engineering,* vol. 17, no. 4, pp. 333–340, 1992.
[128] V. W. Porto, "Alternative methods for training neural networks," in *Proc. of the First Ann. Conf. on Evolutionary Programming,* D. B. Fogel and W. Atmar, Eds. La Jolla, CA: Evolutionary Programming Society, pp. 100–110, 1992.
[129] L. A. Tamburino, M. A. Zmuda and M. M. Rizki, "Applying evolutionary search to pattern recognition problems," in *Proc. of the Sec. Ann. Conf. on Evolutionary Programming,* D. B. Fogel and W. Atmar, Eds. Evolutionary Programming Society, La Jolla, CA, pp. 183–191, 1993.
[130] M. M. Rizki, L. A. Tamburino and M. A. Zmuda, "Evolving multi-resolution feature detectors," in *Proc. of the Sec. Ann. Conf. on Evolutionary Programming,* D. B. Fogel and W. Atmar, Eds. La Jolla, CA: Evolutionary Programming Society, pp. 108–118, 1993.
[131] D. B. Fogel and P. K. Simpson, "Evolving fuzzy clusters," *Intern. Conf. on Neural Networks 1993,* IEEE, pp. 1829–1834, 1993.
[132] S. Haffner and A. V. Sebald, "Computer-aided design of fuzzy HVAC controllers using evolutionary programming," in *Proc. of the Sec. Ann. Conf. on Evolutionary Programming,* D. B. Fogel and W. Atmar, Eds. La Jolla, CA: Evolutionary Programming Society, pp. 98–107, 1993.
[133] S. H. Rubin, "Case-based learning: A new paradigm for automated knowledge acquisition," *ISA Transactions,* Special Issue on Artif. Intell. for Eng., Design and Manuf., vol. 31, pp. 181–209, 1992.
[134] W. Atmar, "Notes on the simulation of evolution," *IEEE Trans. on Neural Networks,* vol. 5. no. 1, 1994.
[135] G. B. Fogel, "An introduction to the protein folding problem and the potential application of evolutionary programming," in *Proc. of the Sec. Ann. Conf. on Evolutionary Programming,* D. B. Fogel and W. Atmar, Eds. La Jolla, CA: Evolutionary Programming Society, pp. 170–177, 1993.
[136] M. Conrad, "Molecular computing: The lock-key paradigm," *Computer,* Special Issue on Molecular Computing, M. Conrad, Ed. Nov., pp. 11–20, 1992.
[137] J. O'Callaghan and M. Conrad, "Symbiotic interactions in the EVOLVE III ecosystem model," *BioSystems,* vol. 26, pp. 199–209, 1992.
[138] G. Rudolph, "Parallel approaches to stochastic global optimization," in *Parallel Computing: From Theory to Sound Practice.* W. Joosen and E. Milgrom, Eds. Amsterdam: IOS Press pp. 256–267, 1992.
[139] B. S. Duncan, "Parallel evolutionary programming," in *Proc. of the Sec. Ann. Conf. on Evolutionary Programming,* D.B. Fogel and W. Atmar, Eds. La Jolla, CA: Evolutionary Programming Society, pp. 202–208, 1993.
[140] F. Hoffmeister, "Scalable Parallelism by Evolutionary Algorithms," in *Parallel Comp. & Math. Opt.,* D. B. Grauer, Ed. Heidelberg, Berlin: Springer-Verlag, pp. 177–198, 1991.
[141] D. B. Fogel and W. Atmar Eds., *Proc. of the Sec. Ann. Conf. on Evolutionary Programming,* La Jolla, CA: Evolutionary Programming Society, 1993.
[142] R. Männer and B. Manderick, Eds., *Proc. of the Sec. Parallel Problem Solving from Nature Conf.* The Netherlands: Elsevier Science Press, 1992.
[143] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning.* Reading, MA: Addison Wesley, 1989.
[144] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial Intelligence through Simulated Evolution.* New York: John Wiley, 1966.

David B. Fogel (M'89), for a photograph and biography, see this issue, p. 2.