Review of
L. D. Davis,
Handbook of Genetic Algorithms.
New York: Van Nostrand Reinhold, 1991.

Melanie Mitchell
Santa Fe Institute
1399 Hyde Park Road
Santa Fe, NM 87501
Email: mm@santafe.edu

Mimicking biological evolution and harnessing its power for adaptation are problems that have intrigued computer scientists for at least four decades. Genetic algorithms (GAs), invented by John Holland in the 1960s, are the most widely used approaches to computational evolution. In his book *Adaptation in Natural and Artificial Systems* (Holland, 1992, also reviewed in this issue), Holland presented GAs in a general theoretical framework for adaptation in nature. Holland's motivation was largely scientific— he was attempting to understand and link diverse types of natural phenomena—but he also proposed potential engineering applications of GAs. Since the publication of Holland's book, the field of GAs has grown into a significant sub-area of artificial intelligence and machine learning. Nowadays one can find several international conferences each year as well as a number of journals devoted to GAs and other "evolutionary computation" approaches. Research on GAs has spread from computer science to engineering and, more recently, to fields such as molecular biology, immunology, economics, and physics.

One result of this growth in interest has been a division of the field of GAs into several subspecies. One major division is between research on GAs as engineering tools and research on GAs as scientific models of evolutionary processes. This split roughly parallels the split of artificial intelligence research into work on engineering tools and work on models of cognition. In the GA world, in spite of some overlap between these two groups, it is usually clear which camp a particular project falls into. In scanning proceedings of the International Conference on Genetic Algorithms (e.g., Goodman, 1997) or the Parallel Problem Solving from Nature conference (e.g., Voigt, 1996), one finds that most papers have a distinct engineering bent. GA evolutionary modelers tend to publish instead in the Artificial Life and European Artificial Life conference proceedings, in the Simulation of Adaptive Behavior conference proceedings, or in journals such as *Adaptive Behavior* or *Biological Cybernetics*. A third group concentrates on the theory of GAs, usually apart from any particular application or model. These theorists tend to publish in the Foundations of Genetic Algorithms conference proceedings or in special issues of journals such as *Annals of Mathematics and Artificial Intelligence*. There is less communication among the various groups than one might hope for.

Lawrence Davis' book, *Handbook of Genetic Algorithms*, is squarely in the engineering camp. Davis, formally of Bolt, Beranek, and Newman, Inc. and Texas Instruments, is the founder and current president of Tica Associates, a consulting firm that assists industrial and government clients in applying GAs to various problems. Davis' main interest is in applying GAs to real-world problems, not in using GAs to learn more about natural systems.

Handbook of Genetic Algorithms strongly reflects this motivation. It is a practical guide for people who want to apply GAs to real problems. Beyond a simple statement of Holland's "Schema Theorem", there is no attempt to present or develop any theory of GAs or to make any claims about GAs as models of evolution. In fact, Davis' philosophy is to move far away from biological realism. For example, he recommends using numerical representations and hybrid search methods, and he advocates keeping explicit statistics on the success of crossover and mutation in order to adaptively vary their rates. Early on, Davis dismisses the GA's use to date as a biological model, saying,

> You should know that, although the findings of evolutionary biologists inspired the field of genetic algorithms in its early years, and although the findings of biologists and geneticists continue to influence the field somewhat, this influence is for the most part unidirectional. I know of no genetic algorithm application in the area of genetics, nor, to my knowledge, have the findings in our field impacted the theories of biologists. (p. 3)

This statement is a bit outdated, since there is now a growing literature of GA applications in molecular genetics (e.g., DNA fragment assembly, Cedeno & Vemuri, 1993; Parsons, Forrest, & Burks, 1993) and in human genetics (e.g., clustering of genetic traits for predicting disease, Congdon, Sing, & Reilly, 1993); but Davis is correct in saying that theoretical biology has not to date been widely influenced by findings in GAs. My suspicion is that this will change as modelers incorporate more biologically realistic elements into their GAs such as endogenous rather than explicit fitness, diploidy, co-evolution, and interactions between evolution and learning. There is already some interest in GA-like models in the theoretical population genetics literature (e.g., Bergman & Feldman, 1992) and in the ecology literature (e.g., Koza, Rice, & Roughgarden, 1992).

The *Handbook of Genetic Algorithms* is meant to give just enough information about GAs to help an interested scientist or engineer apply them to a particular problem. The book does not give a history of computational evolution or a review of past and current research such as that given in various GA textbooks (Goldberg, 1989, Michalewicz, 1992, Fogel, 1995, Schwefel, 1995, Bäck, 1996, Mitchell, 1996). Instead, Davis' book consists of two main parts: a 100-page tutorial written by Davis, giving his admittedly idiosyncratic approach to GAs, and thirteen chapters ("case studies") by various authors describing real-world GA applications. There is also a short third part that describes two GA software packages—GENESIS (in C) and OOGA (in Common Lisp with CLOS)—available for order. An order form is provided in the book.

The tutorial in Part 1 is based on Davis' philosophy of GAs, which emphasizes carefully tailoring a GA to each application rather than attempting to develop a robust, general purpose algorithm that performs well on a range of problems. Davis asserts that the latter approach "is a goal orthogonal to that of producing the best optimization algorithms for a particular problem." (p. 64.) He makes the important point that while GAs are robust weak methods—they work reasonably well across a range of problems and are robust in the presence of noisy evaluations—they are almost never the best optimization method for any particular problem. Davis' approach is to tailor the GA to the problem at hand, to incorporate domain knowledge into the GA as much as possible, and to hybridize the GA with other optimization methods that work well. Assuming that there is a method currently in use for a given problem and the GA is to be hybridized to improve the current method, Davis gives three principals of hybridization:

1. *Use the current encoding.* That is, represent candidate solutions in the same way that they are represented under the current method. For example, suppose that the problem is to find weights for a neural network that will perform a classification task, and that back-propagation starting from random weights is the current method. Then a candidate solution should be encoded as it is for back-propagation: as a real-valued weight vector. Davis points out that until recently most GA practitioners have used bit-string representations to represent candidate solutions, largely because the theory of GAs as developed by Holland and others relies on such representations. But Davis finds such representations unnatural and unnecessary in most cases. In the neural-network example, a real-valued weight vector is a more natural representation for the problem, or at least is a more widely used and understood representation.

2. *Hybridize where possible.* That is, combine useful features of the current algorithm with the GA wherever possible. For the neural-network example, one might hybridize back-propagation with the GA by using back-propagation until no more improvements are found, and then using the GA on a population of mutants of the current weight vector in order to make bigger jumps in the search space via crossover.

3. *Adapt the genetic operators to the problem.* That is, invent new forms of mutation and crossover that are appropriate to the problem's natural encoding. For example, mutation on real-valued weight vectors might consist of randomly incrementing or decrementing particular weights. Likewise, crossover might consist of producing two offspring from two parents by exchanging sets of weights that are on the incoming links to a given node. An alternative crossover scheme might average the corresponding weights of the parents.

The expectation is that, with these guidelines, the hybrid algorithm will perform better than the current algorithm or the GA alone. Such adaptations of the GA to a particular problem and other to optimization methods are, according to Davis, the art of real-world application of GAs.

In the tutorial section, Davis guides the reader through his approach to GAs with a friendly and conversational tone. The tutorial starts with a description of the simplest GA, and in each subsequent chapter the simple version is augmented with a number of more sophisticated strategies. The effect of each new strategy is illustrated by graphing the new GA's performance (fitness versus number of evaluations) on the same numerical optimization problem called "binary f6". Each new strategy is shown to yield improved performance on that problem.

The simplest GA uses fitness-proportionate selection, single-point crossover, and single-point mutation. This is similar to the original GA proposed by Holland (1992). Subsequent versions include the following revisions:

- *Linear normalization*: Fitness rank rather than absolute fitness determines the number of expected offspring of an individual.
- *Elitism*: The best individual in the population is always saved in the next generation.
- *Steady-state reproduction*: Only one or two individuals in the population are replaced at a time.
- *Uniform crossover*: Rather than choosing a single point at which to cross the parents, each gene in the offspring is chosen randomly from one or the other parent.
- *Fitness-based selection of operators*: Each operator (e.g., crossover and mutation) has a "fitness" value associated with it and its rate of application depends on its fitness.
- *Interpolating operator fitnesses*: The fitness of each operator is interpolated during a run (e.g., crossover fitness starts high and is reduced while mutation fitness starts low and is increased).
- *Adapting operator fitnesses*: Statistics are kept on the performance of each operator to determine the extent to which it is, on average, generating improvements in the population. The operator's fitness (and thus its rate of application) is periodically modified during a run according to these statistics.

Davis also discusses a number of techniques for "order-based" GAs—a version of the GA for combinatorial optimization problems such as the Traveling Salesman Problem, where the GA's task is to find an optimal ordering of elements.

All of these ideas have been discussed in various places in the GA literature (fitness-based selection of operators and adapting operator fitnesses are Davis' original contributions). However, there is no general consensus as to when these modifications to the original GA will produce better performance. Davis does not discuss that issue in the tutorial; he merely demonstrates the extent to which each of these modifications improves GA performance on the particular optimization problem binary f6. There is not much attempt to argue that improved performance on binary f6 is a good indication of improved performance in general, so the reliance on binary f6 as the only illustration of the effects of these modifications may be misleading. However, Davis is not attempting to prove that these modifications will always improve the GA's performance; he is only stating that, in his experience, they tend to yield improvements. Most of the advice in the book has this quality. There are no formal or rigorous arguments given to show that a particular suggestion will work well; Davis' advice comes from his experience in the field rather than from any theoretical or even systematic empirical evidence. The tutorial does not attempt to give any detailed guidelines for the types of problems on which GAs will work well or will perform better than other search methods. Fair enough, since there is currently no rigorous understanding of this in the GA research community. (It must be said that similar theoretical problems plague methods such as neural networks and simulated annealing.) The question of how to best use GAs is currently being addressed by a number of theorists (e.g., see Belew and Vose, 1997). However, for most people who want to apply GAs to real problems, Davis' experience-based advice will be a very helpful starting point.

I say "starting point", because readers cannot expect to go away and instantly build a successful GA application. A good deal of tinkering with representations, operators, and parameters will typically be necessary. The tutorial does not give detailed advice on how to actually implement a GA—this is not a book of "GA recipes". However, some help can be found via the software packages Davis discusses, and also by reading the case studies found in Part 2 of the book. The case studies—each given as a chapter written by a different group of authors—detail real-world applications of GAs and GA hybrids. Many of these applications were done in industrial rather than university settings; names such as Lockheed, the US Bureau of Mines, Honeywell, General Electric, US West, and Hewlett Packard are the author affiliations in about half the chapters. This is a striking demonstration of the extent to which GAs have penetrated into industry and will probably soon be the source of a number of marketed products. In addition to the industries represented in this book, there is currently widespread work on using GAs in biotechnology and financial forecasting, among other applications.

These case studies taken together give a very good answer to the question, "What are some successful applications of GAs?" They include descriptions of how GAs have been used to optimize the parametric design of aircraft, air-injected hydrocyclones (mineral-separating devices), and in other engineering design problems. We also find descriptions of GAs optimizing call routing in a US West telecommunications network, path planning for robot-arm motion, and parameters for a model of international arms races. The GA is also seen evolving strategies (encoded as production systems) for aircraft missile avoidance, neural network architectures, diagnoses for faults in a microwave communication network, and DNA conformations based on spectrometric data. Finally, we see a GA optimizing parameters for an expert system that processes sonar signals, weights for a neural network that performs a sonar-signal classification task, schedules for activities in a laboratory, and tours for the Traveling Salesman Problem. These case studies make for very interesting reading for those involved in the applications side of GAs. Many of the case studies include detailed comparisons between the GA and other methods, and numerous examples of hybridization, generally following the guidelines Davis outlines in the tutorial section.

The *Handbook of Genetic Algorithms* is meant to be a practical guide for practitioners, not, say, a textbook for a machine learning course. As a high-level introduction, the tutorial serves this purpose well, and is strongly supplemented by the case studies. The book is clearly written and enjoyable to read, and, short of hiring Davis himself as a consultant, reading his book is probably the quickest and easiest way to get off the ground for a first real GA application.

References

T. Bäck (1996). *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms.* Oxford: Oxford University Press.

R. K. Belew and M. D. Vose (1997). *Foundations of Genetic Algorithms 4.* San Francisco, CA: Morgan Kaufmann.

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

W. Cedeno and V. Vemuri (1993). An investigation of DNA mapping with genetic algorithms: Preliminary results. In *Proceedings of the Fifth Workshop on Neural Networks: An International Conference on Computational Intelligence: Neural Networks, Fuzzy Systems, Evolutionary Programming, and Virtual Reality*, 133–140. San Diego, CA: The Society for Computer Simulation.

C. B. Congdon, C. F. Sing, and S. L. Reilly (1993). Genetic algorithms for identifying combinations of genes and other risk factors associated with coronary artery disease. *Proceedings of the Workshop on Artificial Intelligence and the Genome.* International Joint Conference on Artificial Intelligence. Chambery, France.

D. B. Fogel (1995). *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence.* IEEE Press.

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

E. Goodman (editor) (1997). *Proceedings of the Seventh International Conference on Genetic Algorithms.* San Francisco, CA: Morgan Kaufmann.

J. H. Holland, (1992). *Adaptation in natural and artificial systems.* Cambridge, MA: MIT Press. (First edition, 1975, University of Michigan Press.)

J. R. Koza, J. P. Rice, and J. Roughgarden (1992). Evolution of food foraging strategies for the Caribbean Anolis lizard using genetic programming. *Adaptive Behavior, 1(2)*, 47-74.

Z. Michalewicz (1992). *Genetic Algorithms + Data Structures = Evolution Programs.* Berlin: Springer-Verlag.

M. Mitchell (1996). *An Introduction to Genetic Algorithms.* Cambridge, MA: MIT Press.

R. Parsons, S. Forrest, and C. Burks (1993). Genetic operators for the DNA fragment assembly problem. Submitted to *Machine Learning.*

H.-P. Schwefel (1995). *Evolution and Optimum Seeking.* New York: Wiley.

H.-M. Voigt (editor) (1996). *Parallel problem solving from nature.* Berlin: Springer-Verlag (Lecture Notes in Computer Science).