Genetic Algorithms

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

ACM Computing Surveys (to appear)[1]
January, 12, 1996

A genetic algorithm is a computational model of biological evolution. Genetic algorithms are useful, both as search methods for solving problems and for modeling evolutionary systems. In genetic algorithms, binary strings are stored in a computer's memory, and over time, the strings are modified in much the same way that populations of individuals evolve under natural selection. Although the computational setting is highly simplified when compared with the natural world, genetic algorithms are capable of evolving surprisingly complex and interesting structures. These structures, called individuals, can represent solutions to problems, strategies for playing games, visual images, or computer programs.

Genetic algorithms are loosely based on ideas from population genetics. First, a population of individuals is created randomly. In the simplest case, each individual is a bit string and can be thought of as a candidate solution for some problem of interest, as shown in Figure 1. Variations among individuals in the population result in some individuals being more fit than others (e.g., better problem solutions). These differences are used to bias the selection of a new set of candidate solutions at the next time step, referred to as selection. During selection, a new population is created by making copies of more successful individuals and deleting less successful ones. However, the copies are not exact. There is a probability of mutation (random bit flips), crossover (exchange of corresponding substrings between two individuals), or other changes to the bit string during the copy operation. By transforming the previous set of good individuals to a new one, the mutation and crossover operations generate a new set of individuals, or samples, that ideally have a better than average chance of also being good. When this cycle of evaluation, selection, and genetic operations is it-

[1] Significant portions of this paper are excerpted with permission WHICH WE OBTAINED FOR CRC ARTICLE BUT STILL NEED TO GET FOR COMPUTING SURVEYS from S. Forrest, Science 261:872-878, Aug. 13, 1993 (Copyright 1993 American Association for the Advancement of Science) and from [4, 2, 3] NOTE: we also need to acknowledge that this comes from the longer CRC article.

erated for many generations, the overall fitness of the population generally improves, and the individuals in the population represent improved "solutions" to whatever problem was posed in the fitness function. Figure 1 shows an example of a genetic algorithm being used to optimize a simple function.

There are many ways of implementing this idea, the most prevalent being the strategy introduced by Holland [6, 5, 2] and illustrated in Figure 1. In recent years, "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 different 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 genetic algorithm. Books that describe the theory and practice of genetic algorithms in greater detail include [6, 5, 1, 8, 7, 9].

The genetic algorithm is interesting from a computational standpoint, at least in part, because of the claims that have been made about its effectiveness as a biased sampling algorithm. The classical argument about genetic algorithm performance has three components:

- Independent sampling is provided by large populations that are initialized randomly.
- High-fitness individuals are preserved through selection, and this biases the sampling process towards regions of high fitness.
- Crossover combines partial solutions, called "building blocks," from different strings onto the same string, thus exploiting the parallelism provided by maintaining a population of candidate solutions.

A partial solution is taken to be a hyperplane in the search space of strings, and is called a schema. A central claim about genetic algorithms is that schemas capture important regularities in many search spaces, and that a form of "implicit parallelism" exists because one fitness evaluation of an individual comprised of l bits implicitly gives information about the 2^l schemas, or hyperplanes, of which it is an instance. The Schema Theorem states that the genetic algorithm operations guarantee exponentially increasing samples of the observed best schemas in the next time step. By analogy with the k-armed bandit problem it has been argued that the genetic algorithm uses an optimal sampling strategy [6].

Genetic algorithms in various forms have been applied to many scientific and engineering problems. They have been used in a wide variety of optimization tasks, including numerical optimization and combinatorial optimization problems such as circuit design and job shop scheduling. Genetic algorithms have also been used to evolve computer programs for specific

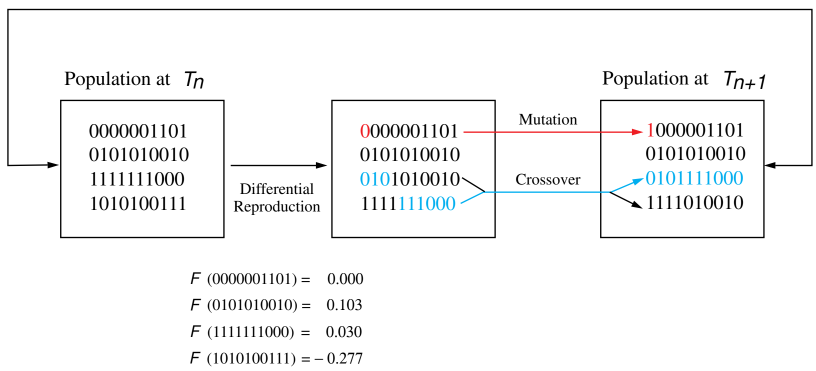

Figure 1: Genetic algorithm overview. A population of four individuals is shown. Each is assigned a fitness value by the function $F(x, y) = yx^2 - x^4$. The fitness values for the population at T_N are shown for each individual (underneath the arrow labeled "Differential Reproduction"). The first five bits are a gray-coded representation of x, and the second five bits are a gray coded representation of y (normalized over the interval $[0, 1)$. On the basis of these fitnesses, the selection phase assigns the first individual (0000001101) one copy, the second (0101010010) two, the third (1111111000) one copy, and the fourth (1010100111) zero copies. After selection, the genetic operators are applied probabilistically; the first individual has its first bit mutated from a 0 to a 1, and crossover combines the last two individuals into two new ones. The resulting population is shown in the box labeled $T_{(N+1)}$.

tasks and to design other computational structures, e.g., cellular automata rules and sorting networks. In machine learning, they have been used to design neural networks, to design rules for rule-based systems, and to design and control robots. Genetic algorithms are known primarily as a problem-solving method, but they can also be used to study and model evolution in various settings, including biological (ecology, immunology, and population genetics), social (such as economies and political systems), and cognitive systems.

Perhaps the most common application of genetic algorithms is multiparameter function optimization. Many problems can be formulated as a search for an optimal value, where the value is a complicated function of some input parameters. In some cases, the parameter settings that lead to the exact greatest (or least) value of the function are of interest. In other cases, the exact optimum is not required, just a near optimum, or even a value that represents a slight improvement over the previously best known value. In these latter cases, genetic algorithms are often an appropriate method for finding good values. The strength of the genetic algorithm lies in its ability to manipulate many parameters simultaneously, and it has been used for hundreds of applications, including aircraft design, tuning parameters for algorithms that detect and track multiple signals in an image, and locating regions of stability in systems of nonlinear difference equations.

Although there are many problems for which the genetic algorithm can evolve a good solution in reasonable time, there are also problems for which it is inappropriate (such as problems in which it is important to find the exact global optimum). It would be useful to have a mathematical characterization of how the genetic algorithm works that is predictive. Research on this aspect of genetic algorithms has not produced definitive answers. The domains for which one is likely to choose an adaptive method such as the genetic algorithm are precisely those about which we typically have little analytical knowledge—they are complex, noisy, or dynamic (changing over time). These characteristics make it virtually impossible to predict with certainty how well a particular algorithm will perform on a particular problem, especially if the algorithm is stochastic, as is the case with the genetic algorithm. In spite of this difficulty, there are fairly extensive theories about how and why genetic algorithms work in idealized settings. These include schema analysis, which views genetic algorithms as a biased sampling process, models developed for population genetics, algebraic models, signal-to-noise analysis, landscape analysis, statistical mechanics models, Markov chains, and analyses based on Probably Approximately Correct (PAC) learning. This work extends and refines schema analysis, and in some cases challenges the claim that recombination through crossover is an important component of genetic algorithm performance.

The idea of using evolution to solve difficult problems and to model natural phenomena is promising. The genetic algorithms described here are one of the first steps in this direction. Necessarily, they have abstracted out much of the richness of biology, and in the future, we can expect a wide variety of evolutionary systems based on the principles of genetic algorithms but less closely tied to these specific mechanisms. More generally, biological mechanisms of all kinds are being incorporated into computational systems, including embryology, viruses, parasites, and immune systems. From an algorithmic perspective, genetic algorithms join a broader class of stochastic methods for solving problems. An important area of future research is to understand carefully how these algorithms relate to one another and which algorithms are best for which problems.

1 Acknowledgments

The author gratefully acknowledges support from the National Science Foundation (grant IRI-9157644), the Office of Naval Research (grant N00014-95-1-0364), ATR Human Information Processing Research Laboratories, and the Santa Fe Institute.

References

[1] L. Davis, editor. *The Genetic Algorithms Handbook.* Van Nostrand Reinhold, 1991.

[2] S. Forrest. Genetic algorithms: Principles of adaptation applied to computation. *Science*, 261:872–878, Aug. 13 1993.

[3] S. Forrest. Genetic algorithms. In A. B. Tucker, editor, *CRC Handbook of Computer Science and Engineering.* CRC Press, Boca Raton, FL, in press.

[4] S. Forrest and M. Mitchell. What makes a problem hard for a genetic algorithm? some anomalous results and their explanation. *Machine Learning*, 13(2/3), 1993.

[5] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison Wesley, Reading, MA, 1989.

[6] J. H. Holland. *Adaptation in Natural and Artificial Systems.* The University of Michigan Press, Ann Arbor, MI, 1975.

[7] J. H. Holland, K. J. Holyoak, R. E. Nisbett, and P. Thagard. *Induction: Processes of Inference, Learning, and Discovery.* MIT Press, 1986.

[8] J. R. Koza. *Genetic Programming.* MIT Press, Cambridge, MA, 1992.

[9] M. Mitchell. *An Introduction to Genetic Algorithm.* MIT Press, Cambridge, MA, 1996.