Preprint
UCRL-JC-137552

Genetic Algorithms

E. Cantú-Paz

This document was submitted to Encyclopedia of Computers and Computer History

February 15, 2000

U.S. Department of Energy

Lawrence
Livermore
National
Laboratory

Genetic Algorithms

Erick Cantú-Paz[1]

(To be published in Encyclopedia of Computers and Computer History, Chicago, IL: Fitzroy Dearborn)

Genetic algorithms are search procedures based on the mechanics of genetics and natural selection. Genetic algorithms work on a population of individuals that represent possible solutions to a problem. Each individual is represented by an artificial chromosome, which can be as simple as a string of zeroes and ones, or as complex as a computer program. The population of individuals may be created entirely at random, or some knowledge about previously known solutions may be used. The individuals are evaluated to determine how well they solve the problem at hand with a fitness function, which is unique to each problem and must be supplied by the user of the algorithms. The individuals with better performance are selected to serve as parents of the next generation. Genetic algorithms create new individuals using simple randomized operators that are similar to sexual recombination (crossover) and mutation in natural organisms. The new solutions are evaluated with the fitness function, and the cycle of selection, recombination, and mutation is repeated until a satisfactory solution is found or a predetermined time limit has elapsed.

The genetic algorithm is controlled by several inputs, such as the size of the population, and the rates that control how often mutation and crossover are used. There is no guarantee that the genetic algorithm will find the best possible solution to the problem. However, a careful manipulation of the inputs and choosing a representation that is adequate to the problem increase the chances of success.

There are many ways to encode a potential solution as a chromosome, and there are many variations of selection methods, crossover, and mutation operators. Some of these choices are better suited to a particular problem than others, and it has been proven that no choice is the best for all problems. The most simple encodings use chromosomes composed of zeroes and ones, but other encodings may be more natural to the problem and may facilitate the search for good solutions. For example, floating-point numbers may be more suitable for a function optimization problem where the parameters to optimize are real numbers, but it may be an awkward match to a problem of finding the shortest route between multiple cities. Likewise, a computer program representation seems a good match to the task of evolving a control program for a soccer-playing robot.

The choice of encoding is related to the operators that are used to produce new solutions from the selected ones. The simplest operator is mutation, and it acts by randomly changing a short piece of the chromosome. For example, when applied to strings of binary digits, it randomly chooses a location in the chromosome of an individual and flips a bit from zero to one or vice versa. Taking a clue from Nature, genetic algorithms do not use mutation very often. The primary mechanism to create new individuals is crossover. In its simplest form, crossover randomly chooses two individuals from those that were selected to be parents, and exchanges segments of their two chromosomes around a single randomly chosen point. The result is two new individuals, each with a segment of chromosome from each parent. Other variants of crossover exchange material around more than one point, and some researchers have experimented with recombining chromosomes from more than two parents. Some of the new solutions will be more fit than the parents, but others will be less fit. The algorithm cannot avoid creating solutions that turn out to be unfit, but selection will eliminate the bad solutions and keep the best.

The selection of the parents can occur in many ways, but all selection methods have the same objective of preserving good individuals and discarding the less fit. Roughly, there are two kinds of selection: hard and soft. Soft selection methods assign to each individual a probability of survival based on their fitness, so that individuals with high fitness are more likely to be selected than individuals with low fitness. The soft selection method then uses the probabilities to select the parents. The hard methods do not involve any probabilities, they simply choose a fixed number of the best solutions available.

The history of genetic algorithms spans several decades, and it can be traced back to the work of biologists in the 1950s and 1960s who used computers to simulate natural genetic systems. The pioneering work of Alexander S. Fraser closely resembles a genetic algorithm with binary strings, crossover and different types of selection, but he was more concerned with modeling and understanding natural evolution than solving specific optimization problems.

---

[1] UCRL-JC-137552. This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

One early example of using evolution to solve problems is due to George E. P. Box. In 1957, he proposed an evolutionary method to improve the productivity of a chemical plant, but his method can be easily adapted to other problems. Box's method did not involve a computer, but rather a committee of plant personnel that would propose variations of the current production setup. After running the plant under each of the production variants for several times and observing their performance, the committee would meet again to identify the best setup and generate new alternatives from it. In Box's evolutionary operation method, there is a single parent solution (the best setup), and the committee acts as the mutation and selection components of the algorithm. Box mentioned the possibility of automating the process.

In 1962, Hans J. Bremermann began solving optimization problems using simulated evolution on a computer. He recognized that biological fitness functions have no known general properties that can be exploited to facilitate their optimization, but the mechanics of natural optimization through selection, recombination, and mutation were known. Although his main interests were in optimization, he realized that his simulations could aid in the understanding of biological evolution. However, he was no purist, and when the methods inspired by natural evolution failed to solve artificial problems, he devised new evolution schemes with no biological counterpart.

The work of John H. Holland is concerned with complex adaptive systems, which are composed of simple rules that collectively exhibit complex behavior. Genetic algorithms are a perfect example of such a system, and Holland focused in trying to understand how they work. In the late 1960s and early 1970s, Holland recognized that genetic algorithms solve difficult optimization problems by implicitly gathering information about the problem and at the same time exploiting this information to solve it. Based on this insight, Holland developed a theoretical explanation for the performance of genetic algorithms that is the foundation of the work of many researchers. His 1975 book, "Adaptation in Natural and Artificial Systems", is widely recognized as one of the most important milestones in the history of genetic algorithms.

Independent research groups in Europe and America created evolutionary methods similar to genetic algorithms. In 1964, Ingo Rechenberg and Hans-Paul Schwefel at the University of Dortmund in Germany developed a method called evolution strategies to automate the design of objects with desired aerodynamic characteristics. Also in the early 1960s, Lawrence J. Fogel, Alvin J. Owens and Michael J. Walsh devised a method to simulate intelligent behavior based on evolving theoretical machines to solve particular tasks. The early versions of these algorithms differ from each other and from genetic algorithms in the way they represent the chromosomes, in how they perform selection, and in how they create new individuals. But, despite their differences, all evolutionary algorithms have many things in common. They all have a population of individuals that represent candidate solutions, a randomized method to create new solutions from the old ones, and a method of selecting the solutions for the future generations. As the interactions between the different research groups intensified in the late 1980s and early 1990s, the differences between their algorithms became less pronounced.

For quite some time, evolutionary algorithms remained unknown to the majority of computer scientists and engineers. David E. Goldberg greatly helped introduced genetic algorithms into mainstream engineering and computer science by writing the first textbook on the subject in 1989. Goldberg's work is mainly concerned with the characterization of problems that are difficult for genetic algorithms and with designing robust problem solvers that can quickly and reliably find good solutions to hard problems.

In the 1990s, John R. Koza promoted the use of genetic algorithms to evolve computer programs. He has shown that, in some cases, artificial evolution can generate solutions that are competitive with human intelligence. His experiments have resulted in the rediscovery of previously patented solutions to difficult problems of design of electric circuits. He has also obtained human-competitive results in evolving robot controllers and in the characterization of protein molecules.

Today, there are many practical applications of genetic and other evolutionary algorithms in various fields of engineering, science, finance, and management. We can expect that as evolutionary methods become more widely known they will be used to solve problems in many more areas. There are still many possibilities to improve the algorithms, and many important questions about how they operate remain unanswered, so the history of these simple algorithms is still being written.

Further reading

Fogel, D. B. (ed.) (1998). *Evolutionary Computation: The Fossil Record.* New York, NY: IEEE Press.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems.* Ann Arbor, MI: University of Michigan Press. (A second edition was published by The MIT press in 1992.)

Koza, J. R. (1992) *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: The MIT Press.

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

Related Topics

Artificial intelligence, artificial life, binary system, heuristic

Word count: 1607