evolving Inventions

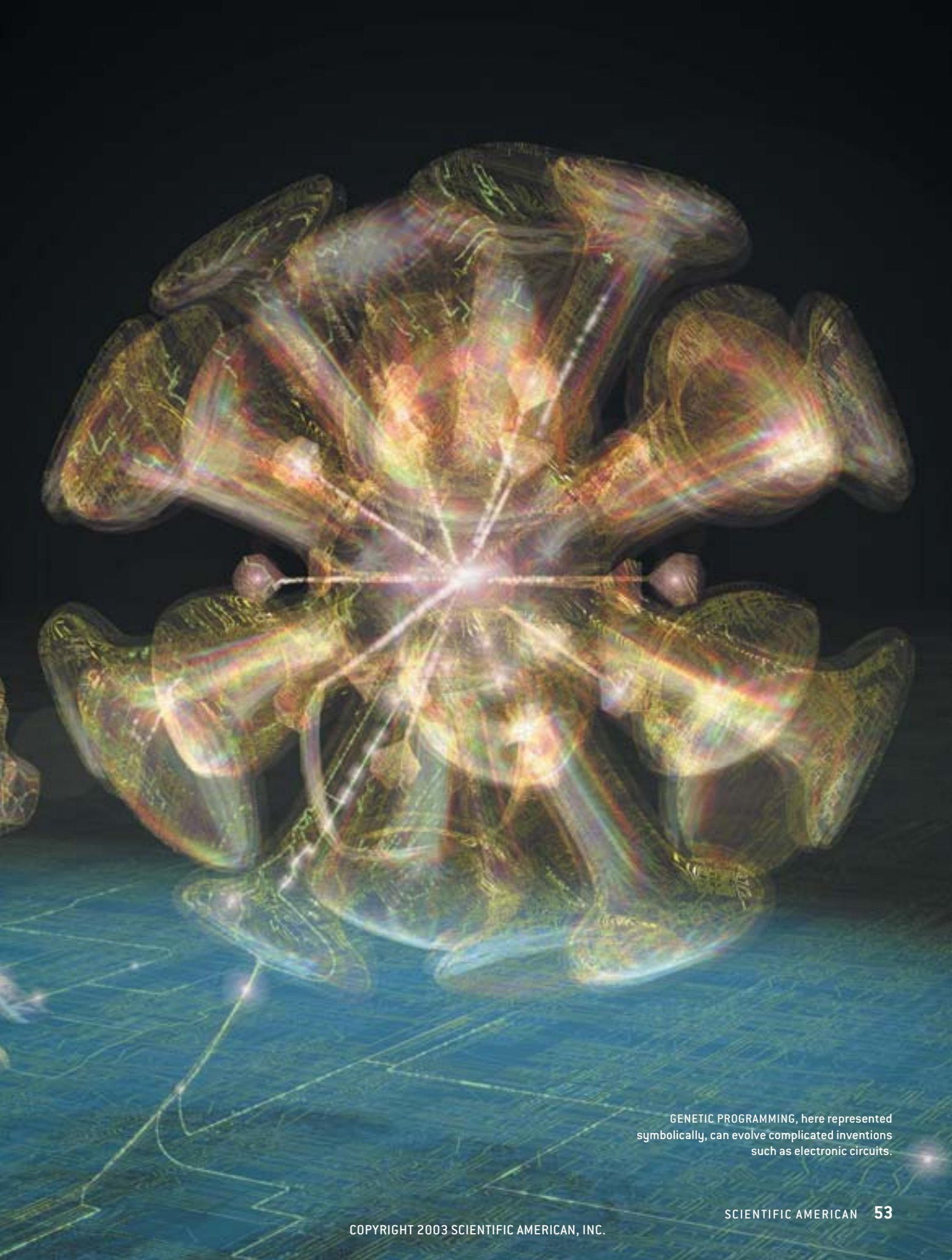
By John R. Koza, Martin A. Keane and Matthew J. Streeter

Evolution is an immensely powerful creative process. From the intricate biochemistry of individual cells to the elaborate structure of the human brain, it has produced wonders of unimaginable complexity. Evolution achieves these feats with a few simple processes—mutation, sexual recombination and natural selection—which it iterates for many generations. Now computer programmers are harnessing software versions of these same processes to achieve machine intelligence. Called genetic programming, this technique has designed computer programs and electronic circuits that perform specified functions.

In the field of electronics, genetic programming has duplicated 15 previously patented inventions, including several that were hailed as seminal in their respective fields when they were first announced [*see box on page 57*]. Six of these 15 existing inventions were patented after January 2000 by major research institutions, which indicates that they represent current frontiers of research in domains of scientific and practical importance. Some of the automatically produced inventions infringe squarely on the exact claims of the previously patented inventions. Others represent new inventions by duplicating the functionality of the earlier device in a novel way. One of these inventions is

Illustrations by Bryan Christie Design

Computer programs that function via Darwinian evolution are creating inventions that are novel and useful enough to be patented



GENETIC PROGRAMMING, here represented symbolically, can evolve complicated inventions such as electronic circuits.

Genetic programming begins with a primordial ooze of randomly generated "organisms."

a clear improvement over its predecessor.

Genetic programming has also classified protein sequences and produced human-competitive results in a variety of areas, such as the design of antennas, mathematical algorithms and general-purpose controllers [*see box on page 59*]. We have recently filed for a patent for a genetically evolved general-purpose controller that is superior to mathematically derived controllers commonly used in industry.

The first practical commercial area for genetic programming will probably be design. In essence, design is what engineers do eight hours a day and is what evolution does. Design is especially well suited to genetic programming because it presents tough problems for which people seek solutions that are very good but not mathematically perfect. Generally there are complex trade-offs between competing considerations, and the best balance among the various factors is difficult to foresee. Finally, design usually involves discovering topological arrangements of things (as opposed to merely optimizing a set of numbers), a task that genetic programming is very good at.

Human engineers tend to look at problems in particular ways, often based on ideal mathematical models. Genetic programming offers the advantage of not being channeled down narrow paths of thinking. Evolution does not know anything about the underlying math; it simply tries to produce a sequence of improved results. Thus, we frequently see creative things come out of the evolutionary process that would never occur to human designers.

Out of the Primordial Ooze

WHATEVER THE FIELD of endeavor, genetic programming begins with a primordial ooze of randomly generated trial "organisms" and a high-level description of what function the organisms are meant to accomplish—the criteria for scoring their fitness. As an example, consider a case in which the organisms are elementary mathematical functions and we are endeavoring to find a function whose graph matches a given curve. The organisms in this case are composed of numerical constants and primitive operations such as addition, subtraction, multiplication and division. The fitness of a function is determined by how closely its graph follows the target curve.

The genetic program evaluates the fitness of each mathematical function in the population. The initial, randomly created functions will, of course, match the target curve quite poorly, but some will be better than others. The genetic program tends to discard the worst functions in the population, and it applies genetic operations to the surviving functions to create offspring. The most important genetic operation is sexual reproduction, or crossover, which mates pairs of the better organisms to sire offspring composed of genetic material from the two parents [*see top illustration on opposite page*]. For instance, mating the functions $(a + 1) - 2$ and $1 + (a \times a)$ might result in the $(a + 1)$ part of the first function substituting for one a of the second function, producing offspring $1 + ((a + 1) \times a)$. Recombining the traits of two relatively fit organisms in this fashion sometimes produces superior offspring.

In addition to sexual reproduction, genetic programming copies about 9 percent of the fittest individuals unaltered into the next generation, which generally ensures that the best organisms in each generation are at least as fit as those of the previous generation. Finally, about 1 percent of the programs undergo mutation—for instance, $a + 2$ might mutate into $(3 \times a) + 2$—in the hope that a random modification of a relatively fit program will lead to improvement.

These genetic operations progressively produce an improved population of mathematical functions. The exploitation of small differences in fitness yields major improvements over many generations in much the same way that a small interest rate yields large growth when compounded over decades.

One can visualize the evolutionary process as being a search in the space of all possible organisms. The crossover operation conducts the most creative kind of search, which is why we use it to produce around 90 percent of the offspring in each generation [*see bottom illustration on opposite page*]. Mutation, in contrast, tends to conduct a local search for advantage near the existing good individuals. We believe that too great a mutation rate results in less efficient evolution except in the case of particularly simple problems.

A more sophisticated example than a mathematical function is the evolution of

Overview/*Darwinian Invention*

- Genetic programming harnesses a computerized version of evolution to create new inventions. Starting from thousands of randomly generated test objects, the method selects the better individuals and applies processes such as mutation and sexual recombination to generate successive generations.
- Over the course of dozens of generations, the population of individuals gradually fulfills the target criteria to a greater degree. At the end of the run, the best individual is harvested as the solution to the posed problem.
- In electronics, the technique has reproduced patented inventions, some of which lie at the forefront of current research and development. Other inventions include antennas, computer algorithms for recognizing proteins, and general-purpose controllers. Some of these computer-evolved inventions should themselves be patentable.
- By the end of the decade, we envision that increased computer power will enable genetic programming to be used as a routine desktop invention machine competing on equal terms with human inventors.

a computer program, such as one employing iterations and memory for classifying protein sequences. In this case, genetic programming can carry out analogues of the biological processes of gene duplication and gene deletion, to create or delete subroutines, iterations, loops and recursions in the evolving population of programs. The evolutionary process itself determines the character and content of the computer program needed to solve the problem.

A low-pass filter circuit provides a good illustration of how genetic programming designs analog electronic circuits. A low-pass filter is used in a hi-fi system to send only the lowest frequencies to the woofer speaker. To create a low-pass filter by using genetic programming, the human user specifies which components are available for building the circuit (say, resistors, capacitors and inductors) and defines the fitness of each candidate circuit to be the degree to which it passes frequencies up to 1,000 hertz at full power while filtering out all higher frequencies.

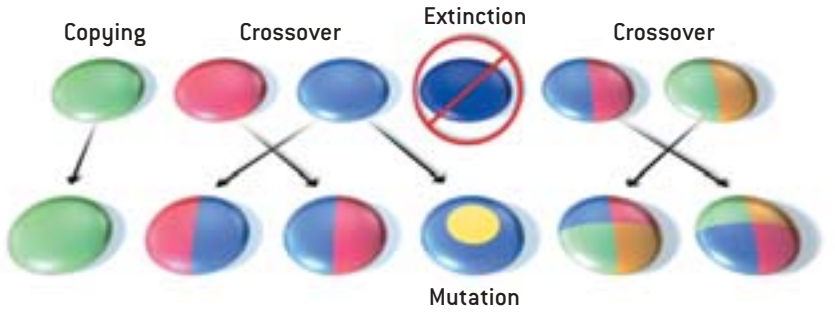
The circuits are generated in a way that borrows mechanisms from developmental biology. Each circuit begins as an elementary "embryo" consisting of a single wire running from the input to the output. The embryonic circuit grows by progressive application of circuit-constructing functions. Some of the circuit-constructing functions insert particular components. Others modify the pattern of connections between components: they might duplicate an existing wire or component in series or parallel, or they might create a connection from a particular point to a power supply, the ground or a distant point in the growing circuit. This developmental process yields both the circuit topology and the numerical component sizes. The system automatically synthesizes circuits without using any advanced know-how from the field of electrical engineering concerning circuit synthesis.

Most of the initial population of rudimentary circuits generated randomly in this way will behave nothing like a low-pass filter. A few, however, will contain an inductor between the circuit's input and output, thereby slightly imped-

UNNATURAL SELECTION

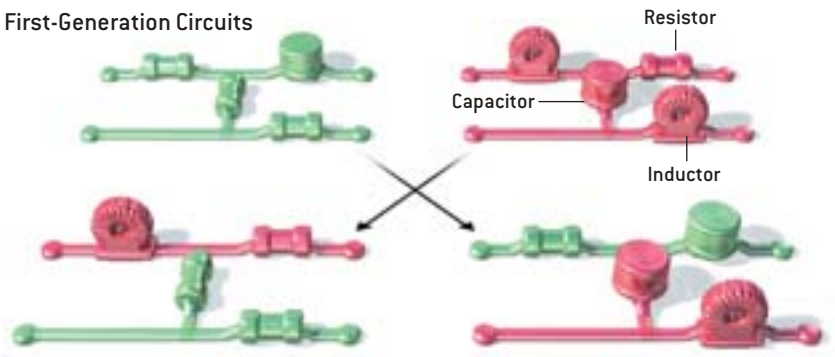
Evolutionary Processes

THREE PROCESSES propagate "organisms" (*represented here by colored disks*) from one generation to the next in a genetic programming run. Some of the better organisms are copied unaltered. Others are paired up for sexual reproduction, or crossover, in which parts are swapped between the organisms to produce offspring. A small percentage are changed randomly by mutation. Organisms not chosen for propagation become extinct. The crossover operation is applied more frequently than copying and mutation because of its ability to bring together new combinations of favorable properties in individual organisms.



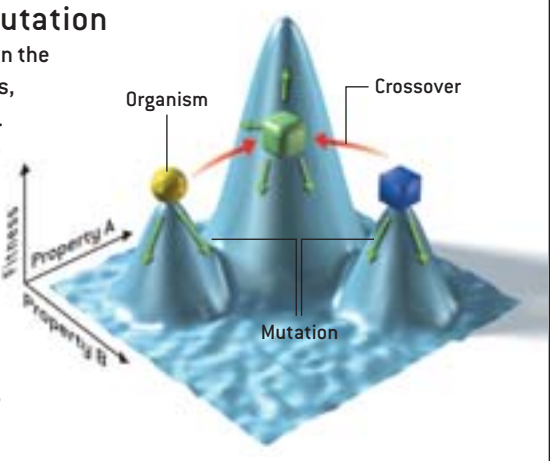
Crossover of Electronics

ACTING ON electronic circuits, the crossover operation takes two circuits and swaps some of their components, producing two new circuits.

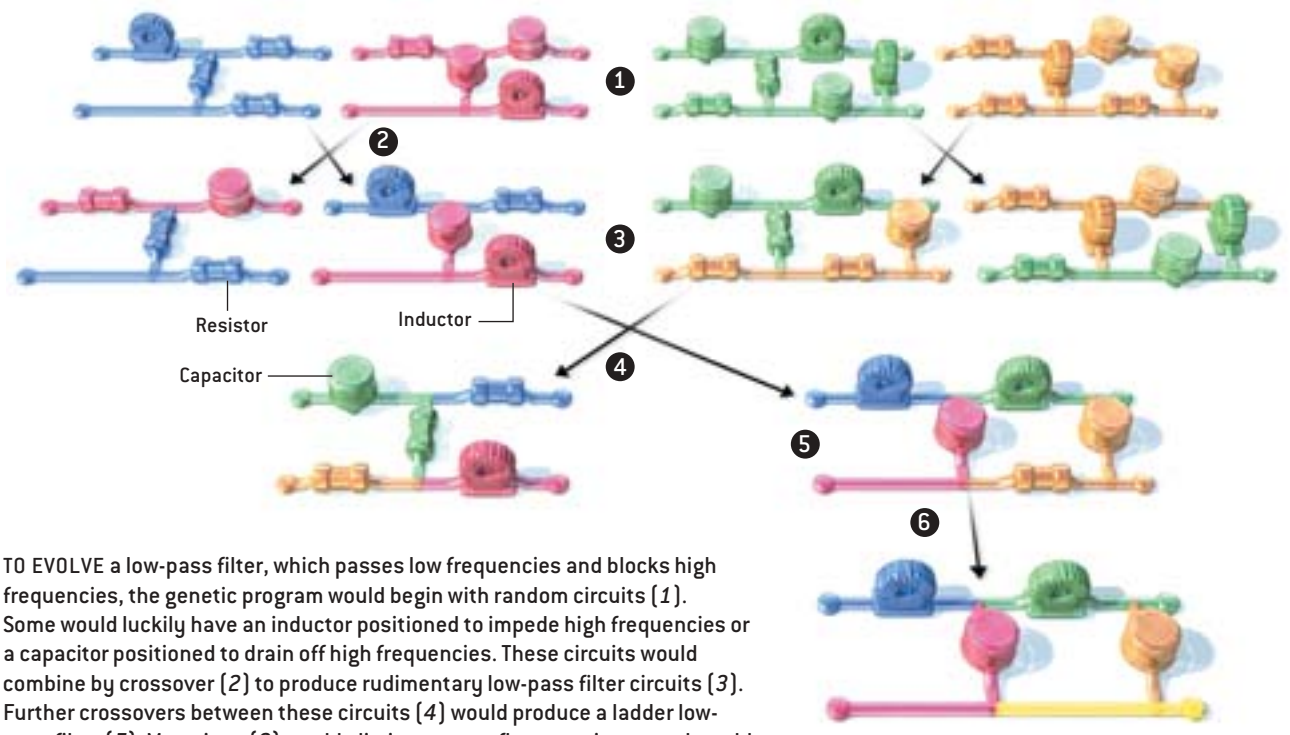


Crossover versus Mutation

EVOLUTION ACTS like a search in the space of all possible organisms, represented here by the plane. Crossover searches this space creatively, occasionally combining disparate good features, leaping to a new region of organism space where much fitter individuals reside (*red arrows*). Mutation, in contrast, tends to find the best organism that is "nearby" (*green arrows*).



EVOLVING A LOW-PASS FILTER



TO EVOLVE a low-pass filter, which passes low frequencies and blocks high frequencies, the genetic program would begin with random circuits (*1*). Some would luckily have an inductor positioned to impede high frequencies or a capacitor positioned to drain off high frequencies. These circuits would combine by crossover (*2*) to produce rudimentary low-pass filter circuits (*3*). Further crossovers between these circuits (*4*) would produce a ladder low-pass filter (*5*). Mutations (*6*) would eliminate superfluous resistors and would fine-tune the values of the components.

ing higher frequencies. Others will have a capacitor running from the input to the ground, thereby slightly draining the power of higher frequencies [*see illustration above*]. Such circuits will be selected to mate more frequently than others, and eventually later generations will contain offspring incorporating both features. The crossover and mutation operations acting on numerical expressions will adjust component values so that the cutoff frequency approaches the desired 1,000 Hz. Other crossovers and mutations will delete resistors that dissipate power. Additional crossovers will double or triple the inductor-capacitor combination, yielding the ladder structure patented in 1917 by George A. Campbell of AT&T.

Other devices are designed with similar combinations of evolutionary and developmental processes. Antennas, for instance, are automatically designed with a "turtle" that deposits (or does not deposit) metal on a plane as it moves and turns under the control of various operations (similar to those in the LOGO programming language).

The primitive ingredients used to create controllers automatically consist of differentiators, integrators and amplifiers. An example of a basic controller is a cruise control on a car, which must reduce fuel intake if the speed rises too high or increase it if the speed falls too low. A good controller will allow for the delayed response to fuel changes and will continuously monitor how the speed is varying to avoid excessive overshooting of the target speed. Of great importance are general-purpose controllers, which can be customized to a variety of specific tasks—such as the control of a home furnace, manufacturing processes in factories or the reading arm of a computer's disk storage device. Small improvements in the "tuning rules" used in customizing a controller can result in large economic savings.

A commonplace controller is the PID controller invented in 1939 by Albert Callender and Allan Stevenson of Imperial

THE AUTHORS

JOHN R. KOZA, MARTIN A. KEANE and *MATTHEW J. STREETER* work closely with one another studying genetic programming using a home-built, 1,000-Pentium parallel computer. Koza received his Ph.D. in computer science from the University of Michigan in 1972. He co-founded Scientific Games, Inc., in Atlanta in 1973, where he co-invented the rub-off instant lottery ticket used by state lotteries. In 1987 Koza invented genetic programming. He is currently consulting professor in the Stanford Biomedical Informatics program in the department of medicine and consulting professor in the university's department of electrical engineering. Keane received a Ph.D. in mathematics from Northwestern University in 1969. From 1976 to 1986 he was vice president for engineering at Bally Manufacturing Corporation in Chicago. He is now chief scientist at Econometrics, Inc., also in Chicago. Streeter received a master's degree from Worcester Polytechnic Institute in 2001. His primary research interest is applying genetic programming to problems of real-world scientific or practical importance. He works at Genetic Programming, Inc., in Los Altos, Calif., as a systems programmer and researcher.

The first patent for an invention created by genetic programming may soon be granted.

Chemical Limited in Northwich, England. PID controllers (the initials stand for the controller's *p*roportional, *i*ntegrative and *d*erivative parts) are used in myriad applications. Our genetic programs have evolved two distinct improvements in this field. First, they have developed a new set of tuning rules for PID controllers. A relatively simple and effective set of PID tuning rules has been in general use since 1942 and was improved on in 1995; our rules outperform the 1995 rules. Second, we evolved three new controller circuit topologies that also outperform PID controllers that use the old tuning rules. We have filed a patent application that covers both the new rules and the new controller topologies. If (as we expect) the patent is granted, we believe that it will be the first one granted for an invention created by genetic programming.

Evolvable Hardware

DURING THE EVOLUTIONARY process, we must efficiently evaluate the fitness of thousands or millions of offspring in each generation. For electronic circuits, we usually use standard circuit-simulator software to predict the behavior of each circuit in the population. In an important emerging area of technology called evolvable hardware, however, microchips can be instantaneously configured to physically implement each circuit of a genetic programming run.

Known as rapidly reconfigurable field-programmable gate arrays, these chips consist of thousands of identical cells, each of which can perform numerous different logical functions, depending on how it is programmed. Sets of memory bits in the "basement" of the chip customize each cell so that it performs a particular logical function. Other configuration bits program interconnection routes on the chip, permitting many different ways of connecting the cells to one another and to the chip's input and output pins. The "personality" of the chip (its logical functions and interconnections) can be changed dynamically in nanoseconds merely by changing its configuration bits.

These rapidly reconfigurable chips are

Patented Inventions Re-created by Computer

TO DATE, genetic programming has re-created 15 inventions that were previously patented by the inventors listed here.

INVENTION	YEAR	INVENTOR	INSTITUTION
LADDER FILTER	1917	George A. Campbell	AT&T, *New York City*
CROSSOVER FILTER	1925	Otto Julius Zobel	AT&T
NEGATIVE FEEDBACK AMPLIFIER	1927	Harold S. Black	AT&T
ELLIPTIC FILTER	1934–36	Wilhelm Cauer	University of Göttingen, *Germany*
PID (proportional, integrative and derivative) CONTROLLER	1939	Albert Callender and Allan Stevenson	Imperial Chemical Limited, *Northwich, England*
SECOND-DERIVATIVE CONTROLLER	1942	Harry Jones	Brown Instrument Company, *Philadelphia*
DARLINGTON EMITTER-FOLLOWER SECTION	1953	Sidney Darlington	Bell Telephone Laboratories, *New York City*
PHILBRICK CIRCUIT	1956	George A. Philbrick	George A. Philbrick Researches, *Boston*
SORTING NETWORK	1962	Daniel G. O'Connor and Raymond J. Nelson	General Precision, *Los Angeles*
MIXED ANALOG-DIGITAL INTEGRATED CIRCUIT for producing variable capacitance	2000	Turgut Sefket Aytur	Lucent Technologies, *Murray Hill, N.J.*
VOLTAGE-CURRENT CONVERTER	2000	Akira Ikeuchi and Naoshi Tokuda	Mitsumi Electric, *Tokyo*
CUBIC FUNCTION GENERATOR	2000	Stefano Cipriani and Anthony A. Takeshian	Conexant Systems, *Newport Beach, Calif.*
LOW-VOLTAGE, HIGH-CURRENT TRANSISTOR CIRCUIT for testing a voltage source	2001	Timothy Daun-Lindberg and Michael Miller	IBM, *Armonk, N.Y.*
LOW-VOLTAGE BALUN CIRCUIT	2001	Sang Gug Lee	Information and Communications University, *Taejon, Korea*
TUNABLE INTEGRATED ACTIVE FILTER	2001	Robert Irvine and Bernd Kolb	Infineon Technologies, *Munich, Germany*

sold by about a dozen companies, but they are primarily of use for digital circuits. Commercially available analog chips are extremely limited in their abilities. We used a reconfigurable digital chip to create a sorting network with fewer steps than the originally patented version.

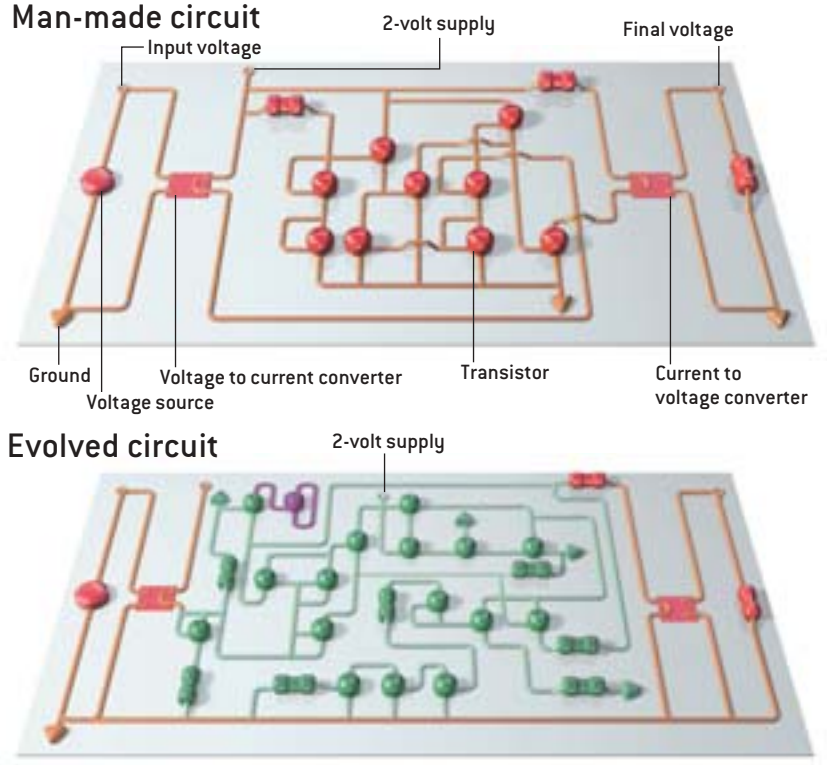
Run Times

NATURAL EVOLUTION has had billions of years of "run time" to produce its wonders. Artificial genetic programming would not be of much use if it took as long. A genetic programming run typically spawns a population of tens or hundreds of thousands of individuals that evolve over dozens or hundreds of generations. A weeklong run on a laptop computer is sufficient to produce half of the human-competitive results listed in the box on the preceding page; however, all six of the inventions patented after 2000 required more horsepower than that.

Evolution in nature thrives when organisms are distributed in semi-isolated subpopulations. The same seems to be true of genetic programming run on a loosely connected network of computers. Each computer can perform the time-consuming step of evaluating the fitness of individuals in its subpopulation. Then, at the end of each generation, a small percentage of individuals (selected based on fitness) migrates to adjacent computers in the network so that each semi-isolated subpopulation gets the benefit of the evolutionary improvement that has occurred elsewhere.

HUMAN VERSUS COMPUTER

THE TWO CIRCUITS shown below are both cubic signal generators. The upper circuit is a patented circuit designed by a human; the green and purple parts of the lower circuit were evolved by genetic programming (the other parts are standard input and output stages). The evolved circuit performs with better accuracy than the human-designed one, but *how* it functions is not understood. The evolved circuit is clearly more complicated but also contains redundant parts, such as the purple transistor, that contribute nothing to its functioning.



We have built a Beowulf-style computer cluster consisting of 1,000 somewhat outdated 350-megahertz Pentium computers [see "The Do-It-Yourself Supercomputer," by William W. Hargrove, Forrest M. Hoffman and Thomas Sterling; SCIENTIFIC AMERICAN, August 2001]. For our most time-consuming problems, evaluation of the fitness of a single candidate individual takes about a minute of computer time. A run involving a population of 100,000 individuals for 100 generations can be completed in about seven days on our cluster.

The 1,000 computers together perform about 350 billion cycles a second. Although this amount of computer time may, at first blush, sound like a lot, it pales in comparison to the amount of computation performed by the trillion cells of the human brain (each of which is thought to have about 10,000 connections and operate at a rate of 1,000 operations a second).

We expect that 50-gigahertz computers (performing 50 billion cycles a second) will be commonly available toward the end of this decade, putting the power to evolve patent-worthy inventions using genetic programming in the hands of anyone owning a moderately priced desktop workstation. We envision that genetic programming will be regularly used as an invention machine.

Passing an Intelligence Test

GENETIC PROGRAMMING is now routinely reproducing human inventions, just half a century after computer pioneer Alan M. Turing predicted that human-competitive machine intelligence would be achieved in about 50 years. During those 50 years, the two main academically fashionable approaches taken by researchers striving to vindicate Turing's prediction have used logical deduction or databases containing accumulated human knowledge and expertise (so-called expert systems). Those two approaches roughly correspond to two broad methods outlined by Turing in 1950. The first (not surprising in light of Turing's work in the 1930s on the logical foundations of computing) was the construction of programs designed to an-

More Human-Competitive Creations

AS WELL AS re-creating patented inventions, genetic programming has generated these results that a human would be proud of.

SOCCER-PLAYING PROGRAM that ranked in the middle of the field of 34 human-written programs in the RoboCup 1998 competition
REAL-TIME ANALOG CIRCUIT for time-optimal control of a robot
FOUR DIFFERENT ALGORITHMS for identifying transmembrane segments of proteins
DERIVING MOTIFS (highly conserved sequences of amino acids) to identify certain families of proteins
ALGORITHMS FOR QUANTUM COMPUTERS that in some cases solve problems better than any previously published result
NAND CIRCUIT for carrying out the NOT AND logical operation on two inputs
ANALOG COMPUTATIONAL CIRCUITS for the square, cube, square root, cube root, logarithm and Gaussian functions
DIGITAL-TO-ANALOG CONVERTER CIRCUIT
ANALOG-TO-DIGITAL CONVERTER CIRCUIT

alyze situations and problems logically and to respond accordingly. The second, which Turing called a cultural search, applied knowledge and expertise gathered from experts.

The goal of artificial intelligence and machine learning is to get computers to solve problems from a high-level statement of what needs to be done. Genetic programming is delivering human-competitive machine intelligence with a minimum of human involvement for each new problem and without using either logical deduction or a database of human knowledge.

Turing also proposed a famous test for machine intelligence. In one widely used restatement of the Turing test, a judge receives messages "over a wall" and tries to decide whether the messages came from a human or a machine. We do not claim that genetic programming has achieved the kind of general imitation of human cognition associated with the Turing test. But it *has* passed a test of creativity and ingenuity that only a relatively small number of humans pass. The U.S. patent office has been administering this test for more than 200 years.

The patent office receives written descriptions of inventions and then judges whether they are unobvious to a person having ordinary skill in the relevant field. Whenever an automated method duplicates a previously patented human-designed invention, the automated method has passed the patent office's intelligence test. The fact that the original, human-designed version satisfied the patent office's criteria of patent-worthiness means that the computer-created duplicate would also have satisfied the patent office.

This intelligence test does not deal with inconsequential chitchat or the playing of a game. When an institution or individual allocates time and money to invent something and to embark on the time-consuming and expensive process of obtaining a patent, it has made a judgment that the work is of scientific or practical importance. Moreover, the patent office requires that the proposed invention be useful. Patented inventions represent nontrivial work by exceptionally creative humans.

Although some people may be surprised that routine human-competitive machine intelligence has been achieved with a nondeterministic method and without resorting to either logic or knowledge, Alan Turing would not be. In his 1950 paper, Turing also identified this third approach to machine intelligence: "the genetical or evolutionary search by which a combination of genes is looked for, the criterion being the survival value."

Turing did not specify how to conduct a "genetical or evolutionary search" to achieve machine intelligence, but he did point out that:

> We cannot expect to find a good child-machine at the first attempt. One must experiment with teaching one such machine and see how well it learns. One can then try another and see if it is better or worse. There is an obvious connection between this process and evolution, by the identifications
>
> Structure of the child machine
> = Hereditary material
> Changes of the child machine
> = Mutations
> Natural selection
> = Judgment of the experimenter

Genetic programming has in many ways fulfilled the promise of Turing's third way to achieve machine intelligence. SA

MORE TO EXPLORE

Computing Machinery and Intelligence. Alan M. Turing in *Mind,* Vol. 59, No. 236, pages 433–460; October 1950. Available at **www.abelard.org/turpap/turpap.htm** by permission of Oxford University Press.

Genetic Programming: On the Programming of Computers by Means of Natural Selection. John R. Koza. MIT Press, 1992.

Genetic Programming: The Movie. John R. Koza and James P. Rice. MIT Press, 1992.

Genetic Programming III: Darwinian Invention and Problem Solving. John R. Koza, Forrest H Bennett III, David Andre and Martin A. Keane. Morgan Kaufmann, 1999.

Genetic Programming III: Videotape: Human-Competitive Machine Intelligence. John R. Koza, Forrest H Bennett III, David Andre, Martin A. Keane and Scott Brave. Morgan Kaufmann, 1999.

Genetic Programming IV: Routine Human-Competitive Machine Intelligence. John R. Koza, Martin A. Keane, Matthew J. Streeter, William Mydlowec, Jessen Yu and Guido Lanza. Kluwer Academic Publishers (in press).

More information can be obtained from Genetic Programming, Inc. (**www.genetic-programming.com**), and the Genetic Programming Conference organization (**www.genetic-programming.org**)