As a high-school student in the 1950s, John Koza yearned for a personal computer. That was a tall order back then, as mass-produced data processors such as the IBM 704 were mainframes several times the size of his bedroom. So the cocksure young man went rummaging for broken jukeboxes and pinball machines, repurposing relays and switches and lightbulbs to make a computer of his own design.

Within certain parameters, his computer was a success, flawlessly reckoning the day of the week whenever he dialed in a calendar date, but the hardwiring made it useless for anything else. Koza's first invention was not about to supplant IBM, but the mothballed gizmo remains in his basement to this day, a reminder to himself that the intelligence of a machine is a matter of adaptability as much as accuracy.

Over the past several decades, Koza has internalized that lesson as deeply as any computer scientist alive and, arguably, made more of the insight than any coder in history. Now 62 and an adjunct professor at Stanford University, Koza is the inventor of genetic programming, a revolutionary approach to artificial intelligence (AI) capable of solving complex engineering problems with virtually no human guidance. Koza's 1,000 networked computers don't just follow a preordained routine. They create, growing new and unexpected designs out of the most basic code. They are computers that innovate, that find solutions not only equal to but better than the best work of expert humans. His "invention machine," as he likes to call it, has even earned a U.S. patent for developing a system to make factories more efficient, one of the first intellectual-property protections ever granted to a nonhuman designer.

Yet as impressive as these creations may be, none are half as significant as the machine's method: Darwinian evolution, the process of natural selection. Over and over, bits of computer code are, essentially, procreating. And over the course of hundreds or thousands of generations, that code evolves into offspring so well-adapted for its designated job that it is demonstrably superior to anything we can imagine. The age of creative machines has arrived. And its prophet is John Koza.

Playing the Lottery

Computer science was still a brand-new discipline in the early 1960s, when Koza went to the University of Michigan. He was the second person anywhere to earn a bachelor's degree in the field. "I was interested in computers, so I studied computer science," he explains with characteristic bluntness. "Why do other people go into medicine or become policemen?" He earned his Ph.D. in December 1972, six months away from an academic job opportunity.

Industry, on the other hand, was eager for computer-science expertise. While in school, Koza had worked part-time for a supermarket rub-off game manufacturer called J&H International, calculating probabilities to keep game layouts unpredictable. A full-time position there was as good as certain. But the week of his graduation, the company shut its doors permanently.

Like many successful innovators, Koza combines unusual competence in his work with supreme confidence in himself. Rather than taking the bankruptcy as a sign that rub-off games were dead, he decided that scratch cards were the future of yet another moribund business: state lotteries. At the time, lotteries were weekly raffles, typically with six-digit tickets. A state might sell $1 million worth of tickets a week. Koza believed that, with a more interesting game, especially one offering instant gratification, he could sell more. He opened his own business with another former J&H employee and took a one-year gamble. By the end of 1974, they landed a contract with the state of Massachusetts for 25 million rub-off games.

The success of the instant-win lottery was, in a word, instantaneous-$2.7 million worth of tickets sold in the first week. "Our basic business was tripling lottery sales," Koza says. By 1982, dozens of states that didn't have a lottery had adopted one, and his company, Scientific Games, was diligently supplying most of them. Koza had invented a machine for printing money. He sold the company to Bally Manufacturing and ran the business under contract until 1987. At which point he found himself very rich but, for the second time in his life, jobless.

Darwin in the Machine

John Koza hovers over a computer terminal in a cramped office a mile from the building where he keeps his invention machine. Seated at the terminal is a clean-cut researcher named Lee Jones, one of Koza's two employees. His other employee, Sameer Al-Sakran, leans over a second terminal, stroking his facial stubble.

Jones is reviewing one of the invention machine's latest accomplishments, which Koza is preparing to present at the annual Genetic and Evolutionary Computation Conference, familiarly called GECCO. In this instance, the machine has created a complex lens system that outperforms a wide-field eyepiece for telescopes and binoculars patented just six years ago by lens designers Noboru Koizumi and Naomi Watanabe-and which does so, moreover, without infringing on the Koizumi-Watanabe patent.

Jones calls up an optical simulator known as KOJAC. From a prescription (which numerically describes the curvature, thickness and glass type of lens components), KOJAC predicts how the compound lens will function in the real world. The numerous variables make the effect of simple changes difficult to predict. As a result, lens designers are a creative bunch, who depend as heavily on intuition as on knowledge.

What Koza has done is to automate the creative process. To begin, the invention machine randomly generates 75,000 prescriptions. It then analyzes them in KOJAC, which assigns each a fitness rating based on how close it comes to a desired set of specifications-in this case, a wide field of view with minimal distortion. None of the 75,000 members of the first generation will be usable wide-field telescopic eyepieces. But a few of these primitive systems will be marginally effective at focusing a wide field of view, and a couple others might slightly reduce distotrtion in one way or another.

From there, it's Darwinism 101. The invention machine mates some systems together, redistributing characteristics from two parent lens systems into their offspring. Others it mutates, randomly altering a single detail. Other lenses pass on to the next generation unchanged. And then there are the cruel necessities of natural selection: The machine expels most lenses with low fitness ratings from the population, kills them off so their genetic material won't contaminate the others.

Koza asks Jones to pull up the stats on the wide-field telescopic eyepiece. Amid a rush of figures, he reads off the number "295." That's how many generations it took for genetic programming to engineer around the Koizumi-Watanabe patent. In fact, the invention machine's lens is *better* than the Koizumi-Watanabe system: Because it keeps breeding until all design specs are met, often some performance requirements are exceeded by the end of the run. The final field-of-view for Koza's eyepiece is a remarkable 10 degrees higher than the 55 degrees achieved by Koizumi and Watanabe.

Jones swiftly rotates through several other recent inventions, all generated using the same technique as the lens system. There are logic circuits and amplifiers and filters, some of them suitable for the challenging low-power requirements of cellphones and laptops. Each took between one day and one month to evolve, generating an electricity bill of more than $3,000 a month.

**
The Breeding Grounds**

Like every engineering breakthrough, genetic programming did not emerge fully formed from the ether. Rather it grew out of two promising yet unfulfilled lines of research in computer science: genetic algorithms and artificial intelligence.

Koza's thesis adviser at the University of Michigan was John Holland, the man widely regarded as the father of genetic algorithms. While Holland was a grad student studying mathematics at Michigan in the 1950s, he'd happened upon a book called *The Genetical Theory of Natural Selection*, written by English biologist Ronald Fisher in 1930. The book laid out, in strict mathematical terms, the basic mechanism of variation in plants and animals. "I thought I'd try to figure out a program that did that," Holland recalls. He envisioned a system that, through small, incremental improvements, would breed good code the way a farmer breeds good corn-an early forerunner to genetic programming in the sense that, say, Mendelian inheritance was the first step toward understanding Darwinian evolution.

Holland and his student David Goldberg implemented the idea in 1980. Goldberg had studied civil engineering before working on his doctorate and was interested in the practical problem of how computers could be used to optimize the capacity of gas pipelines. They began by creating a rough model of an efficient layout for the pipeline. The software then made small random changes to the system-alternately varying the pressure, flow rate and pumping schedule-and simulated the gas flow anew following each mutation cycle. The computer retained any alteration that improved performance, even by the slightest bit, and discarded all those that didn't. Over 20 or 30 generations, the system evolved, by almost imperceptible steps, to become markedly better than it was at the start.

The power of genetic algorithms increased in step with the power of the processors they ran on. By the mid-1980s, Holland's process had spawned a small cottage industry, complete with dedicated academic conferences and myriad industrial applications.

Artificial intelligence, on the other hand, was decidedly less practical then. Its goal was (and largely remains) to model human cognitive functions, such as language use and pattern recognition, in computer systems-to make machines think. AI researchers were also hugely optimistic. (One conference topic, seriously debated, was "Should AI run for president?")

Koza had just left the lottery business. He was interested in the commercial potential of genetic algorithms and thought, given his financial success and the strong economy, that he would become a venture capitalist. He read Holland's book on genetic algorithms, subscribed to journals, and attended meetings, all of which reminded him how much he had enjoyed pure research as a grad student and how much he missed it. "I became more and more interested in the technical problems," he recalls. "I realized that venture capital was just another hectic business."

He took a position at Stanford as an adjunct professor, which gave him time to absorb the latest research in genetic algorithms and artificial intelligence. Yet he felt that both disciplines were somehow fundamentally lacking. Pragmatic by nature, Koza was frustrated-as many others would later become-by the growing gap between promise and performance in artificial intelligence. Meanwhile genetic algorithms presented a different kind of frustration: Although they proved to be excellent optimizers, perfect for tweaking well-defined systems, they lacked the creative capability to come up with novel solutions to their problems.

In 1987 Koza was on an airplane, returning to California from an AI conference in Italy, when he had the crucial insight that Holland himself would later deem revolutionary. AI was all promise, an underachieving prodigy. Genetic algorithms were all performance, reliable drones. Koza was 30,000 feet above Greenland when he asked himself why a genetic algorithm, so adept at refining pipelines, couldn't be used to evolve its own software. Why couldn't a computer program adapt *itself* and, in doing so, solve any problem fed into it?

The key was mating bits of computer programs together, not just strings of numbers. The old genetic algorithms worked to optimize specific parameters; Koza's leap in genetic programming allowed for open-ended evolutions of basic structure and so produced more novel and sophisticated designs. "If you're trying to breed a better racehorse, you could take a herd of horses out into a field and irradiate them and hope that, through random mutation, you get a better horse," Koza explains. "Or you could mate a champion with another champion."

His breakthrough moment came in October 1995. Working with only the most rudimentary information, he watched his computer evolve a circuit. Unlike the gas pipelines bred by genetic algorithms, his circuit didn't start with an inferior design that was optimized. Beginning from a pile of unconnected components-assorted resistors, capacitors and the like-his computer devised a complex electronic circuit.

Koza made a drawing of his new circuit and showed it to a colleague, who told him that it was a low-pass filter-a circuit used for cleaning up the signal passing through an amplifier. He also learned that someone had taken out a patent on it, which made him wonder whether genetic programming could evolve other patented circuits from scratch. He programmed the machine to design circuits with the attributes of

other patented devices and started churning out infringements by the dozen. “That’s when we began to see that genetic programming could be human-competitive,” he says. “If you remake a patented circuit, you’re doing something that people consider inventive.”

**NASA Signs On
**

January 25, 2005, looms large in the history of computer science as the day that genetic programming passed its first real Turing test: The examiner had no idea that he was looking at the intellectual property of a computer. This is especially significant because the U.S. Patent and Trademark Office requires a “non-obvious step”-a break from established practices or what someone might deduce from them-to grant an invention intellectual-property protection. The machine was demonstrably creative.

And that was just the start. Every day now, genetic programs continue to create the unexpected, the counterintuitive or the just plain weird. Take, for example, the antenna that was set to launch on NASA’s Space Technology 5 mission, a test platform for new technologies, as this magazine went to press. Several years ago, Koza protg Jason Lohn took a job as a computer scientist at NASA Ames Research Center, which had taken an interest in evolutionary approaches to problem solving, including genetic algorithms. “Antenna design has a black-art quality to it,” Lohn says. “A lot of it is intuition.”

Lohn got his hands on the antenna specs for the Space Technology 5 mission. He plugged in an antenna’s basic requirements and let the software run. What he got, several hundred generations later, appeared to be a mistake. “It looked like a bent paper clip,” he remembers. Lohn had no background in antenna design, but that was beside the point. “There’s no chapter in the textbooks on crooked wire antennas,” he says. He had his bent paper clip prototyped and put it in a test chamber. Sure enough, it provided the tricky combination of wide bandwidth and wide beam that NASA required. Like the duck-billed platypus, it looked preposterous but proved perfectly suited to its niche.

John Holland has lately been researching what such ingenuity might tell us about the creative process in humans. He believes that revolutionary ideas don’t come at random but are “new combinations of fairly standard parts with which we’re already familiar.” He cites as examples the internal combustion engine and the airplane, for which all the components were available long before the invention came along, lacking only someone with adequately broad knowledge, deep resources and the temperament to combine them. “Evolution is good at recombining building blocks to get innovations,” Holland says. The machine has inspired a new way to think about our own creative process: Perhaps extraordinary thinking is simply the product of gradual refinements and serendipitous recombinations. Darwin’s combination of mutation, sex and selection creates not just new species, or antennas: It spawns creativity itself.

**
The Search for Problems**

As genetic programming becomes pervasive over the next decade, the process of finding good solutions to difficult engineering problems will become efficient in the way that once-arduous tasks such as 3-D rendering have become routine. But, as with 3-D rendering, the real challenge will lie in deciding what to create. This is no trivial matter-to invent a car, even with an invention machine, you must be able to conceive of wanting a horseless carriage in the first place. In the future, as solutions become plentiful and cheap, the real test of creativity will come in the search for problems.

Meanwhile, Koza thinks he has already found a good one to solve, although it will require more than just an invention machine. Confident as ever in the power of rational thought (and in himself), he has undertaken the mission of reengineering U.S. elections. More specifically, he has developed a strategy to effectively eliminate the American electoral college. Here’s how it works:Since the Constitution allows each state to assign its electors any way it chooses, a critical mass of states could pledge to assign its electors to vote for the winner of the nationwide popular vote. In theory, as few as 11 states could make this work. The plan is slowly gaining traction among politicians and pundits, and a bill was recently introduced in the Illinois legislature that would make that state the first to back the plan.

As Koza waits for the machinations of politics to grind, he has his sights set on one more goal, perhaps genetic programming's ultimate Turing test: an invention that succeeds in the marketplace. (Passing muster with a patent examiner is nothing compared to nabbing a couple million customers.) That means finding a problem serious enough that people will pay to have it solved. Alas, figuring out what people really need is one thing genetic programming can't do. Koza will have to invent that for himself.

Jonathon Keats is writing a collection of fables called The Book of the Unknown.

AI