# Human-Competitive Machine Intelligence by Means of Genetic Algorithms

**John R. Koza**
Section on Medical Informatics
Department of Medicine
Stanford University
Stanford, California 94305
koza@stanford.edu
http://www.smi.stanford.edu/people/koza

## ABSTRACT

This paper demonstrates the correctness of John Holland's expectation that the genetic algorithm would have "applications to ... artificial intelligence" by showing examples of the automatic creation of human-competitive computer programs from a high-level statement of a problem's requirements. The paper argues that the field of design is a useful testbed for determining whether an automated technique can produce results that are competitive with human-produced results and then presents several results that are competitive with the products of human creativity and inventiveness.

## 1 Introduction

The subtitle of John Holland's pioneering l975 book *Adaptation in Natural and Artificial Systems* correctly anticipated that the genetic algorithm would have "applications to ... artificial intelligence."

When the entities in the evolving population are computer programs (as opposed to, say, fixed-length binary character strings), Holland's genetic algorithm can be used to perform the task of searching the space of computer programs for a program that solves, or approximately solves, a problem. This variation of the genetic algorithm (called genetic programming) can solve many types of problems, including problems of design, control, classification, and system identification.

In particular, genetic programming can be used to address the challenge of getting a computer to solve a problem without explicitly programming it. This challenge calls for an automatic system whose input is a high-level statement of a problem's requirements and whose output is a working program that solves the problem. Paraphrasing Arthur Samuel (1959), this challenge concerns

> How can computers be made to do what needs to be done, without being told exactly how to do it?

As Arthur Samuel (1983) further explained,

> "The aim [is] ... to get machines to exhibit behavior, which if done by humans, would be assumed to involve the use of intelligence."

There are now a number of instances where genetic programming has produced a computer program that is competitive with human performance on particular problems from the fields of computational molecular biology, cellular automata, sorting networks, and the synthesis of the design of complex structures, such as analog electrical circuits.

Section 3 describes genetic programming. Section 2 states what we mean when we say that an automatically created solution to a problem is competitive with the product of human creativity. Section 4 discusses the suitability of design problems for evaluating whether an automatic technique can produce results that are competitive with human-produced results. Section 5 describes three illustrative problems of automatic circuit synthesis (design) of analog electrical circuits. Section 6 describes how genetic programming is applied to such problems. Section 7 shows the results. Section 8 mentions additional problems of automatic circuit synthesis. Section 9 discusses the importance of illogic in creativity and inventiveness.

## 2 Genetic Programming

Genetic programming progressively breeds a population of computer programs over a series of generations by starting with a primordial ooze of thousands of randomly created computer programs and using the Darwinian principle of natural selection, recombination (crossover), mutation, gene duplication, gene deletion, and certain mechanisms of developmental biology. Specifically, genetic programming starts with an initial population of randomly generated computer programs composed of the given primitive functions and terminals. The programs in the population are, in general, of different sizes and shapes. The creation of the initial random population is a blind random search of the space of computer programs composed of the problem's available functions and terminals.

On each generation of a run of genetic programming, each individual in the population of programs is evaluated

as to its fitness in solving the problem at hand. The programs in generation 0 of a run almost always have exceedingly poor fitness for non-trivial problems of interest. Nonetheless, some individuals in a population will turn out to be somewhat more fit than others. These differences in performance are then exploited so as to direct the remainder of the search into promising areas of the search space. The Darwinian principle of reproduction and survival of the fittest is used to probabilistically select, on the basis of fitness, individuals from the population to participate in various operations. A small percentage (e.g., 9%) of the selected individuals are reproduced (copied) from one generation to the next. A very small percentage (e.g. 1%) of the selected individuals are mutated in a random way. The vast majority of the selected individuals participate in the genetic operation of crossover (sexual recombination) in which two offspring programs are created by recombining genetic material from two parents. The creation of the initial random population and the creation of offspring by the genetic operations are all performed so as to create syntactically valid, executable programs. After the genetic operations are performed on the current generation of the population, the population of offspring (i.e., the new generation) replaces the old generation. The tasks of measuring fitness, Darwinian selection, and genetic operations are then iteratively repeated over many generations. Detailed information about genetic programming can be found in Koza 1992; Koza and Rice 1992; Koza 1994a, 1994b; Koza, Bennett, Andre, and Keane 1999; Koza, Bennett, Andre, Keane, and Brave 1999, Banzhaf, Nordin, Keller, and Francone 1998; Langdon 1998; Kinnear 1994; Angeline and Kinnear 1996; Spector, Langdon, O'Reilly, and Angeline 1999; Koza, Goldberg, Fogel, and Riolo 1996; Koza, Deb, Dorigo, Fogel, Garzon, Iba, and Riolo 1997; Koza, Banzhaf, Chellapilla, Deb, Dorigo, Fogel, Garzon, Goldberg, Iba, Riolo 1998; and Banzhaf, Poli, Schoenauer, and Fogarty 1998; and Poli, Nordin, Langdon, and Fogarty 1999.

# 3 Human-Competitive Machine Intelligence

What do we mean when we say that an automatically created solution to a problem is competitive with human-produced results?

We are not referring to the fact that a computer can rapidly print ten thousand payroll checks or that a computer can compute $\pi$ to a million decimal places. Instead, we think it is fair to say that an automatically created result is competitive with one produced by human engineers, designers, mathematicians, or programmers if it satisfies any of the following eight criteria (or any other similarly stringent criterion):

**(A)** The result was patented as an invention in the past, is an improvement over a patented invention, or would qualify today as a patentable new invention.

**(B)** The result is equal to or better than a result that was accepted as a new scientific result at the time when it was published in a peer-reviewed scientific journal.

**(C)** The result is equal to or better than a result that was placed into a database or archive of results maintained by an internationally recognized panel of scientific experts.

**(D)** The result is publishable in its own right as a new scientific result — *independent* of the fact that the result was mechanically created.

**(E)** The result is equal to or better than the most recent human-created solution to a long-standing problem for which there has been a succession of increasingly better human-created solutions.

**(F)** The result is equal to or better than a result that was considered an achievement in its field at the time it was first discovered.

**(G)** The result solves a problem of indisputable difficulty in its field.

**(H)** The result holds its own or wins a regulated competition involving human contestants (in the form of either live human players or human-written computer programs).

Note that each of these criteria are couched in terms of producing results, that the results are measured in terms of standards outside the fields of artificial intelligence and machine learning.

Table 1 shows 14 instances of results where genetic programming has produced results that are competitive with the products of human creativity and inventiveness (Koza, Bennett, Andre, and Keane 1999). Each claim is accompanied by the particular criterion (from the list above) that establishes the basis for the claim. The instances in the table include classification problems from the field of computational molecular biology, a long-standing problem involving cellular automata, a problem of synthesizing the design of a minimal sorting network, and several problems of synthesizing the design of analog electrical circuits. As can be seen, 10 of the 14 instances in the table involve previously patented inventions.

# 4 Design as a Testbed for Machine Intelligence

The design process entails creation of a complex structure to satisfy user-defined requirements. The field of design is a good source of problems that can be used for determining whether an automated technique can produce results that are competitive with human-produced results. Design is a major activity of practicing engineers. Since the design process typically entails tradeoffs between competing considerations, the end product of the process is usually a satisfactory and compliant design as opposed

to a perfect design. Design is usually viewed as requiring creativity and human intelligence.

The design process for electrical circuits begins with a high-level description of the circuit's desired behavior and characteristics and entails creation of the topology and sizing of a satisfactory circuit.

**Table 1 Fourteen instances where genetic programming has produced results that are competitive with human-produced results.**

| | Claimed instance | Basis for claim |
|---|---|---|
| 1 | Creation of four different algorithms for the transmembrane segment identification problem for proteins | B, E |
| 2 | Creation of a sorting network for seven items using only 16 steps | A, D |
| 3 | Rediscovery of the Campbell ladder topology for lowpass and highpass filters | A, F |
| 4 | Rediscovery of "*M*-derived half section" and "constant *K*" filter sections | A, F |
| 5 | Rediscovery of the Cauer (elliptic) topology for filters | A, F |
| 6 | Automatic decomposition of the problem of synthesizing a crossover filter | A, F |
| 7 | Rediscovery of a recognizable voltage gain stage and a Darlington emitter-follower section of an amplifier and other circuits | A, F |
| 8 | Synthesis of 60 and 96 decibel amplifiers | A, F |
| 9 | Synthesis of analog computational circuits for squaring, cubing, square root, cube root, logarithm, and Gaussian functions | A, D, G |
| 10 | Synthesis of a real-time analog circuit for time-optimal control of a robot | G |
| 11 | Synthesis of an electronic thermometer | A, G |
| 12 | Synthesis of a voltage reference circuit | A, G |
| 13 | Creation of a cellular automata rule for the majority classification problem that is better than the Gacs-Kurdyumov-Levin (GKL) rule and all other known rules written by humans | D, E |
| 14 | Creation of motifs that detect the D–E–A-D box family of proteins and the manganese superoxide dismutase family | C |

The *topology* of a circuit includes specifying the gross number of components in the circuit, the type of each component (e.g., a capacitor), and a *netlist* specifying where each lead of each component is to be connected. *Sizing* involves specifying the values (typically numerical) of each of the circuit's components.

The field of design of analog and mixed analog-digital electrical circuits is especially challenging because there is no previously known general technique for automatically creating the topology and sizing of an analog circuit from a high-level statement of the design goals of the circuit.

Although considerable progress has been made in automating the synthesis of certain categories of purely digital circuits, the synthesis of analog circuits has not proved to be as amenable to automation. As O. Aaserud and I. Ring Nielsen (1995) observe,

> "Analog designers are few and far between. In contrast to digital design, most of the analog circuits are still handcrafted by the experts or so-called 'zahs' of analog design. The design process is characterized by a combination of experience and intuition and requires a thorough knowledge of the process characteristics and the detailed specifications of the actual product.
>
> "Analog circuit design is known to be a knowledge-intensive, multiphase, iterative task, which usually stretches over a significant period of time and is performed by designers with a large portfolio of skills. It is therefore considered by many to be a form of art rather than a science."

The remainder of this paper focuses on problems of synthesis of analog electrical circuits.

# 5 Three illustrative Problems of Automatic Circuit Synthesis

We start with three particular problems of analog circuit synthesis, namely the design of a lowpass filter circuit, the design of a high-gain, low-distortion, low-bias amplifier, and the design of a cube root computational circuit.

A simple analog *filter* is a one-input, one-output circuit that receives a signal as its input and passes the frequency components of the incoming signal that lie in a specified range (called the *passband*) while suppressing the frequency components that lie in all other frequency ranges (the *stopband*). Specifically, the goal is to design a lowpass filter composed of capacitors and inductors that

passes all frequencies below 1,000 Hertz (Hz) and suppresses all frequencies above 2,000 Hz.

An amplifier is a one-input, one-output circuit whose output is a constant multiple of its input. We are seeking a high-gain, low-distortion, low-bias amplifier composed of transistors, diodes, capacitors, resistors, and connections to power sources.

An analog computational circuit is a one-input, one-output circuit whose output is a specified mathematical function. The design of computational circuits is exceedingly difficult even for seemingly mundane mathematical functions. Success often relies on the clever exploitation of some aspect of the underlying device physics of the components that is unique to the particular desired mathematical function. Because of this, the implementation of each different mathematical function typically requires an entirely different clever insight and an entirely different circuit. We are seeking a computational circuit composed of transistors, diodes, capacitors, resistors, and connections to power sources.

# 6 Applying Genetic Programming to Circuit Synthesis

Genetic programming can be applied to the problem of synthesizing circuits if a mapping is established between the program trees (rooted, point-labeled trees with ordered branches) used in genetic programming and the labeled cyclic graphs germane to electrical circuits. The principles of developmental biology provide the motivation for mapping trees into circuits by means of a developmental process that begins with a simple embryo. For circuits, the initial circuit typically includes a test fixture consisting of certain fixed components (such as a source resistor, a load resistor, an input port, and an output port) as well as an embryo consisting of one or more modifiable wires. Until the modifiable wires are modified, the circuit does not produce interesting output. An electrical circuit is developed by progressively applying the functions in a circuit-constructing program tree to the modifiable wires of the embryo (and, during the developmental process, to succeeding modifiable wires and components). A single electrical circuit is created by executing the functions in an individual circuit-constructing program tree from the population. The functions are progressively applied in a developmental process to the embryo and its successors until all of the functions in the program tree are executed. That is, the functions in the circuit-constructing program tree progressively side-effect the embryo and its successors until a fully developed circuit eventually emerges. The functions are applied in a breadth-first order.

The functions in the circuit-constructing program trees are divided into five categories:

(1) topology-modifying functions that alter the topology of a developing circuit,,
(2) component-creating functions that insert components into a developing circuit,
(3) development-controlling functions that control the development process by which the embryo and its successors become a fully developed circuit,
(4) arithmetic-performing functions that appear in subtrees as argument(s) to the component-creating functions and specify the numerical value of the component, and
(5) automatically defined functions that appear in the automatically defined functions and potentially enable certain substructures of the circuit to be reused (with parameterization).

Before applying genetic programming to a problem of circuit design, seven major preparatory steps are required: (1) identify the embryonic circuit, (2) determine the architecture of the circuit-constructing program trees, (3) identify the primitive functions of the program trees, (4) identify the terminals of the program trees, (5) create the fitness measure, (6) choose control parameters for the run, and (7) determine the termination criterion and method of result designation.

A detailed discussion concerning how to apply these seven preparatory steps to particular problems is found in Koza, Bennett, Andre, and Keane 1999 (chapter 25).

# 7 Results for Illustrative Problems

## 7. 1 Campbell 1917 Ladder Filter Patent

The best circuit (figure 1) of generation 49 of one run of genetic programming on the problem of synthesizing a lowpass filter is a 100% compliant circuit.

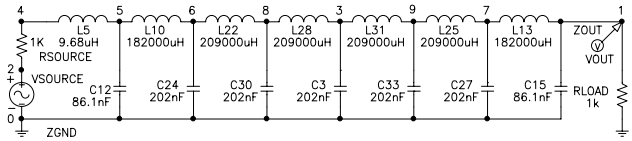


**Figure 1 Evolved Campbell filter.**

The evolved circuit is what is now called a cascade (ladder) of identical π sections and is shown and analyzed in Koza, Bennett, Andre, and Keane 1999 (chapter 25). The evolved circuit has the recognizable topology of the circuit for which George Campbell of American Telephone and Telegraph received U. S. patent 1,227,113 in 1917. Claim 2 of Campbell's patent covered,

> "An electric wave filter consisting of a connecting line of negligible attenuation composed of a plurality of sections, each section including a capacity element and an inductance element, one of said elements of each section being in series with the line and the other in shunt across the line, said capacity and inductance elements having precomputed values dependent upon the upper limiting frequency and the lower limiting frequency of a range of frequencies it is desired to transmit without attenuation, the values of said capacity and inductance elements being so proportioned that the structure transmits with practically negligible attenuation sinusoidal currents of all frequencies

lying between said two limiting frequencies, while attenuating and approximately extinguishing currents of neighboring frequencies lying outside of said limiting frequencies."

In addition to possessing the topology of the Campbell filter, the numerical value of all the components in the evolved circuit closely approximate the numerical values specified in Campbell's 1917 patent. But for the fact that this 1917 patent has expired, the evolved circuit would infringe on the Campbell patent.

The legal criteria for obtaining a U. S. patent are that the proposed invention be "new" and "useful" and

> "... the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would [not] have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains." (35 *United States Code* 103a).

The fact that genetic programming rediscovered both the topology and sizing of an electrical circuit that was unobvious "to a person having ordinary skill in the art" establishes that this evolved result satisfies Arthur Samuel's criterion for artificial intelligence and machine learning (quoted in section 1).

Since filing for a patent entails the expenditure of a considerable amount of time and money, patents are generally sought, in the first place, only if an individual or business believes the inventions are likely to be useful in the real world and economically rewarding. Patents are only issued if an arms-length examiner is convinced that the proposed invention is novel, useful, and satisfies the statutory test for unobviousness.

## 7.2 Zobel 1925 "*M*-Derived Half Section" Patent

Since the genetic algorithm is a probabilistic algorithm, different runs produce different results. In another run of this same problem of synthesizing a lowpass filter, a 100%-compliant circuit (figure 2) was evolved in generation 34.

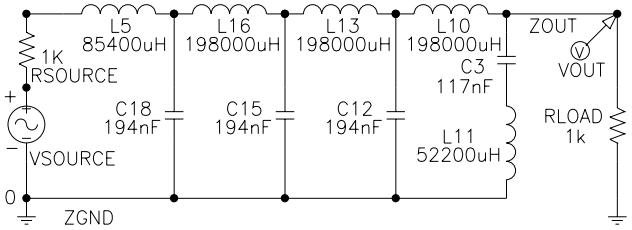


**Figure 2 Evolved Zobel filter.**

This evolved circuit (presented in Koza, Bennett, Andre, and Keane 1999, chapter 25) is equivalent to a cascade of three symmetric T-sections and an *M*-derived half section. Otto Zobel of American Telephone and Telegraph Company invented and received a patent for an "*M*-derived half section" used in conjunction with one or more "constant K" sections.

## 7.3 Cauer 1934 – 1936 Elliptic Patents

In yet another run of this same problem of synthesizing a lowpass filter, a 100% compliant circuit (figure 3) emerged in generation 31 (Koza, Bennett, Andre, and Keane 1999, chapter 27).

This circuit has the recognizable elliptic topology that was invented and patented by Wilhelm Cauer in 1934, 1935, and 1936. The Cauer filter was a significant advance (both theoretically and commercially) over the earlier filter designs of Campbell, Zobel, Johnson, Butterworth, and Chebychev. For example, for one commercially important set of specifications for telephones, a fifth-order elliptic filter matches the behavior of a 17th-order Butterworth filter or an eighth-order Chebychev filter. The fifth-order elliptic filter has one less component than the eighth-order Chebychev filter. As Van Valkenburg (1982) relates in connection with the history of the elliptic filter:

> "Cauer first used his new theory in solving a filter problem for the German telephone industry. His new design achieved specifications with one less inductor than had ever been done before. The world first learned of the Cauer method not through scholarly publication but through a patent disclosure, which eventually reached the Bell Laboratories. Legend has it that the entire Mathematics Department of Bell Laboratories spent the next two weeks at the New York Public library studying elliptic functions. Cauer had studied mathematics under Hilbert at Goettingen, and so elliptic functions and their applications were familiar to him."

Genetic programming did not, of course, study mathematics under Hilbert or anybody else. Instead, the elliptic topology emerged from a run of genetic programming as a natural consequence of the problem's fitness measure and natural selection – not because the run was primed with domain knowledge about elliptic functions or filters or electrical circuitry. Genetic programming opportunistically *reinvented* the elliptic topology because necessity (fitness) is the mother of invention.

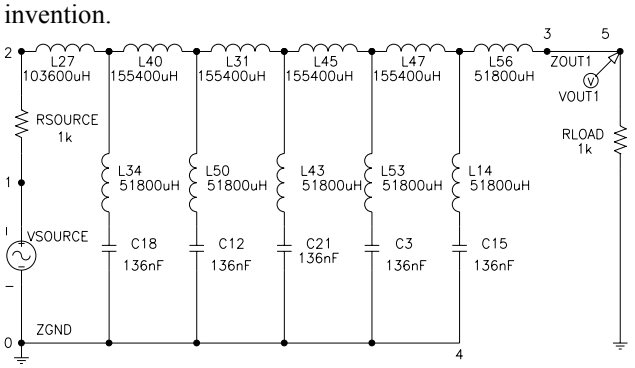


**Figure 3 Evolved Cauer (elliptic) filter topology.**

## 7.4 Darlington 1952 Emitter-Follower Patent

Sidney Darlington of the Bell Telephone Laboratories obtained some 40 patents on numerous fundamental electronic circuits. In particular, he obtained U. S. patent 2,663,806 for what is now called the Darlington emitter-follower section. Darlington emitter-follower sections have been evolved on numerous occasions in the process of solving problems of analog circuit synthesis.

Claim 1 of Darlington's 1952 patent covers

> "A signal translating device comprising a pair of transistors of like conductivity type and each including a base, an emitter and a collector, means directly connecting the collectors together, means directly connecting the emitter of one transistor to the base of the other, and individual electrical connections to the other emitter and base."

In a similar vein, claim 3 covers

> "A signal translating device comprising a pair of transistors of like conductivity type and each including a base, an emitter and a collector, means directly connecting the emitters together, means directly connecting the collector of one transistor to the base of the other, and individual electrical connections to the other collector and base."

Claim 5 is more general and covers the case where any two like electrodes of the transistor are connected.

> "A signal translating device comprising a pair of transistors of like conductivity type and each including a base, an emitter and a collector, means directly connecting two like electrodes of said transistors together, means directly connecting another electrode of one transistor to an unlike electrode, other than one of said like electrodes, of the other transistor, and individual electrical connections to the other emitter and base."

The Darlington patent also refers to an optional external connection to the connection between the leads of the two transistors. For example, claim 2 is a dependent claim based on claim 1 (where the collectors are connected together) and covers

> "A signal translating device in accordance with claim 1 comprising an additional electrical connection to the connected emitter and base."

Similarly, claim 4 is based on claim 3 (where the emitters are connected together) and covers

> "A signal translating device in accordance with claim 3 comprising an additional electrical connection to the connected collector and base."

Table 2 shows 12 instances in Koza, Bennett, Andre, and Keane 1999 where genetic programming evolved a circuit containing one of the canonical Darlington sections. The table identifies the particular claims (1, 2, 3, or 4) of U. S. patent 2,663,806 that genetic programming has infringed.

For example, figure 4 shows the best circuit from generation 86 of a run of the problem of evolving a high-gain, low-distortion, low-bias amplifier.

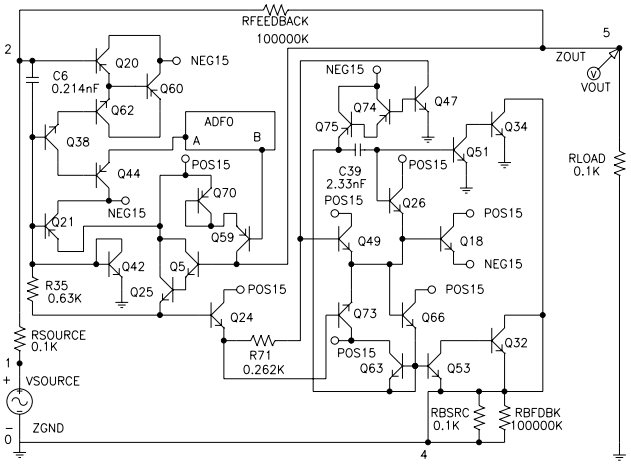


**Figure 4 Evolved 96 dB amplifier.**

The circuit has 25 transistors, no diodes, two capacitors, and two resistors and contains a Darlington emitter-follower section (involving transistors Q25 and Q5).

As another example, figure 5 shows the best-of-run circuit from generation 57 of the problem of synthesizing a cube root computational circuit. The circuit has 38 transistors, seven diodes, and 18 resistors.

**Table 2 Twelve instances where genetic programming appears to have infringed Darlington's emitter-follower patent.**

| Problem | Type | Patent claim |
|---|---|---|
| 96 dB amplifier | *npn* | 1 |
| 96 dB amplifier | *npn* | 3 |
| Squaring circuit | *npn* | 1 |
| Squaring circuit | *pnp* | 4 |
| Cubing circuit | *pnp* | 3 |
| Cubing circuit | *pnp* | 3 |
| Cubing circuit | *pnp* | 3 |
| Square root circuit | *pnp* | 2 |
| Cube root circuit | *pnp* | 2 |
| Cube root circuit | *pnp* | 1 |
| Cube root circuit | *pnp* | 2 |
| Logarithmic circuit | *pnp* | 4 |

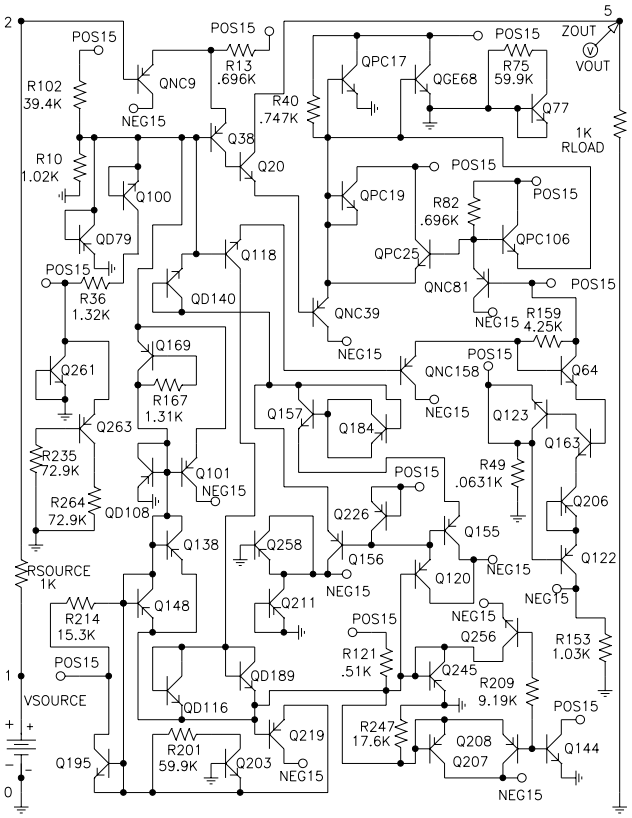


**Figure 5 Evolved cube root computational circuit.**

# 8 Additional Results

In addition, genetic programming has been applied to the problem of automatic synthesis of both the topology and sizing of additional analog electrical circuits, including other filters (highpass, bandpass, crossover, comb, and asymmetric filters), other amplifiers, other computational circuits (square root, squaring, cubing, logarithmic, and Gaussian), a time-optimal controller circuit, source identification circuits, a temperature-sensing circuit, and a voltage reference circuits. See Koza, Bennett, Andre, Keane, and Dunlap 1997; Koza, Bennett, Andre, and Keane 1999; Koza, Bennett, Andre, Keane, and Brave 1999). High-gain amplifiers, computational circuits, electronic thermometers, and voltage reference circuits were all covered by one or more patents when they were first invented.

# 9 The Illogical Nature of Creativity and Evolution

The biological metaphor underlying the genetic algorithm and genetic programming is very different from the underpinnings of all other techniques that have previously been tried in pursuit of the goal of automatically creating computer programs.

Many computer scientists and mathematicians unquestioningly assume that every problem-solving technique must be logically sound, deterministic, logically consistent, and parsimonious. Accordingly, most conventional methods of artificial intelligence and machine learning are constructed so as to possess these characteristics. However, logic does not govern two of the most important and significant types of processes for solving complex problems, namely the invention process (performed by creative humans) and the evolutionary process (occurring in nature).

A new idea that can be logically deduced from facts that are known in a field, using transformations that are known in a field, is not considered to be an invention. There must be what the patent law refers to as an "illogical step" (i.e., an unjustified step) to distinguish a putative invention from that which is readily deducible from that which is already known. Humans supply the critical ingredient of “illogic” to the invention process. Interestingly, everyday usage parallels the patent law concerning inventiveness: People who mechanically apply existing facts in well-known ways are summarily dismissed as being uncreative. Logical thinking is unquestionably useful for many purposes. It usually plays an important role in setting the stage for an invention. But, at the end of the day, logical thinking is not sufficient in the invention process.

Recalling his invention in 1927 of the negative feedback amplifier, Harold S. Black (1977) said,

> "Then came the morning of Tuesday, August 2, 1927, when the concept of the negative feedback amplifier came to me in a flash while I was crossing the Hudson River on the Lackawanna Ferry, on my way to work. For more than 50 years, I have pondered how and why the idea came, and I can't say any more today than I could that morning. All I know is that after several years of hard work on the problem, I suddenly realized that if I fed the amplifier output back to the input, in reverse phase, and kept the device from oscillating (singing, as we called it then), I would have exactly what I wanted: a means of canceling out the distortion of the output. I opened my morning newspaper and on a page of *The New York Times* I sketched a simple canonical diagram of a negative feedback amplifier plus the equations for the amplification with feedback."

Of course, inventors are not oblivious to logic and knowledge. They do not thrash around using blind random search. Black did not try to construct the negative feedback amplifier from neon bulbs or doorbells. Instead, "several years of hard work on the problem" set the stage and brought his thinking into the proximity of a solution. Then, at the critical moment, Black made his “illogical” leap. This unjustified leap constituted the invention.

The design of complex entities by the evolutionary process in nature is another important type of problem-solving that is not governed by logic. In nature, solutions to design problems are discovered by the probabilistic process of evolution and natural selection. This process is not guided by mathematical logic. Indeed, inconsistent and contradictory alternatives abound. In fact, such genetic diversity is necessary for the evolutionary process

to succeed. Significantly, the solutions evolved by evolution and natural selection almost always differ from those created by conventional methods of artificial intelligence and machine learning in one very important respect. Evolved solutions are not brittle; they are usually able to grapple with the perpetual novelty of real environments.

Since genetic programming is a probabilistic process that is not encumbered by the preconceptions that often channel human thinking down familiar paths, it often creates novel designs.

Similarly, genetic programming is not guided by the inference methods of formal logic in its search for a computer program to solve a given problem. When the goal is the automatic creation of computer programs, we believe that the non-logical approach used in the invention process and in natural evolution are far more fruitful than the logic-driven and knowledge-based principles of conventional artificial intelligence and machine learning.

# 10 Conclusion

This paper has demonstrated the correctness of Holland's expectation that genetic algorithms would have "applications to ... artificial intelligence" and, in particular, to the automatic creation of computer programs from a high-level statement of a problem's requirements.

# Acknowledgments

The work described in various parts of this paper has been done in collaboration with Forrest H Bennett III, David Andre, Martin A. Keane, Frank Dunlap, and Jason Lohn.

# References

Aaserud, O. and Nielsen, I. Ring. 1995. Trends in current analog design: A panel debate. *Analog Integrated Circuits and Signal Processing*. 7(1) 5-9.

Angeline, Peter J. and Kinnear, Kenneth E. Jr. (editors). 1996. *Advances in Genetic Programming 2*. Cambridge, MA: The MIT Press.

Banzhaf, Wolfgang, Nordin, Peter, Keller, Robert E., and Francone, Frank D. 1998. *Genetic Programming – An Introduction*. San Francisco, CA: Morgan Kaufmann and Heidelberg: dpunkt.

Banzhaf, Wolfgang, Poli, Riccardo, Schoenauer, Marc, and Fogarty, Terence C. 1998. *Genetic Programming: First European Workshop. EuroGP'98. Paris, France, April 1998 Proceedings. Paris, France. April l998*. Lecture Notes in Computer Science. Volume 1391. Berlin, Germany: Springer-Verlag.

Black, Harold S. 1977. Inventing the negative feedback amplifier. *IEEE Spectrum*. December 1977. Pp. 55 – 60.

Campbell, George A. 1917. *Electric Wave Filter*. Filed July 15, 1915. U. S. Patent 1,227,113. Issued May 22, 1917.

Cauer, Wilhelm. 1934. *Artificial Network*. U. S. Patent 1,958,742. Filed June 8, 1928 in Germany. Filed December 1, 1930 in United States. Issued May 15, 1934.

Cauer, Wilhelm. 1935. *Electric Wave Filter*. U. S. Patent 1,989,545. Filed June 8, 1928. Filed December 6, 1930 in United States. Issued January 29, 1935.

Cauer, Wilhelm. 1936. *Unsymmetrical Electric Wave Filter*. Filed November 10, 1932 in Germany. Filed November 23, 1933 in United States. Issued July 21, 1936.

Holland, John H. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. Ann Arbor, MI: University of Michigan Press 1975. Second edition. Cambridge, MA: The MIT Press 1992.

Kinnear, Kenneth E. Jr. (editor). 1994. *Advances in Genetic Programming*. Cambridge, MA: MIT Press.

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

Koza, John R. 1994a. *Genetic Programming II: Automatic Discovery of Reusable Programs.* Cambridge, MA: MIT Press.

Koza, John R. 1994b. *Genetic Programming II Videotape: The Next Generation*. Cambridge, MA: MIT Press.

Koza, John R., Banzhaf, Wolfgang, Chellapilla, Kumar, Deb, Kalyanmoy, Dorigo, Marco, Fogel, David B., Garzon, Max H., Goldberg, David E., Iba, Hitoshi, and Riolo, Rick. (editors). 1998. *Genetic Programming 1998: Proceedings of the Third Annual Conference*. San Francisco, CA: Morgan Kaufmann.

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

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

Koza, John R., Bennett III, Forrest H, Andre, David, Keane, Martin A, and Dunlap, Frank. 1997. Automated synthesis of analog electrical circuits by means of genetic programming. *IEEE Transactions on Evolutionary Computation*. 1(2). Pages 109 – 128.

Koza, John R., Deb, Kalyanmoy, Dorigo, Marco, Fogel, David B., Garzon, Max, Iba, Hitoshi, and Riolo, Rick L. (editors). 1997. *Genetic Programming 1997: Proceedings of the Second Annual Conference* San Francisco, CA: Morgan Kaufmann.

Koza, John R., Goldberg, David E., Fogel, David B., and Riolo, Rick L. (editors). 1996. *Genetic Programming 1996: Proceedings of the First Annual Conference*. Cambridge, MA: The MIT Press.

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

Langdon, W. B. 1998. *Genetic Programming and Data Structures: Genetic Programming + Data Structures = Automatic Programming!* Amsterdam: Kluwer.

Poli, Riccardo, Nordin, Peter, Langdon, William B., and Fogarty, Terence C. 1999. *Genetic Programming: Second European Workshop. EuroGP'99. Proceedings*. Lecture Notes in Computer Science. Volume 1598. Berlin, Germany: Springer-Verlag.

Samuel, Arthur L. 1959. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development.* 3(3): 210–229.

Samuel, Arthur L. 1983. AI: Where it has been and where it is going. *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*. Los Altos, CA: Morgan Kaufmann. Pages 1152 – 1157.

Spector, Lee, Langdon, William B., O'Reilly, Una-May, and Angeline, Peter (editors). 1999. *Advances in Genetic Programming 3*. Cambridge, MA: The MIT Press.

Valkenburg, M. E. 1982. *Analog Filter Design*. Fort Worth, TX: Harcourt Brace Jovanovich.

Zobel, Otto Julius. 1925. *Wave Filter*. Filed January 15, 1921. U. S. Patent 1,538,964. Issued May 26, 1925.