Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2004), *18*, 245–269. Printed in the USA.

DOI: 10.1017/S089006040404017X

Invention and creativity in automated design by means of genetic programming

JOHN R. KOZA,[1] MARTIN A. KEANE,[2] MATTHEW J. STREETER,[3] THOMAS P. ADAMS,[3] AND LEE W. JONES[3]

[1]Stanford University, School of Medicine, Biomedial Informatics Program, 251 Campus Drive, Stanford, California 94305-5479, USA
[2]Econometrics Inc., 1300 North Lake Shore #22B, Chicago, Illinois 60610, USA
[3]Genetic Programming Inc., 990 Villa Street, Mountain View, California 94041, USA

(RECEIVED September 26, 2003; ACCEPTED August 19, 2004)

Abstract

Some designs are sufficiently creative that they are considered to be inventions. The invention process is typically characterized by a singular moment when the prevailing thinking concerning a long-standing problem is, in a "flash of genius," overthrown and replaced by a new approach that could not have been logically deduced from what was previously known. This paper discusses such logical discontinuities using an example based on the history of one of the most important inventions of the 20th century in electrical engineering, namely, the invention of negative feedback by AT&T's Harold S. Black. This 1927 invention overthrew the then prevailing idiom of positive feedback championed by Westinghouse's Edwin Howard Armstrong. The paper then shows how this historically important discovery can be readily replicated by an automated design and invention technique patterned after the evolutionary process in nature, namely, genetic programming. Genetic programming employs Darwinian natural selection along with analogs of recombination (crossover), mutation, gene duplication, gene deletion, and mechanisms of developmental biology to breed an ever improving population of structures. Genetic programming rediscovers negative feedback by conducting an evolutionary search for a structure that satisfies Black's stated high-level goal (i.e., reduction of distortion in amplifiers). Like evolution in nature, genetic programming conducts its search probabilistically without resort to logic using a process that is replete with logical discontinuities. The paper then shows that genetic programming can routinely produce many additional inventive and creative results. In this regard, the paper discusses the automated rediscovery of numerous 20th-century patented inventions involving analog electrical circuits and controllers, the Sallen–Key filter, and six 21st-century patented inventions. In addition, two patentable new inventions (controllers) have been created in the same automated way by means of genetic programming. The paper discusses the promising future of automated invention by means of genetic programming in light of the fact that, to date, increased computer power has yielded progressively more substantial results, including numerous human-competitive results, in synchrony with Moore's law. The paper argues that evolutionary search by means of genetic programming is a promising approach for achieving creative, human-competitive, automated design because illogic and creativity are inherent in the evolutionary process.

Keywords: Analog Circuits; Automated Design; Creativity; Evolutionary Computation; Genetic Programming

1. INTRODUCTION

Design is a major activity of practicing engineers. Engineers are often called upon to design complex structures (e.g., electrical circuits, controllers, antennas, aerodynamic shapes, mechanical systems, and networks of chemical reactions) that satisfy prespecified high-level design and performance goals. Some designs are sufficiently creative that they are considered to be inventions.

In designing complex structures, human engineers typically employ logic and previously known domain knowledge about the field of interest. Conventional approaches to automated design (such as those employing artificial intel-

Reprint requests to: John R. Koza, PO Box K, Los Altos, CA 94023, USA. E-mail: koza@stanford.edu

ligence) are typically knowledge intensive, logically sound, and deterministic.

Two of the most successful approaches to design, namely, the evolutionary process (occurring in nature) and the invention process (performed by creative humans), are not logical, deterministic, or knowledge intensive. The fact that these two highly successful approaches are so different from conventional approaches to automated design suggests that there may be important lessons to be learned from them.

The design of complex entities by the evolutionary process in nature is a nondeterministic process that is not governed by logic. In nature, solutions to design problems are discovered by means of evolution and natural selection. Evolution is not deterministic. It does not rely on a knowledge base. Also, it is certainly is not guided by mathematical logic. Indeed, one of the most important characteristics of the evolutionary process is that it actively generates and actively maintains inconsistent and contradictory alternatives throughout the process. Logically sound systems, of course, do not do that. In fact, the generation and maintenance of inconsistent and contradictory alternatives (called *genetic diversity*) is a precondition for the success of the evolutionary process.

Likewise, the invention process (performed by creative humans) is a nondeterministic process that is not governed by logic. The invention process is typically characterized by a singular moment when the prevailing thinking concerning a long-standing problem is, in a "flash of genius," overthrown and replaced by an new approach that could not have been logically deduced from what was previously known. That is, inventions are characterized by a logical discontinuity that distinguishes the creative new design from that which can be logically deduced. In this connection, it is noteworthy that 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 worthy of a patent. A new idea is patentable only if there is an "illogical step," that is, a logically unjustified step. In the patent law, this legally required illogical step is sometimes referred to as a flash of genius, and it is the essence of inventiveness and creativity. In short, both the invention process and the evolutionary process in nature and are very different from conventional approaches to automated design (such as those employing artificial intelligence).

Section 2 provides general background on genetic programming: an automated design and invention technique patterned after the evolutionary process in nature.

Section 3 discusses the logical discontinuities inherent in the invention process using an example based on the history of one of the most important inventions of the 20th century in electrical engineering, namely, the invention of negative feedback by AT&T's Harold S. Black. This 1927 invention overthrew the then prevailing idiom of positive feedback championed by Westinghouse's Edwin Howard Armstrong. This section shows how negative feedback can be readily reinvented in an automated way by means of genetic programming. Genetic programming accomplishes this by searching for a structure that satisfies Black's stated high-level goal (i.e., reduction of distortion in amplifiers). Like evolution in nature, the genetic search is conducted probabilistically without resort to logic, without a knowledge base, and using a process that is replete with logical discontinuities.

Section 4 describes how an additional significant 20th-century invention (the Sallen–Key filter) can be readily reinvented in a similar automated way by means of genetic programming. Section 5 then shows that six 21st-century patented inventions can be reinvented by genetic programming. (Section 2 demonstrates that a 19th-century patented invention in mechanical engineering and numerous 20th-century patented inventions can be similarly reinvented by genetic programming.) Section 6 shows that patentable new inventions can be invented by means of genetic programming in a similar automated way. Together, Sections 3–6 show that genetic programming can routinely produce inventive and creative results.

Section 7 discusses the promising future of automated invention by means of genetic programming in light of the fact that, to date, increased computer power has yielded progressively more substantial results in synchrony with Moore's law. Section 8 discusses the commercial practicality of automated analog circuit synthesis by means of genetic programming. Section 9 is the conclusion.

2. BACKGROUND ON GENETIC PROGRAMMING

Genetic programming starts from a high-level statement of what needs to be done and automatically creates a computer program to solve the problem. Genetic programming uses the Darwinian principle of natural selection along with analogs of recombination (crossover), mutation, gene duplication, gene deletion, and mechanisms of developmental biology to breed an ever improving population of programs (Koza, 1990, 1992, 1994*a*, 1994*b*; Koza & Rice, 1992; Banzhaf et al., 1998; Koza, Bennett, Andre, & Keane, 1999; Koza, Bennett, Forrest, Andre, Keane, & Scott, 1999; Koza, Keane, Streeter, Mydlowec, Yu, & Lanza, 2003; Koza, Keane, Streeter, Mydlowec, Yu, Lanza, & Fletcher, 2003). Genetic programming is an extension of the genetic algorithm (Holland, 1975) to the arena of computer programs. For additional sources of information about genetic programming, visit www.genetic-programming.org.

2.1. The genetic programming algorithm

Genetic programming typically starts with a population of randomly generated computer programs composed of the available programmatic ingredients. Genetic programming iteratively transforms a population of computer programs into a new generation of the population by applying ana-

https://doi.org/10.1017/S089006040404017X Published online by Cambridge University Press

logs of naturally occurring genetic operations. These operations are applied to individual(s) selected from the population. The individuals are probabilistically selected to participate in the genetic operations based on their fitness at solving the problem at hand. The iterative transformation of the population is executed inside the main loop (called a *generation*) of a run of genetic programming.

Specifically, genetic programming breeds computer programs to solve problems by executing the following three steps:

1. Generate an initial set (called the *population*) of compositions (typically random) of the functions and terminals (explained later) appropriate for the problem.
2. Iteratively perform the following group of substeps (called a *generation*) on the population of programs until the termination criterion has been satisfied:
   A. Execute each program in the population and assign it a fitness value using the problem's fitness measure.
   B. Create a new population (the next generation) of programs by applying the following operations to program(s) selected from the population with a probability based on fitness (with reselection allowed).
      i. *Reproduction:* Copy the selected program to the new population.
      ii. *Crossover:* Create a new offspring program for the new population by recombining randomly chosen parts of two selected programs.
      iii. *Mutation:* Create one new offspring program for the new population by randomly mutating a randomly chosen part of the selected program.
      iv. *Architecture-altering operations:* Create one new offspring program for the new population by applying a selected architecture-altering operation to the selected program.
3. Designate an individual program (e.g., the individual with the best fitness) as the run's result. This result may be a solution (or approximate solution) to the problem.

2.2. Human-competitive results produced by genetic programming

Genetic programming can be applied to problems in a variety of fields, including design problems.

There are, at the time of this writing, 24 known instances where genetic programming has duplicated the functionality of a previously patented invention, infringed a previously issued patent, or created a patentable new invention (Koza, 2003). Specifically, there is one instance where genetic programming has created an entity that either infringes or duplicates the functionality of a previously patented 19th-century invention, 15 instances where genetic programming has done the same with respect to a previously patented 20th-century invention, six instances where genetic programming has done the same with respect to a previously patented 21st-century invention, and two instances where genetic programming has created a patentable new invention (discussed later in Section 6).

Table 1 provides information on 22 of the above human-competitive results that relate to previously patented inventions. (The two patentable new inventions are discussed later in Section 6.) Twelve of the results in Table 1 infringe previously issued patents, and 10 duplicate the functionality of previously patented inventions in a noninfringing way.

It should also be mentioned that there are, at the time of this writing, 14 other known instances where genetic programming has produced a human-competitive result (using the detailed definition of this term found in Koza, Keane, Streeter, Mydlowec, Yu, & Lanza, 2003) that are not patent related. These include the design of an X-Band Antenna for NASA's Space Technology 5 Mission (Lohn et al., 2004), quantum computing elements (Spector et al., 1998; Spector, Barnum, & Bernstein, 1999; Spector, Barnum, Bernstein, & Swamy, 1999; Spector, 2004), a sorting network (Koza, Bennett, Andre, & Keane, 1999), game-playing strategies (Luke, 1998; Andre & Teller, 1999), algorithms for cellular automata (Andre et al., 1996), and algorithms for protein segment classification (Koza, Bennett, Andre, & Keane, 1999).

2.3. Automatic design of analog electrical circuits by means of genetic programming

Starting in Section 3, this paper presents a number of instances where genetic programming has automatically created both the topology (graphical structure) and sizing (numerical component values) for analog electrical circuits composed of transistors, capacitors, resistors, and other components. In each instance, genetic programming starts from a high-level statement of a circuit's desired behavior and characteristics (e.g., its desired output given its input). The process entails creation of both the topology and the sizing of a satisfactory circuit.

Specifically, the *topology* of a circuit comprises

- the total number of components in the circuit;
- the type of each component (e.g., resistor, capacitor, transistor) at each location in the circuit; and
- a list of all the connections that may exist between the leads of the circuit's components, input ports, output ports, power sources (if any), and ground.

The *sizing* of a circuit consists of the value(s), if any, associated with each component. The sizing of a component is usually a numerical value (e.g., the capacitance of a capacitor).

As Aaserud and Nielsen (1995) noted,

https://doi.org/10.1017/S089006040404017X Published online by Cambridge University Press

Table 1. *Twenty-two previously patented inventions reinvented by genetic programming*

Invention	Date	Inventor	Place	Patent	Reference
Mechanical system composed of rigid members for drawing a straight line without reference to another straight line	1841	Robert Willis	Great Britain	British 6258	Lipson (2004)
Ladder filter	1917	George Campbell	AT&T	US 1,227,113	Section 25.15.1 of Koza et al. (1999*a*) and Section 5.2 of Koza et al. (2003*b*)
Crossover filter	1925	Otto Julius Zobel	AT&T	US 1,538,964	Section 32.3 of Koza et al. (1999*a*)
"*M*-derived half section" filter	1925	Otto Julius Zobel	AT&T	US 1,538,964	Section 25.15.2 of Koza et al. (1999*a*)
Cauer (elliptic) topology for filters	1934–1936	Wilhelm Cauer	University of Gottingen	US 1,958,742, US 1,989,545	Section 27.3.7 of Koza et al. (1999*a*)
Negative feedback	1937	Harold S. Black	AT&T	US 2,102,670, US 2,102,671	Chapter 14 of Koza et al. (2003*b*)
Proportional, integrative, and derivative controller	1939	Albert Callender & Allan Stevenson	Imperial Chemical Limited	US 2,175,985	Section 9.2 of Koza et al. (2003*b*)
Second-derivative controller	1942	Harry Jones	Brown Instrument Co.	US 2,282,726	Section 3.7 of Koza et al. (2003*b*)
Darlington emitter–follower section	1953	Sidney Darlington	Bell Telephone Laboratories	US 2,663,806	Section 42.3 of Koza et al. (1999*a*)
Philbrick circuit	1956	George Philbrick	George A. Philbrick Research	US 2,730,679	Section 4.3 of Koza et al. (2003*b*)
Sorting network	1962	Daniel G. O'Connor & Raymond J. Nelson	General Precision, Inc.	US 3,029,413	Sections 21.4.4, 23.6, and 57.8.1 of Koza et al. (1999*a*)
NAND circuit	1971	David H. Chung & Bill H. Terrell	Texas Instruments Inc.	US 3,560,760	Section 4.4 of Koza et al. (2003*b*)
Computational circuits	Numerous	Numerous	Numerous	Numerous	Section 47.5.3 of Koza et al. (1999*a*)
Electronic thermometer	Numerous	Numerous	Numerous	Numerous	Section 49.3 of Koza et al. (1999*a*)
Voltage reference circuit	Numerous	Numerous	Numerous	Numerous	Section 50.3 of Koza et al. (1999*a*)
60- and 96-dB amplifiers	Numerous	Numerous	Numerous	Numerous	Section 45.3 of Koza et al. (1999*a*)
Cubic function generator	2000	Stefano Cipriani & Anthony A. Takeshian	Conexant Systems, Inc.	US 6,160,427	Section 15.4.5 of Koza et al. (2003*b*)
Mixed analog–digital variable capacitor circuit	2000	Turgut Sefket Aytur	Lucent Technologies Inc.	US 013,958	Section 15.4.2 of Koza et al. (2003*b*)
Voltage–current conversion circuit	2000	Akira Ikeuchi & Naoshi Tokuda	Mitsumi Electric Co., Ltd.	US 6,166,529	Section 15.4.4 of Koza et al. (2003*b*)
Low-voltage balun circuit	2001	Sang Gug Lee	Information and Communications University	US 6,265,908	Section 15.4.1 of Koza et al. (2003*b*)
High-current load circuit	2001	Timothy Daun–Lindberg & Michael Miller	IBM	US 6,211,726	Section 15.4.3 of Koza et al. (2003*b*)
Tunable integrated active filter	2001	Robert Irvine & Bernd Kolb	Infineon Technologies AG	US 6,225,859	Section 15.4.6 of Koza et al. (2003*b*)

> [M]ost . . . 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.

In addition, as Balkir, Dundar, and Ogrenci (2003) stated,

> The major reason underlying this lack of analog design automation tools has been the difficulty of the problem, in our opinion. Design in the analog domain requires creativity because of the large number of free parameters and the sometimes obscure interactions between them. . . . Thus, analog design has remained more of an "art" than a "science."

Genetic programming can be used to automatically create both the topology and sizing of an electrical circuit by

1. establishing a representation for electrical circuits suitable for use in a run of genetic programming, and
2. defining a fitness measure that measures how well the behavior and characteristics of a candidate circuit satisfy the problem's high-level design requirements.

The representation and fitness measure are then used during the run of genetic programming. During the run, the evaluation of the fitness of each individual in the population involves

1. converting each individual program tree in the population into the type of input accepted by a circuit simulator (i.e., a netlist listing each component in the circuit and the connections between the leads of the components),
2. obtaining the behavior of the individual circuit by simulating it, and
3. using the circuit's behavior and characteristics to calculate its fitness.

When genetic programming is used to automatically create computer programs, the programs are usually represented as program trees (i.e., rooted, point-labeled trees with ordered branches) in the style of the LISP programming language. A program tree is an *acyclic* graph. However, the structures that engineers typically desire to design are usually not acyclic graphs. In particular, electrical circuits are composed of loops (and, in fact, no dangling leads), and are therefore ordinarily represented by cyclic graphs (or hypergraphs). Genetic programming can be applied to the problem of automatic circuit synthesis by searching for a computer program (an acyclic structure) consisting of the instructions necessary to construct the circuit (a cyclic structure).

Specifically, our approach to the automatic synthesis of circuits using genetic programming employs a developmental process inspired by the principles of developmental biology, Wilson's (1987) pioneering application of developmental biology to genetic algorithms, the use of developmental genetic algorithms to evolve neural networks (Kitano, 1990), the use of developmental genetic programming (cellular encoding) to evolve neural networks (Gruau, 1992*a*, 1992*b*), and the use of developmental genetic programming to evolve Lindenmayer systems (Koza, 1993).

The developmental process here is used to transform a program tree (an acyclic graph) into a fully developed electrical circuit (a cyclic graph or hypergraph). The developmental process entails the execution of functions in a circuit-constructing program tree.

The starting point for the developmental process consists of a simple initial circuit. The initial circuit consists of an embryo and a test fixture.

The embryo herein consists of a single isolated modifiable wire that is not initially connected to external inputs or outputs. If the original modifiable wire is not modified by the developmental process, the circuit produces only trivial output. All development originates from the embryo's modifiable wire(s). An analog electrical circuit is developed by progressively applying the functions in a circuit-constructing program tree to the embryo's initial modifiable wire(s) and to succeeding modifiable wires and modifiable components.

The execution of the functions in the program tree transforms the initial circuit into a fully developed circuit. 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.

Test fixtures are commonly used in electrical engineering to measure the behavior of a circuit. When genetic programming is being used to automatically synthesize an electrical circuit, the test fixture is the entity (external to the circuit that is being automatically created) that facilitates measurement of the behavior of the fully developed circuit. The test fixture feeds external input(s) into the circuit of interest. It also enables the circuit's output(s) to be probed. The test fixture is a hard-wired structure composed of nonmodifiable wires and nonmodifiable electrical components. The test fixture has one or more ports that enable the embryo (and, later, the fully developed circuit) to be embedded into it. In turn, the embryo (and, later, the fully developed circuit) has one or more ports that enable it to communicate with the test fixture in which it is embedded. The hard-wired components of the test fixture often include a source resistor and a load resistor. The test fixture supplies the measurements that enable the fitness measure to assign a single numerical value of fitness to the behavior and char-

https://doi.org/10.1017/S089006040404017X Published online by Cambridge University Press

acteristics of the fully developed circuit. The test fixture can obtain the measurements needed by the fitness measure in various ways. One way (the way that is used for the work described in this paper) is to simulate individual candidate circuits or controllers using a general-purpose simulator, such as SPICE (Quarles et al., 1994). Another way is to create, at very high speeds, a temporary physical embodiment of each candidate circuit using a field-programmable transistor array (Stoica et al., 2001) and directly measure the circuit's behavior and characteristics. Although it would be impractical, another way would be to create a physical embodiment of each candidate circuit on a breadboard or silicon and directly measure the circuit's behavior and characteristics.

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

- topology-modifying functions (e.g., series division, parallel division, cut, via connecting two distant points in a circuit, via connecting a point in the circuit to ground, via connecting a point to a power supply, via connecting a point to an incoming signal, via connecting a point to an output port) that modify the topology of the developing circuit;
- component-creating functions that insert components (e.g., resistors, capacitors, transistors) into the developing circuit;
- development-controlling functions that control the developmental process by which the embryo and its successor circuits are converted into a fully developed circuit (e.g., the no-operation function);
- arithmetic-performing functions (e.g., addition, subtraction) that may appear in a value-setting subtree that is an argument to a component-creating function and that specifies the numerical value of the component; and
- automatically defined functions that enable certain substructures to be reused (including parameterized reuse).

The component-creating functions have value-setting subtree(s), whereas topology-modifying functions and development-controlling functions do not.

The *terminals* in the circuit-constructing program trees may include

- constant numerical values,
- perturbable numerical values,
- symbolic values (e.g., discrete alternative types for certain components),
- externally supplied free variables, and
- zero-argument functions (e.g., the development-ending function).

In applying the genetic programming algorithm (Section 2.1) to a particular problem of circuit synthesis, the first step is to generate an initial population (generation 0) of random compositions of the above functions and terminals. These functions and terminals permit the construction (through the developmental process just described) of any circuit composed of transistors, resistors, and capacitors.

Most of the randomly created circuits in generation 0 are, of course, very poor at solving the problem at hand (i.e., have very poor fitness). Many of these randomly created individuals are functionless or nonsensical. For example, many randomly created circuits in generation 0 do not make a connection to all input signals, all output ports, and all necessary power sources. Moreover, many randomly created circuits are so pathological that they cannot even be simulated by the general-purpose circuit simulator (SPICE, described in Quarles et al., 1994) used for the work herein. Nonetheless, even in this primordial ooze of randomly generated circuits, some score a better value of fitness than others. The evolutionary process builds on small differential advantages among the randomly created circuits of generation 0 (and the differential advantages of circuits in successive generations of the run of genetic programming).

In each generation, after the fitness of each individual in the population is ascertained, genetic programming probabilistically selects relatively more fit individuals from the population to participate in the problem-independent genetic operations used in generation programming (e.g., reproduction, mutation, and crossover). An important feature of Darwinian selection is that the selection is not greedy. Individuals that are known to be inferior will be selected to a certain degree. The best individual in the population in a given generation is not guaranteed to be selected. Moreover, the worst individual in the population in a given generation will not necessarily be excluded.

The genetic operations are applied to the selected individuals in the population to create a new population (i.e., a new generation) of offspring individuals. This process of executing each individual in the population to develop a circuit, evaluating the fitness of each fully developed circuit, selecting individuals from the population to participate in the genetic operations, and creating a new population is iterated over many generations.

Over successive generations, the fitness of the best individual in the population and that of the average individual in the population tend to progressively improve. At the same time, functionless, nonsensical, and nonsimulatable individuals tend to disappear as the run of genetic programming progresses. For example, the percentage of nonsimulatable individuals is sometimes as high as 90% for the randomly created individuals of generation 0. However, this percentage typically drops to low single digits after just a couple of generations (because of the effect of selecting more fit individuals to participate in the genetic operations that create the next generation).

Note that, in automatically synthesizing analog electrical circuits composed of transistors, resistors, and capacitors, genetic programming uses only a *de minimus* amount of platitudinous knowledge about analog circuits. Specifi-

https://doi.org/10.1017/S089006040404017X Published online by Cambridge University Press

cally, genetic programming employs a circuit simulator (e.g., SPICE) for the *analysis* of already created candidate circuits, but it does not use any knowledge about how to *synthesize* electrical circuits.

The runs of genetic programming for the various problems described in this paper are intentionally highly uniform. For example, the same function set, the same terminal set, and the same developmental process is used on each particular design problem described in this paper (and also in applying this process to many other problems over a period of years). The main difference between the runs is that a different fitness measure is used for each problem. Construction of a fitness measure requires translating the problem's high-level requirements into a precise computation using measurable characteristics or behavior of the circuit (obtained by means of a general-purpose circuit simulator). In particular, the fitness measure for a problem reflects the performance and characteristics of the desired circuit. The fitness measure specifies the desired time-domain or frequency-domain output value(s), given various specified input value(s). A problem-specific test fixture consisting of certain fixed components (such as an incoming signal source, a source resistor, a load resistor, and output probe points) is connected to the relevant input port(s) and the output port(s) of each candidate circuit.

3. INVENTION OF NEGATIVE FEEDBACK IN 1927

In a commencement address at Worcester Polytechnic Institute in 2000, C. Michael Armstrong, CEO of AT&T, recounted the following:

> On a sweltering summer morning in August 1927, a young man was seated on a passenger ferry as it churned across Upper New York Bay toward Manhattan. He was gazing idly at the Statue of Liberty when suddenly he jumped from his seat and began frantically searching his pockets for a scrap of paper.
>
> Coming up empty, he raced to the newsboy on deck and bought a copy of *The New York Times*. The man tore through the pages until he found one that was nearly free of type. He uncapped his fountain pen, sketched a couple of crude diagrams, and surrounded them with mathematical equations.

Holding up the now-famous page from *The New York Times* (Fig. 1), C. Michael Armstrong continued:

> When the ferryboat docked at Manhattan, he raced to his office at Bell Laboratories. He showed his diagrams and equations to one of his coworkers who read them carefully. Then his friend let out a big whoop and they both scrawled their initials on the newspaper page.
>
> The young man on the ferryboat was Harold Black, Worcester Polytechnic Institute Class of 1921. And the scribblings on his newspaper were the blueprint for the negative-feedback amplifier, a device that played a vital role in 20th century electronics.

Referring to the scribblings on this newspaper page, Mervin Kelly, then president of Bell Labs, said the following in 1957 (Black, 1977):

> Although many of Harold's inventions have made great impact, that of the negative feedback amplifier is indeed the most outstanding. It easily ranks coordinate with De Forest's invention of the audion as one of the two inventions of broadest scope and significance in electronics and communications of the past 50 years . . . It is no exaggeration to say that without Black's invention, the present long-distance telephone and television networks which cover our entire country and the transoceanic telephone cables would not exist. The application of Black's principle of negative feedback has not been limited to telecommunications. . . . [T]he entire explosive extension of the area of control, both electrical and mechanical, grew out of an understanding of the feedback principle.

3.1. The once universal idiom of positive feedback

In *The Design of CMOS Radio-Frequency Integrated Circuits*, Lee (1998) recounts the history that predated Black's 1927 invention of negative feedback.

Early work on feedback in amplifiers employed positive feedback. This work included rocket pioneer Robert Goddard's 1915 patent for a vacuum tube oscillator using positive feedback (Goddard, 1915) and Edwin Howard Armstrong's 1914 patent on amplifiers, again using positive feedback (Armstrong, 1914). As Lee (1998) observes,

> [P]rogress in electronics in those early years was largely made possible by Armstrong's regenerative [positive feedback] amplifier, since there was no other economical way to obtain large amounts of gain from the primitive (and expensive) vacuum tubes of the day. . . .
>
> Armstrong was able to get gain from a single stage that others could obtain only by cascading several. This achievement allowed the construction of relatively inexpensive, high-gain receivers and therefore also enabled dramatic reductions in transmitter power because of the enhanced sensitivity provided by this increased gain. In short order, *the positive feedback (regenerative) amplifier became a nearly universal idiom*, and Westinghouse (to whom Armstrong had assigned patent rights) kept its legal staff quite busy trying to make sure that only licensees were using this revolutionary technology. [emphasis added]

However, Westinghouse's "nearly universal idiom" did not solve a major problem facing AT&T at the time, namely, distortion in amplifiers. As Lee (1998, p. 387) points out,

https://doi.org/10.1017/S089006040404017X Published online by Cambridge University Press

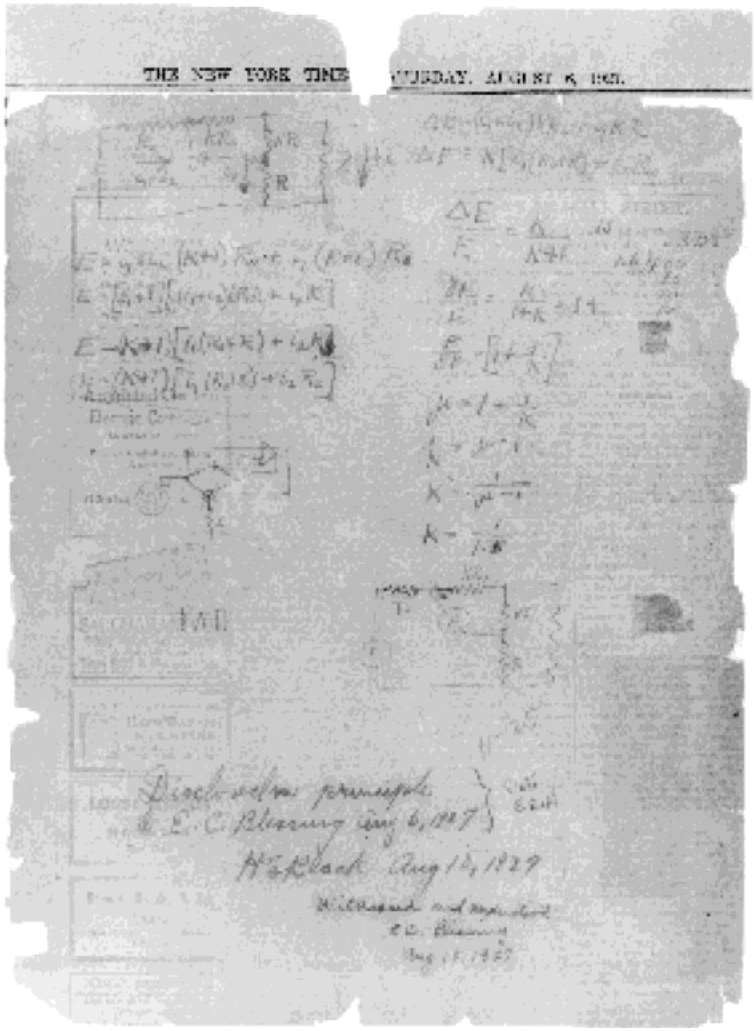

Fig. 1. Notes written by Harold S. Black on a page of *The New York Times* while commuting on the Lackawanna Ferry in 1927. Reprinted with permission of Lucent Technologies Inc./Bell Labs Inc.

Although Armstrong's regenerative amplifier pretty much solved the problem of obtaining large amounts of gain from vacuum tube amplifiers, a different problem preoccupied the telephone industry. In trying to extend communications distances, amplifiers were needed to compensate for transmission-line attenuation. Using amplifiers available in those early days, distances of a few hundred miles were routinely achievable and, with great care, perhaps 1,000–2,000 miles was possible, but the quality was poor. . . .

The problem wasn't one of insufficient amplification; it was trivial to make the signal at the end of the line quite loud. Rather the problem was distortion. Each amplifier contributed some small (say, 1%) distortion. Cascading a hundred of these things guaranteed that what came out didn't very much resemble what went in.

The main "solution" at the time was to (try to) guarantee "small signal" operation of the amplifiers. That is, by restricting the dynamic range of the signals to a tiny fraction of the amplifier's overall capability, more linear operation could be achieved. Unfortunately, this strategy is quite inefficient since it requires the construction of, say, 100-W[att] amplifiers to process milliwatt signals. Because of the arbitrary distance between a signal source and an amplifier (or possibly between amplifiers), though, it was difficult to guarantee that the input signals were always sufficiently small to satisfy linearity.

3.2. Black's first solution in 1923

Such was the state of affairs when Harold S. Black started working in 1921 at AT&T on the problem of reducing amplifier distortion.

As will be seen below, Black solved the problem twice. Black's first solution (described in this subsection) was physically implemented, but was ultimately regarded as impractical. Black's second solution (described in the next

https://doi.org/10.1017/S089006040404017X Published online by Cambridge University Press

subsection) is now regarded as one of the most important inventions of the 20th century in electrical engineering.

Both of Black's solutions were characterized by a logical discontinuity and a singular moment when the previous thinking about this vexatious and long-standing problem was, in a "flash of genius," overthrown and replaced by a new approach that could not have been logically deduced from what was previously known.

After 2 years of work on the problem, Black had tentatively reached the conclusion in 1923, "There was just no way to meet our ambitious goal" (Black, 1977). As Black then recounts,

> This might have been the end of it, except that, on March 16, 1923, I was fortunate enough to attend a lecture by the famous scientist and engineer, Charles Proteus Steinmetz. . . .
>
> I no longer remember the subject, but . . . I was so impressed by how Steinmetz got down to the fundamentals that when I returned home at 2 A.M., I restated my own problems as follows: Remove all distortion products from the amplifier output. In doing this, I was accepting an imperfect amplifier and regarding its output as composed of what was wanted plus what was not wanted. I considered what was not wanted to be distortion (regardless of whether it was due to nonlinearity, variation in the tube gain, or whatever), and I asked myself how to isolate and then eliminate this distortion. I immediately observed that by reducing the output to the same amplitude as the input, and subtracting one from the other, only the distortion would remain. This distortion could then be amplified in a separate amplifier and used to cancel out the distortion in the original amplifier. . . .
>
> The next day, March 17, I sketched two such embodiments and thereby invented the feed-forward amplifier. . . .
>
> Later that day, I set up each embodiment in the laboratory. Both worked as expected.

Unfortunately, Black's 1923 invention did not turn out to be practical. As Black (1977) laments,

> [T]he invention required precise balances and subtractions that were hard to achieve and maintain with the amplifiers available at that time. . . .
>
> Over the next four years, I struggled with the problem of turning my intention into an amplifier that was practical. . . .
>
> [F]or my purpose the gain had to be absolutely perfect.
>
> For example, every hour on the hour—24 hours a day—somebody had to adjust the filament current to its correct value. . . .
>
> In addition, every six hours it became necessary to adjust the B battery voltage, because the amplifier gain would get out of hand.
>
> There were other complications too, but these were enough!

The bottom line concerning the feed-forward amplifier that Black invented in 1923 was "Nothing came of my efforts, however, because every circuit I devised turned out to be far too complex to be practical."

3.3. Black's second solution in 1927: The flash of genius on the Lackawanna Ferry

Despite this false start, Black continued to work on the problem of reducing distortion in amplifiers for several additional years.

After working on the problem for a total of 6 years, Black (1977) recounted the following:

> 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. I signed the sketch, and 20 minutes later, when I reached the laboratory at 463 West Street, it was witnessed, understood, and signed by the late Earl C. Blessing.

As previously mentioned, Edwin Howard Armstrong's approach to amplification using positive feedback was "a nearly universal idiom" during the early part of the 20th century. Black's approach did not flow logically from Armstrong's approach or from existing knowledge. Instead, Black's approach represented a logical break from prevailing thinking.

Despite the elegance and demonstrated practical effectiveness of negative feedback, Armstrong's approach was so entrenched in the thinking of electrical engineers that there was widespread resistance to Black's concept of negative feedback for many years after its invention. As Black (1977) recalls,

> Although the invention had been submitted to the US Patent Office on August 8, 1928, more than nine years would elapse before the patent was issued on December 21, 1937. . . . One reason for the delay was that *the con-*

https://doi.org/10.1017/S089006040404017X Published online by Cambridge University Press

cept was so contrary to established beliefs. [emphasis added]

The British Patent Office was even more resistant. Black (1977) recounted ". . . our patent application was treated in the same manner as one for a perpetual motion machine."

The British Patent Office continued to maintain that negative feedback would not work despite the fact that AT&T had "70 amplifiers working successfully in the telephone building at Morristown" for a number of years.

We believe that one reason why it took an inordinate amount of time for negative feedback to gain acceptance was that human thinking often becomes channeled along the well-traveled paths of "established beliefs." In this situation, logical thinking can become the enemy of creativity.

Of course, when we say that the invention process is inherently illogical, we do not mean that inventors are oblivious to logic or that logical thinking is not helpful to inventors. Logical thinking often plays the important role of setting the stage for an invention. As Black (1977) himself noted, "[s]everal years of hard work on the problem" brought his thinking into the proximity of a solution. However, Black was not able to arrive at the invention by means of logic. Then, at the critical moment, Black made the necessary illogical leap during his now famous ferryboat ride. Although logical thinking may play a role in invention and creativity, at the end of the day, the critical element is a logical discontinuity from established ideas.

Many inventions over the years (like Black's two solutions to the problem of reducing distortion in amplifiers) were conceived in a singular moment when the prevailing thinking was replaced by a new approach that could not have been logically deduced from what was previously known.

3.4. Reinvention of negative feedback by genetic programming

In this section, we will show how the once perplexing problem of designing an electrical circuit to reduce distortion in amplifiers can be readily solved in an automated way by means of genetic programming. As will be seen below, Black's solution involving negative feedback readily flows from an evolutionary search guided by a high-level statement of the problem.

Before proceeding, note that one difference between Black's solution in 1927 and our present-day run of genetic programming is that Black invented negative feedback in the era of vacuum tubes. We did not have access to accurate models for the vacuum tubes actually used by Black. However, his patents state levels of performance that he achieved with his inventions. Present-day transistors operate in a manner similar to vacuum tubes. In particular, field-effect transistors (FETs) closely resemble the behavior of vacuum tubes in that they are voltage controlled. Therefore, the transistor-inserting function that we used in our present-day run of genetic programming inserts an IRFZ44 FET into the developing circuit. We maintained substantial consistency with the performance levels obtained by Black in 1927 by using an initial circuit with a voltage source supplying an incoming sine wave with a 4-V amplitude, a 50-ohm source resistor, a 100-ohm load resistor, and a 60-V power supply.

3.4.1. Preparatory steps

The human user communicates the high-level statement of the problem to the genetic programming system by performing certain well-defined preparatory steps.

The five major preparatory steps for the basic version of genetic programming require the human user to specify the following:

1. the set of terminals (e.g., the independent variables of the problem, zero-argument functions, and random constants) for each branch of the program that is to be evolved,
2. the set of primitive functions for each branch of the program that is to be evolved,
3. the fitness measure (for measuring the fitness of individuals in the population),
4. certain parameters for controlling the run, and
5. the termination criterion and method for designating the result of the run.

As previously mentioned, we approach each problem of synthesis of analog electrical circuits (including the eight specific problems discussed in this paper) in substantially the same way. We employ the generic set of functions described in Section 2.3 of this paper (e.g., generic component-creating functions that insert resistors, capacitors, and transistors; generic topology-modifying functions; generic development-controlling functions) and we employ the generic set of terminals described in Section 2.3 (e.g., constant numerical values, perturbable numerical values, symbolic values, zero-argument development-ending functions).

In preparing a run of genetic programming on a particular problem (e.g., the problem of reducing distortion in amplifiers), the emphasis is on the fitness measure. The construction of a problem's fitness measure requires the human user to identify exactly what is wanted. Focusing on what Black wanted leads to a multiobjective fitness measure based on the degree to which a candidate electrical circuit

- amplifies the incoming signal,
- minimizes distortion, and
- minimizes the number of expensive components.

Suppose that we adopt the usual convention that a smaller value of fitness is better, with zero being the (perhaps unattainable) ideal value. Suppose also that the specified amount of amplification is 10 dB. In that event, if a circuit were to receive a perfect sine wave as its input, the desired output of the circuit would be an inverted perfect sine wave

https://doi.org/10.1017/S089006040404017X Published online by Cambridge University Press

whose amplitude is 3.16 times that of the incoming signal. The fitness measure for a candidate circuit can thus be based on the difference between this desired waveform and the output actually produced by the candidate circuit.

We first consider the first element of the three-element fitness measure. If the average absolute difference between a candidate circuit's output and the desired output is small (say, $<1\%$), the circuit can be deemed to deliver a satisfactory amount of amplification and this first element of the fitness measure can be set to 0. Otherwise, the first element of the fitness measure is set to the average absolute difference between the desired output and the actual output.

The second element of the fitness measure is based on total harmonic distortion (THD):

$$\text{THD} = \frac{\sqrt{\sum_{i=2}^{N} A_i^2}}{A_1},$$

where A_1 is the magnitude of the first harmonic (i.e., the fundamental frequency) and A_i is the magnitude of the ith harmonic (Vladimirescu, 1994). For audio signals of interest over a telephone, it would be reasonable to choose 1000 Hz as the fundamental frequency and to consider $N = 9$ harmonics. If the total harmonic distortion is less than, say, -45 dB, a circuit can be deemed to be satisfactory in terms of reducing distortion and the second element of the fitness measure can be set to 0. Otherwise, the second element is equal to

$$10 \times (1 + |\text{THD} - (-45)|).$$

The third element of the fitness measure assigns a cost of 1.0 for each transistor and 0.01 for each resistor and capacitor (a choice suggestive of the cost difference in the 1920s of a vacuum tube and a resistor).

The three required elements of this multiobjective problem can be combined in a lexicographic way as follows: fitness can be defined as the sum of the first element (amplification), 10^{-6} times the second element (distortion), and 10^{-18} times the third element (parsimony). This lexicographic scheme causes the search process to first look for a circuit that provides the desired amplification. Once the contribution of the first element (amplification) reaches zero, the second element's contribution remains significant and distortion is taken into account. Once the contribution of the second element (distortion) reaches zero, the third element's contribution remains significant and parsimony is taken into account. This multiobjective fitness measure provides a way to take each of the three required elements into account and to readily compare the performance (desirability) of one candidate circuit to another.

Note that this fitness measure was created by focusing the problem's high-level requirements: amplification, distortion, and parsimony. The fitness measure is concerned with "what needs to be done," not with "how to do it." Notice that the fitness measure does not mandate a feed-forward approach (i.e., Armstrong's well-established approach), a feedback approach, or any other particular approach. In addition, if feedback is used at all, the fitness measure is agnostic as to whether the feedback is Armstrong's positive type of feedback or Black's negative type of feedback.

The remaining preparatory steps for Black's problem of designing a circuit to reduce distortion in amplifiers are identical to those for numerous other problems of circuit synthesis that have been solved using genetic programming. For details, see Koza, Keane, Streeter, Mydlowec, Yu, and Lanza (2003).

An important point about the just-described preparatory steps supplied by the human user prior to a run of genetic programming is that genetic programming requires only a *de minimus* amount of platitudinous knowledge about analog circuits. No knowledge base concerning the synthesis of analog electrical circuits is used. (Indeed, we know of no knowledge-based approach that is capable of automatically synthesizing the topology and sizing of analog circuits from a high-level statement of design requirements). Instead, we use a generic set of primitive functions (capable of creating any circuit composed of resistors, capacitors, and transistors), a generic set of terminals, and a fitness measure that expresses the high-level design requirements of the problem at hand. In the present instance, we used Black's high-level requirements (amplification, distortion, and parsimony) to specify "what needs to be done."

3.4.2. Results for the problem of reducing amplifier distortion

The best circuit from among the 1,000,000 individuals in the population at generation 0 delivers amplification of -2.91 dB. That is, the best-of-generation circuit acts as an attenuator rather than an amplifier. However, even amplification of -2.91 dB can provide a toe-hold for the evolutionary process aimed at creating an amplifier.

The first best-of-generation circuit that acts as an amplifier appeared in generation 9. Specifically, this individual acts as a 5.37-dB amplifier. In addition to having inadequate amplification, this circuit is unsuitable in that it has a total harmonic distortion of -5.65 dB.

The first circuit in the run satisfying the problem's amplification criterion (10 dB) and having a total harmonic distortion of less than -45 dB appeared in generation 46. This circuit has a total harmonic distortion of -54.2 dB.

This best-of-generation circuit from generation 46 (Fig. 2) consists of three FETs and two resistors (ignoring the source resistor RSRC and the load resistor RLOAD that were hard wired into the test fixture). In viewing the figures, note that a FET's source corresponds to a vacuum tube's cathode, the FET's drain corresponds to the tube's anode, and the FET's gate corresponds to the tube's grid. In this circuit, transistor Q3 is biased off so its removal does not affect the circuit's behavior. However, transistors Q1 and Q2 are overlaid in

https://doi.org/10.1017/S089006040404017X Published online by Cambridge University Press

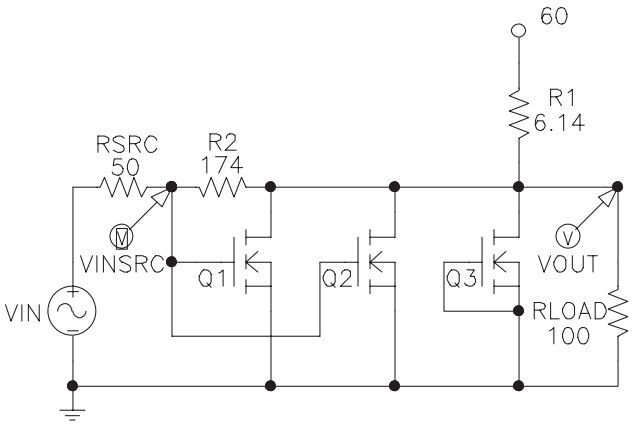

Fig. 2. The best-of-generation circuit from generation 46 for the problem of reducing amplifier distortion.

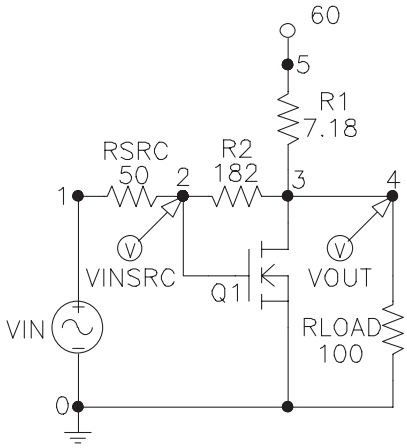

Fig. 3. The parsimonious best-of-run circuit from generation 48 for the problem of reducing amplifier distortion.

parallel and together act as a single transistor with twice the transconductance and interelectrode capacitance of Q1. The removal of one of these two identical transistors slightly decreases the circuit's amplification (to 9.64 dB), adversely affects its total harmonic distortion (to −51.9 dB), and changes its bias.

The 174-ohm resistor R2 in Figure 2 is the mechanism for providing negative feedback from Q1 and Q2. Because Q1 and Q2 reverse the phase of the incoming signal, the feedback is negative; that is, the signal from the drains (corresponding to a vacuum tube's plate) of Q1 and Q2 is subtracted from the incoming signal at the point-labeled VINSRC.

Table 2 shows, in column 2, the amplitude (decibels) of the fundamental frequency (1000 Hz) and various harmonics for the best-of-generation circuit from generation 46 (Fig. 2). Column 3 shows the amplitude (decibels) with respect to the fundamental frequency.

The best-of-run circuit (Fig. 3) emerged in generation 48. The average absolute error (measuring amplification) of this best-of-run circuit from generation 48 is 0.065 V (about two-thirds of that of the best-of-generation circuit from generation 46). The best-of-run circuit has amplification of 10.06 dB and total harmonic distortion of −51.2 dB. This circuit is more parsimonious than the best-of-generation circuit from generation 46 in that it has only one transistor and two resistors (again ignoring RSRC and RLOAD). As can be seen, the 182-ohm resistor R2 is the mechanism for providing the negative feedback from the drain of transistor Q1 to the point-labeled VINSRC.

Table 3 shows, in column 2, the amplitude (decibels) of the fundamental frequency (1000 Hz) and various harmonics for the best-of-run circuit from generation 48 (Fig. 3). Column 3 shows the amplitude (decibels) with respect to the fundamental frequency.

Thus, after 48 generations, genetic programming succeeded in recreating the invention of negative feedback. In reinventing negative feedback, the genetic programming algorithm did not rely on logic. Instead, it conducted a probabilistic search in the space of circuit-constructing computer programs. This probabilistic search was guided by the

Table 2. *Distortion for the best-of-generation circuit from generation 46 for the problem of reducing amplifier distortion*

Harmonic	Amplitude	Amplitude wrt Fund. Freq.
1000 (fund. freq.)	9.96	0
2000	−44.3	−54.3
3000	−61.5	−71.5
4000	−68.7	−78.7
5000	−71.0	−81.0
6000	−78.8	−88.8
7000	−85.6	−95.6
8000	−81.6	−91.6
9000	−81.4	−91.4

Table 3. *Distortion of the best-of-run circuit from generation 48 for the problem of reducing amplifier distortion*

Harmonic	Amplitude	Amplitude wrt Fund. Freq.
1000 (fund. freq.)	10.06	0
2000	−41.3	−51.4
3000	−54.7	−64.8
4000	−73.2	−83.3
5000	−66.9	−77.0
6000	−68.8	−18.9
7000	−73.2	−83.3
8000	−69.9	−80.0
9000	−73.3	−83.4

https://doi.org/10.1017/S089006040404017X Published online by Cambridge University Press

high-level requirements of Black's problem of minimizing distortion in amplifiers. In addition, this probabilistic search process arrived at the same creative solution that Black conceived in 1927.

Black received US patents 2,003,282 (Black, 1935), 2,102,670 (Black, 1937*a*), and 2,102,671 (Black, 1937*b*) that relate to his work on the problem of reducing distortion in amplifiers, as well as US patent 1,686,792 (Black, 1928,) for the earlier impractical solution described in Section 1.2. The overall goal of all these efforts is stated in the description of US patent 2,102,670 (Black, 1937*a*):

> It is common experience that increase of the power output of vacuum tubes or electric space discharge devices tends to increase distortion of signaling or other waves transmitted by the devices, and tends to lower the gain of the circuits of the devices. . . .
>
> Therefore, a major problem in devising vacuum tube systems, as for example vacuum tube amplifier systems, is the securing of high output of power without attendant disadvantages, as for example without increase of first cost or decrease of operating efficiency of the systems, and especially in the case of vacuum tube amplifiers and repeaters, without sacrifice of quality of signal reproduction.

The best-of-run circuit from generation 48 (Fig. 3) infringes claims 1 and 3 of US patent 2,102,671 (Black, 1937*b*). Claim 1 covers,

> In a wave translating device or system having amplifying properties, an input portion and an output portion, means to apply fundamental waves to said input portion, said system carrying fundamental components in said output portion, and having means producing other wave components in said output portion, and means controlling the relative magnitudes of said components in said output portion comprising means to feed waves from said output portion to said input portion to decrease the gain of the system.

The FET Q1 is the "wave translating device or system." The incoming voltage signal source is the "means to apply fundamental waves to said input portion." Resistor R2 is the "means to feed waves from said output portion to said input portion to decrease the gain of the system." The negative feedback "decrease[s] the gain of the system."

Claim 3 of US patent 2,102,671 (Black, 1937*b*) covers the following:

> In a wave translating system operating to amplify applied fundamental waves, and to produce distortion components as a function of nonlinearity in the system, means to increase the ratio of the amplified fundamental wave component to distortion components comprising means to utilize a portion of the waves translated by said system to reduce the gain of the system below the gain with zero feedback in the system of the waves translated by the system.

Thus, we have seen that if one begins with a high-level statement of the problem that Black was trying to solve, Black's solution readily flows from a run of genetic programming. It does so because Black's solution is a correct solution to the problem of reducing distortion in amplifiers and, as they say, necessity is the mother of invention.

One of the virtues of genetic programming is that it approaches a problem in an open-ended way that is not encumbered by previous human thinking. Genetic programming is not aware, much less concerned, about whether a solution is "contrary to established beliefs." For this reason, genetic programming often unearths solutions that might have never occurred to human scientists and engineers who are steeped in the thinking of the day.

4. REINVENTION OF A SALLEN–KEY FILTER BY GENETIC PROGRAMMING

Genetic programming can routinely produce inventive and creative results. We illustrate this point in this section by considering the problem of automatically synthesizing a circuit that duplicates the behavior of another important invention in the field of electrical engineering, namely, the Sallen–Key active filter composed of op amps (Kardontchik, 1992; Lancaster, 1995; Karki, 1999).

4.1. Preparatory steps

As previously mentioned, we approach each problem of analog circuit synthesis in substantially the same way. We use the generic set of functions described in Section 2.3 (e.g., generic component-creating functions that insert resistors, capacitors, and transistors; generic topology-modifying functions; and generic development-controlling functions) and the generic set of terminals described in Section 2.3 (e.g., constant numerical values, perturbable numerical values, symbolic values, zero-argument development-ending functions). Thus, in preparing for a run of genetic programming on the problem of synthesizing a Sallen–Key filter, the emphasis is on the fitness measure.

The fitness measure consists of 10 elements. The 10 elements are divided into two groups, each consisting of 5 elements. Within each group of 5 elements, the first element of each group evaluates the circuit in the frequency domain with the aim of getting a circuit whose frequency-domain behavior matches that of the model circuit whose behavior represents that of a Sallen–Key filter. For the second and third elements of each group, the circuit is evaluated in the passband with the aim of getting its output to match the incoming waveform as closely as possible. For the fourth and fifth elements within each group, the circuit is evaluated in the stopband with the aim of getting its

https://doi.org/10.1017/S089006040404017X Published online by Cambridge University Press

output to be suppressed to nearly zero (without regard to the shape of the suppressed waveform).

For all elements of the fitness measure, all internal points of the circuit are probed. If any voltage at any point in the circuit ever falls outside the interval $[-30, +30]$, the individual receives a high-penalty fitness (e.g., 10^8). An individual also receives a high-penalty fitness if the circuit cannot be simulated. In the absence of these two extreme situations, fitness is a weighted sum of the *performance penalty* and the *parsimony penalty*. The performance penalty is the sum, over all 10 elements of the fitness measure, of the detrimental contribution to fitness associated with that element of the fitness measure. The parsimony penalty is equal to the number of op amps in the circuit plus 0.25 times the number of other components in the circuit.

The performance penalty and the parsimony penalty can be combined into a multiobjective fitness measure in a lexicographic way similar to that used in the previous section of this paper. During the early part of the run, the emphasis is on performance. After a satisfactory level of performance is achieved, the emphasis is on parsimony.

4.2. Results

The best-of-run individual (Fig. 4) was produced at generation 183.

Figure 5 shows the behavior, in the frequency domain, of the genetically evolved best-of-run circuit from generation 183. The attenuation for the decade of frequency between 1000 and 10,000 Hz is 39.1 dB. The frequency response of this circuit indicates that it is a satisfactory solution to the problem. Examination of the topology of the circuit (Fig. 4) reveals that the genetically evolved solution is a Sallen–Key filter.

5. REINVENTION OF SIX 21ST-CENTURY PATENTED INVENTIONS BY GENETIC PROGRAMMING

We further illustrate the point that genetic programming can routinely produce inventive and creative results by considering six instances where genetic programming automatically created both the topology and sizing for analog electrical circuits patented after January 1, 2000. The six 21st-century patented inventions are shown as the last six entries in Table 1.

The main difference between the runs of genetic programming for the six problems described in this section is that we supplied a different fitness measure for each problem. Construction of a fitness measure requires translating the problem's high-level requirements into a precise computation. For each problem, the original patent document was used to determine the performance the invention was supposed to achieve.

The commercially common 2N3904 (*npn*) and 2N3906 (*pnp*) transistor models were used for the problems in this section, unless the patent document called for a different model. Five-volt power supplies were used, unless the patent specified otherwise. The control parameters and termination criterion were the same for all six problems, except that different population sizes were used on certain problems so as to approximately equalize each run's elapsed time per generation.

We now describe the six fitness measures. For additional details, see Koza, Keane, Streeter, Mydlowec, Yu, and Lanza (2003).

5.1. Fitness measures for the six 21st-century patented inventions

5.1.1. Low-voltage balun circuit

The purpose of a balun (balance/unbalance) circuit is to produce two outputs from a single input, each output having half the amplitude of the input, one output being in phase with the input, while the other is 180° out of phase with the input, with both outputs having the same DC offset. The patented balun circuit uses a power supply of only 1 V. The fitness measure consisted of a frequency sweep analysis designed to ensure the correct magnitude and phase at the two outputs of the circuit and a Fourier analysis designed to penalize harmonic distortion.

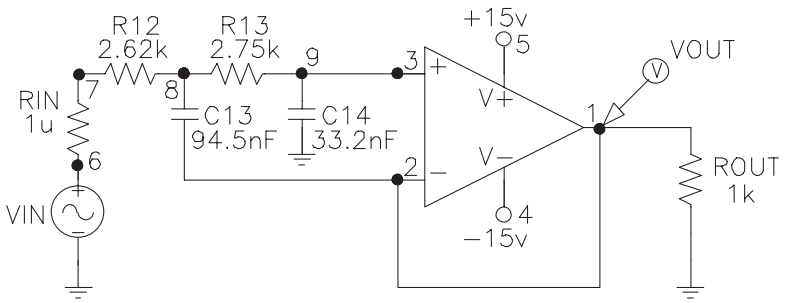

Fig. 4. The genetically evolved best-of-run circuit from generation 183.

https://doi.org/10.1017/S089006040404017X Published online by Cambridge University Press

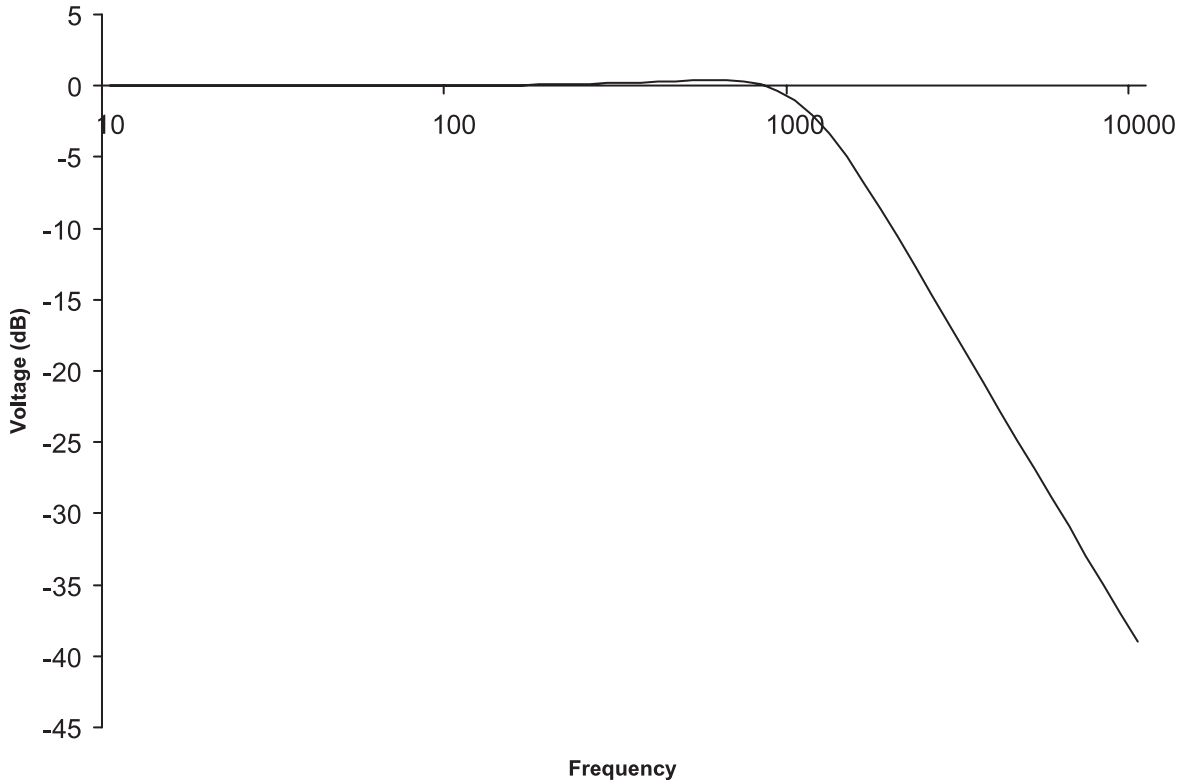

Fig. 5. The behavior in the frequency domain of the genetically evolved best-of-run circuit from generation 183.

5.1.2. Mixed analog–digital register-controlled variable capacitor

This mixed analog–digital circuit (Aytur, 2000) has a capacitance that is controlled by the value stored in a digital register. The fitness measure employed 16 time-domain fitness cases. The 16 fitness cases ranged over all eight possible values of a 3-bit digital register for two different analog input signals.

5.1.3. Voltage–current conversion circuit

The purpose of the voltage–current conversion circuit (Ikeuchi & Tokuda, 2000) is to take two voltages as input and to produce a stable current whose magnitude is proportional to the difference of the voltages. The fitness measure utilized four time-domain input signals (fitness cases). A time-varying voltage source was placed beneath the output probe point to ensure that the output current produced by the circuit was stable with respect to any subsequent circuitry to which the output of the circuit might be attached.

5.1.4. High-current load circuit

The patent covers a circuit designed to sink a time-varying amount of current in response to a control signal. The patented circuit employs a number of FETs arranged in parallel, each of which sinks a small amount of the desired current. The fitness measure comprised two time-domain simulations, each representing a different control signal.

5.1.5. Low-voltage cubic signal generator

The patent covers an analog computational circuit that produces the cube of an input signal as its output. The circuit is "compact" in that it contains a voltage drop across no more than two transistors. The fitness measure for this problem consisted of four time-domain fitness cases using various input signals and time scales. The compactness constraint was enforced by providing only a 2-V power supply.

5.1.6. Tunable integrated active filter

The patent (Irvine & Kolb, 2001) covers a tunable integrated active filter that performs the function of a low-pass filter whose passband boundary is dynamically specified by a control signal. The circuit has two inputs: a to-be-filtered incoming signal and a control signal.

The fitness measure for this problem consisted of a performance penalty and a parsimony penalty. The passband boundary (f) ranges over nine values between 441 and 4414 Hz. The performance penalty is a weighted sum, over 61 frequencies for each of the nine values of f, of the absolute weighted deviation between the output of the individ-

https://doi.org/10.1017/S089006040404017X Published online by Cambridge University Press

ual candidate circuit at its probe point and the target output. The parsimony penalty is equal to the number of components in the circuit.

5.2. Results for the six 21st-century patented inventions

5.2.1. Low-voltage balun circuit

Genetic programming automatically created the circuit shown in Figure 6. This best-of-run evolved circuit was produced in generation 97 and has a fitness of 0.429. The patented circuit has a fitness of 1.72. That is, the evolved circuit is roughly a fourfold improvement (less being better) over the patented circuit in terms of our fitness measure. In addition, the evolved circuit is superior to the patented circuit both in terms of its frequency response and its harmonic distortion.

In the patent documents, Lee (2001) shows a previously known conventional (prior art) balun circuit (Fig. 7).

Lee's (2001) patented low-voltage balun circuit is shown in Figure 8. Lee states that the essential difference between the prior art and his invention is a coupling capacitor (C_2) located between the base and the collector of the transistor (Q_2). Lee explains the essence of his invention as follows:

> The structure of the inventive balun circuit [Fig. 8 of this paper] is identical to that of [Fig. 7 of this paper] except that a capacitor C_2 are further provided thereto. The capacitor C_2 is a coupling capacitor disposed between the base and the collector of the transistor Q_2 and serves to block DC components that may be fed to the base of the transistor Q_2 from the collector of the transistor Q_2.

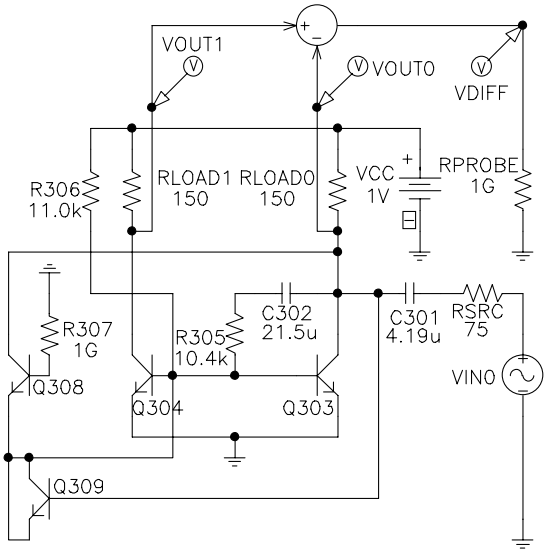

Fig. 6. An evolved balun circuit.

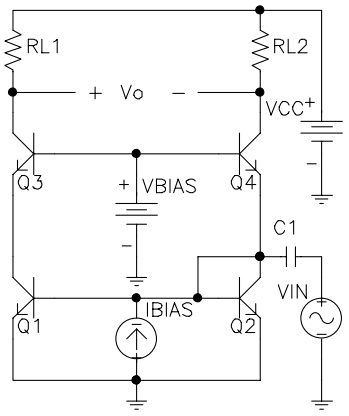

Fig. 7. A prior art balun circuit shown in US patent 6,265,908 (Lee, 2001).

As can be seen, the best-of-run genetically evolved circuit (Fig. 6) possesses the very feature that Lee identifies as the essence of his invention, namely, the coupling capacitor C_{302} in Figure 6 and C_2 in Figure 8.

The genetically evolved circuit also possesses three additional elements of claim 1 of Lee's 2001 patent. Interestingly, although the genetically evolved circuit embodies the essence of Lee's invention, it does not infringe Lee's patent because it does not possess the remaining elements of claim 1.

5.2.2. Mixed analog–digital register-controlled variable capacitor

Over our 16 fitness cases, the patented circuit has an average error of 0.803 mV. In generation 95, a circuit emerged with average error of 0.808 mV, or approximately 100.6% of the average error of the patented circuit. During the course of this run, we harvested the smallest individuals produced on each processing node with a certain maximum

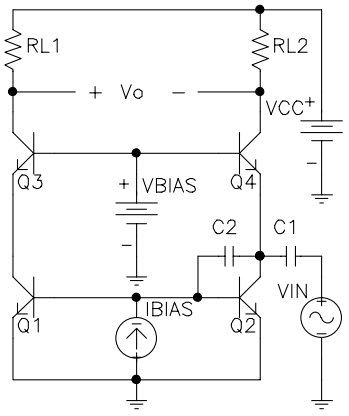

Fig. 8. The Lee (2001) low-voltage balun circuit shown in US patent 6,265,908.

https://doi.org/10.1017/S089006040404017X Published online by Cambridge University Press

level of error. Examination of these harvested individuals revealed a circuit from generation 98 (Fig. 9) that approximately matches the topology of the patented circuit (without infringing). The genetically evolved circuit possesses all but one of the elements of claim 1 of the patented circuit (and hence does not infringe the patent).

5.2.3. Voltage–current conversion circuit

A circuit emerged on generation 109 of our run of this problem with a fitness of 0.619 (compared to 1.0 for the patent circuit) That is, the evolved circuit has roughly 62% of the average (weighted) error of the patented circuit. The evolved circuit was subsequently tested on unseen fitness cases that were not part of the fitness measure and outperformed the patented circuit on these new fitness cases. The best-of-run circuit solves the problem in a different manner than the patented circuit.

5.2.4. High-current load circuit

On generation 114, a circuit emerged that duplicated Daun–Lindberg and Miller's (2000) parallel FET structure. This circuit has a fitness (weighted error) of 1.82, or 182% of the weighted error for the patented circuit.

The genetically evolved circuit shares the following feature found in claim 1 of US patent 6,211,726 (Daun–Lindberg & Miller, 2000): ". . . a plurality of high-current transistors having source-to-drain paths connected in parallel between a pair of terminals and a test load."

However, the remaining elements of claim 1 in US patent 6,211,726 (Daun–Lindberg & Miller, 2000) are very specific and the genetically evolved circuit does not read on these remaining elements. In fact, the remaining elements of the genetically evolved circuit bear hardly any resemblance to the patented circuit. In this instance, genetic programming produced a circuit that duplicates the functionality of the patented circuit using a different structure.

5.2.5. Low-voltage cubic signal generator

The best-of-run evolved circuit (Fig. 10) was produced in generation 182 and has an average error of 4.02 mV. The patented circuit had an average error of 6.76 mV. That is, the evolved circuit has approximately 59% of the error of the patented circuit over our four fitness cases.

The claims in US patent 6,160,427 (Cipriani & Takeshian, 2000) amount to a very specific description of the patented circuit. The genetically evolved circuit does not read on these claims and, in fact, bears hardly any resemblance to the patented circuit. In this instance, genetic programming again produced a circuit that duplicates the functionality of the patented circuit with a very different structure.

5.2.6. Tunable integrated active filter

Averaged over the nine values of frequency, the best-of-run circuit from generation 50 (Fig. 11) has 72.7-mV average absolute error for frequencies in the passband and

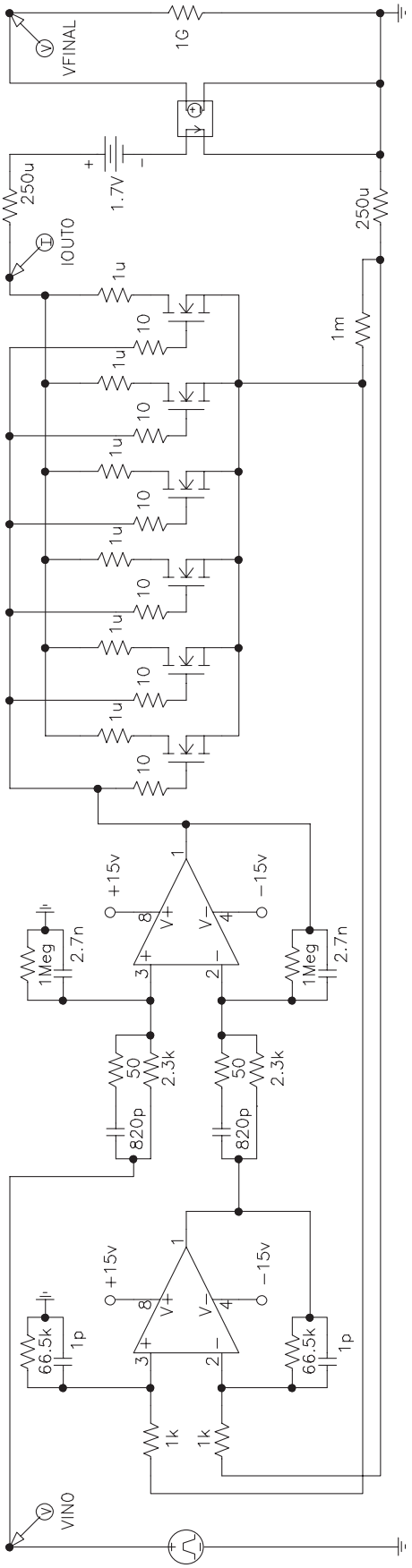

Fig. 9. An evolved compliant register-controlled capacitor circuit.

https://doi.org/10.1017/S089006040404017X Published online by Cambridge University Press

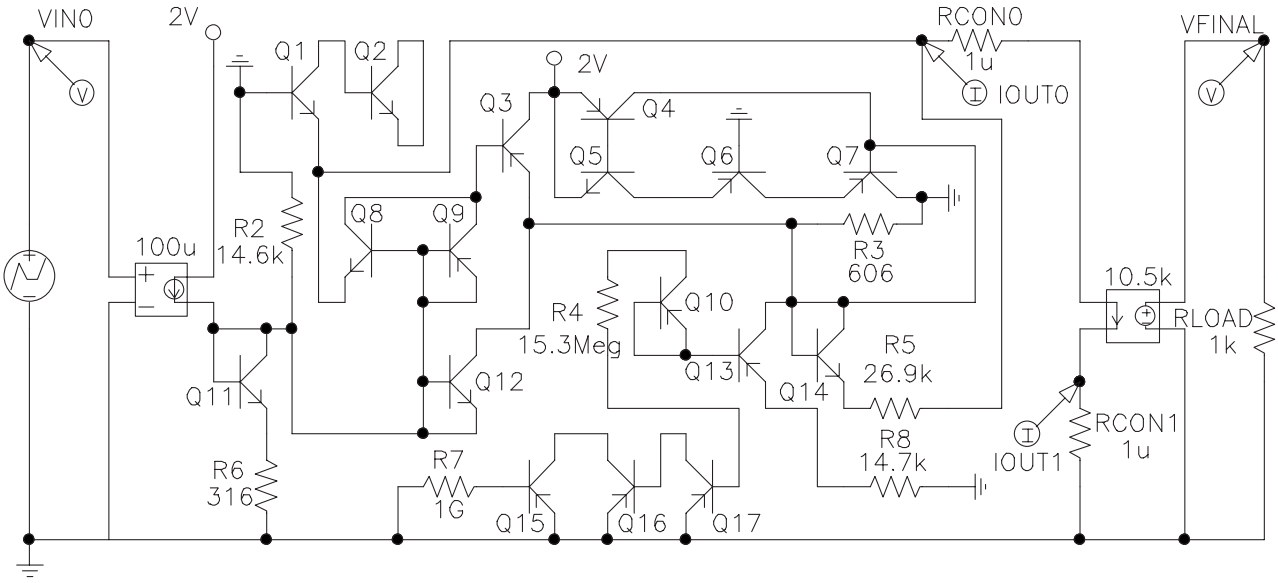

Fig. 10. An evolved cubic signal generation circuit.

0.39-dB average absolute error for frequencies to the right of the passband.

The best-of-run genetically evolved circuit possesses every element of claim 1 of US patent 6,225,859 (Irvine & Kolb, 2001) and therefore infringes the patent.

6. PATENTABLE NEW INVENTIONS CREATED BY GENETIC PROGRAMMING

Given that genetic programming has solved problems whose solutions were previously patented, it is a natural extension to try to use genetic programming to generate patentable new inventions.

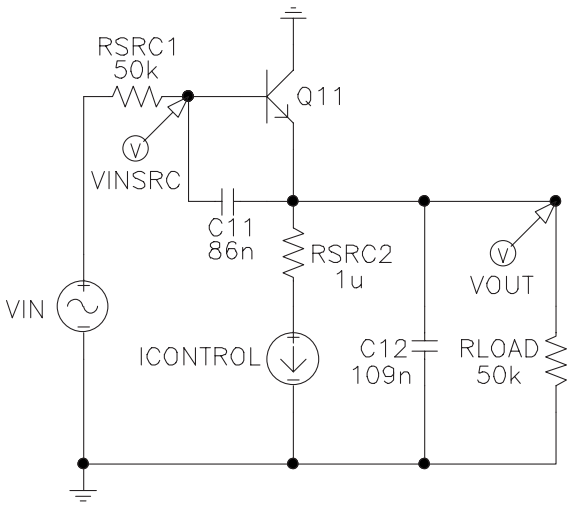

Fig. 11. An evolved circuit for the tunable integrated active filter.

A patent application was filed in 2002 for improved proportional, integrative, and derivative (PID) tuning rules and non-PID controllers that were automatically created by means of genetic programming (Keane et al., 2002; Koza, Keane, Streeter, Mydlowec, Yu, & Lanza, 2003). The genetically evolved tuning rules and controllers outperform controllers tuned using the widely used Ziegler–Nichols tuning rules (Ziegler & Nichols, 1942) and the recently developed Åström–Hägglund tuning rules (Åström & Hägglund, 1995). We believe that the new tuning rules and controllers satisfy the statutory requirement of being "improved" and "useful." They are certainly "new." Because they contain features that would never occur to an experienced control engineer, they are certainly "unobvious" to someone "having ordinary skill in the art." If a patent is granted (and, just prior to final editing of this paper, we have been informed that it will be), it will (we believe) be the first patent granted for an invention that was automatically created by means of genetic programming. For further discussion of the potential of genetic programming as an invention machine, see Koza, Keane, and Streeter (2003).

Table 4 shows the two inventions generated by genetic programming for which a patent application has been filed.

The second invention in Table 4 is what we call a *parameterized topology*. A parameterized topology is a general (parameterized) solution to a problem in the form of a graphical structure whose nodes or edges represent components and where the parameter values of the structure's components are specified by mathematical expressions containing free variables.

In a parameterized topology, the genetically evolved graphical structure represents a complex structure (e.g., electrical circuit, controller, network of chemical reactions,

https://doi.org/10.1017/S089006040404017X Published online by Cambridge University Press

Table 4. *Two patentable inventions created by genetic programming*

Claimed Invention	Patent Applic. Date	Inventors	Reference
Improved general-purpose tuning rules for a PID controller	July 12, 2002	Martin A. Keane, John R. Koza, & Matthew J. Streeter	Keane et al. (2002), Section 12.3 of Koza et al. (2003*b*)
Improved general-purpose non-PID controllers	July 12, 2002	Martin A. Keane, John R. Koza, & Matthew J. Streeter	Keane et al. (2002), Section 13.2 of Koza et al. (2003*b*)

PID, proportional, integrative, and derivative.

antenna, genetic network). In the automated process, genetic programming determines the graph's size (its number of nodes) as well as the graph's connectivity (specifying which nodes are connected). Genetic programming also assigns, in the automated process, component types to the graph's nodes or edges. In the automated process, genetic programming also creates mathematical expressions that establish the parameter values of the components (e.g., the capacitance of a capacitor in a circuit, the amplification factor of a gain block in a controller). Some of these genetically created mathematical expressions contain free variables. The free variables confer generality on the genetically evolved solution by enabling a single genetically evolved graphical structure to represent a general (parameterized) solution to an entire category of problems. Genetic programming can do all the above in an automated way in a single run.

The capability of genetic programming to create parameterized topologies for design problems is illustrated by the automatic creation of a general-purpose non-PID controller (Fig. 12) whose blocks are parameterized by mathematical expressions containing the problem's four free variables: the plant's time constant (T_r), ultimate period (T_u), ultimate gain (K_u), and dead time (L). This genetically evolved controller (Fig. 12) outperforms PID controllers tuned using the widely used Ziegler–Nichols tuning rules (1942) and the recently developed Åström and Hägglund tuning rules (1995) on an industrially representative set of plants.

This controller's overall topology consists of three adders, three subtractors, four gain blocks parameterized by a constant, two gain blocks parameterized by nonconstant mathematical expressions containing free variables, and two lead

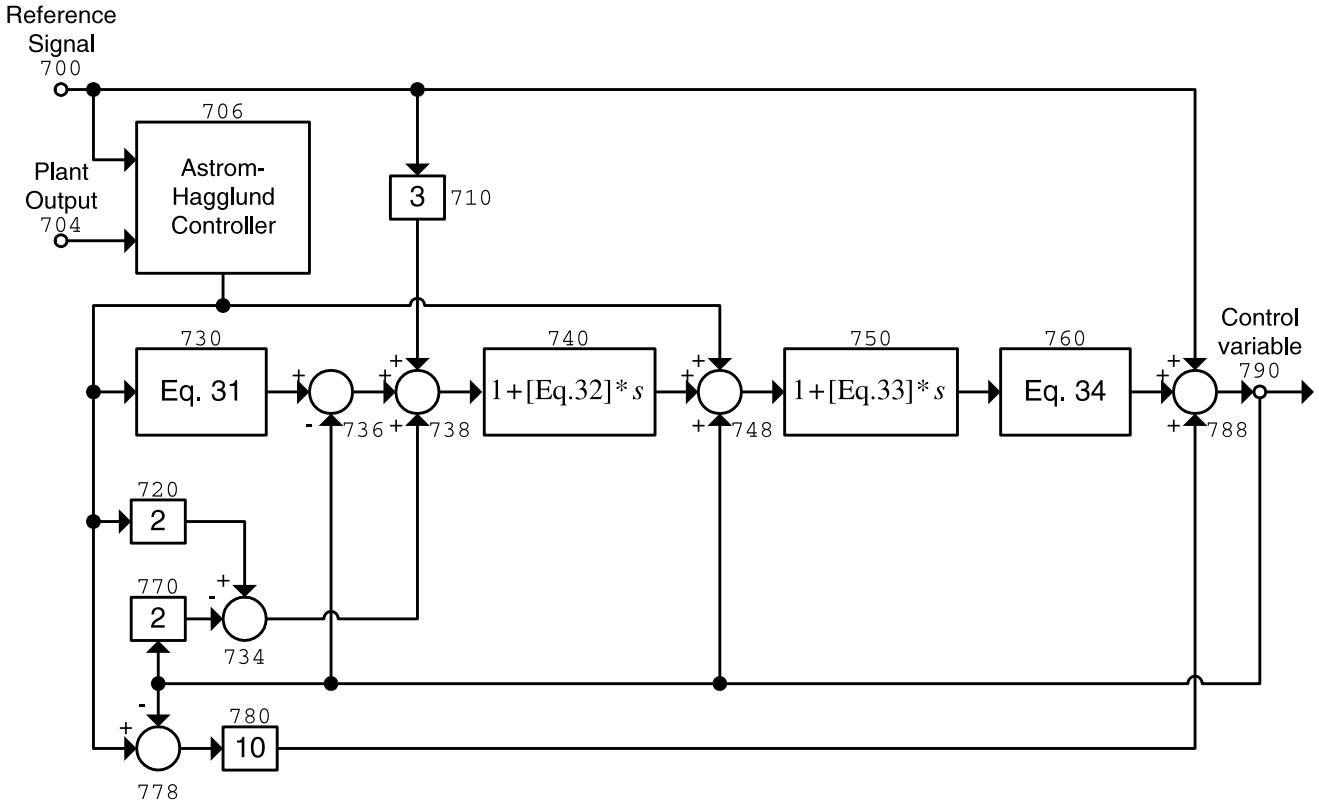

Fig. 12. The parameterized topology for a general-purpose controller.

https://doi.org/10.1017/S089006040404017X Published online by Cambridge University Press

blocks parameterized by nonconstant mathematical expressions containing free variables.

For example, gain block 730 of Figure 12 has a gain that is parameterized by the following nonconstant mathematical expression [Eq. (31) in Fig. 12]:

$$\left|\log\left|T_r - T_{\mathrm{u}} + \log\left|\frac{\log(|L|^L)}{T_{\mathrm{u}}+1}\right|\right|\right|$$

and gain block 760 of Figure 12 has a gain that is parameterized by [Eq. (34) in Fig. 12]

$$|\log|T_r + 1||.$$

Lead block 740 of Figure 12 is parameterized by the following nonconstant mathematical expression [Eq. (32) in Fig. 12]:

$$\mathrm{NLM}(\log|L| - (\mathrm{abs}(L)^L)^2 T_{\mathrm{u}}^3 (T_{\mathrm{u}}+1) T_r e^L - 2T_{\mathrm{u}} e^L),$$

where NLM is the nonlinear mapping described in Koza, Keane, Streeter, Mydlowec, Yu, and Lanza (2003).

Lead block 750 in Figure 12 is parameterized by [Eq. (33) in Fig. 12]

$$\mathrm{NLM}(\log|L| - T_{\mathrm{u}} e^L (2K_{\mathrm{u}}(\log|K_{\mathrm{u}} e^L| - \log|L|)T_{\mathrm{u}} + K_{\mathrm{u}} e^L)).$$

7. OBTAINING PROGRESSIVELY MORE SUBSTANTIAL RESULTS IN SYNCHRONY WITH INCREASING COMPUTER POWER

Table 5 lists the five computer systems used to produce our group's reported work on genetic programming in the 15-year period between 1987 and 2002. Column 7 shows the number of human-competitive results (out of 36) generated by each computer system. These 36 human-competitive results include 21 of the 22 results in Table 1 along with the two patentable new inventions in Table 4 and 13 additional human-competitive results produced by our group that are not patent related (itemized in Koza, Keane, Streeter, Mydlowec, Yu, & Lanza, 2003).

The first entry in Table 5 is a serial computer. The four subsequent entries are parallel computer systems. The presence of four increasingly powerful parallel computer systems in Table 5 reflects the fact that genetic programming has successfully taken advantage of the increased computational power available by means of parallel processing (thereby avoiding a pitfall that often constrains other proposed approaches to machine intelligence).

Table 5 shows the following:

- There is an order of magnitude speedup (column 4) between each successive computer system in the table. Note that, according to Moore's law, exponential increases in computer power correspond approximately to constant periods of time.
- There is a 13,900:1 speedup (column 5) between the fastest and most recent machine (the 1000-node parallel computer system) and the slowest and earliest computer system in Table 5 (the serial LISP machine).
- The slower early machines generated few or no human-competitive results, whereas the faster more recent machines have generated numerous human-competitive results.

Four successive order of magnitude increases in computer power are explicitly shown in Table 5. An additional order of magnitude increase was achieved by making extraordinarily long runs on the largest machine in Table 5 (the 1000-node Pentium II parallel machine). The length of the run that produced the genetically evolved controller (Fig. 12) was 28.8 days, almost an order of magnitude increase (9.3 times) over the overall 3.4-day average for runs of genetic programming that our group has made in recent years. If this final 9.3:1 increase is counted as an additional order of magnitude increases in computer power, the overall increase in computer power shown in Table 5 is 130,660:1.

Table 6 is organized around the five just-explained order of magnitude increases in the expenditure of computing power. Column 4 characterizes the qualitative nature of the

Table 5. *Human-competitive results produced by genetic programming with five computer systems*

System	Period	Petacycles (10^{15} cycles)/ Day for System	Speed-Up	Speed-Up Over First System	Used for Work in Book	Human-Competitive Results
Serial Texas Instruments LISP	1987–1994	0.00216	1 (base)	1 (base)	Koza (1992) and Koza (1994*a*)	0
64-node Transtech transputer parallel	1994–1997	0.02	9	9	A few problems in Koza et al. (1999*a*)	2
64-node Parsytec parallel	1995–2000	0.44	22	204	Most problems in Koza et al. (1999*a*)	12
70-node Alpha parallel	1999–2001	3.2	7.3	1,481	A minority (8) of problems in Koza et al. (2003*b*)	2
1000-node Pentium II parallel	2000–2002	30.0	9.4	13,900	A majority (28) of the problems in Koza et al. (2003*b*)	12

https://doi.org/10.1017/S089006040404017X Published online by Cambridge University Press

Table 6. *Progression of qualitatively more substantial results produced by genetic programming in relation to five order of magnitude increases in computational power*

System	Period	Speed-Up	Qualitative Nature of Results Produced by Genetic Programming
Serial Texas Instruments LISP	1987–1994	1 (base)	• Toy problems of the 1980s and early 1990s from the fields of artificial intelligence and machine learning
64-node Transtech transputer parallel	1994–1997	9	• Two human-competitive results involving 1-dimensional discrete data (not patent-related)
64-node Parsytec parallel	1995–2000	22	• One human-competitive result involving 2-dimensional discrete data • Numerous human-competitive results involving continuous signals analyzed in the frequency domain • Numerous human-competitive results involving 20th-century patented inventions
70-node Alpha parallel	1999–2001	7.3	• One human-competitive result involving continuous signals analyzed in the time domain • Circuit synthesis extended from topology and sizing to include routing and placement (layout)
1000-node Pentium II parallel	2000–2002	9.4	• Numerous human-competitive results involving continuous signals analyzed in the time domain • Numerous general solutions to problems in the form of parameterized topologies • Six human-competitive results duplicating the functionality of 21st-century patented inventions
Long (4-week) runs of 1000-node Pentium II parallel	2002	9.3	• Generation of two patentable new inventions

results produced by genetic programming. The table shows the progression of qualitatively more substantial results produced by genetic programming in terms of five order of magnitude increases in the expenditure of computational resources.

The order of magnitude increases in computer power shown in Table 6 correspond closely (albeit not perfectly) with the following progression of qualitatively more substantial results produced by genetic programming:

- toy problems,
- human-competitive results not related to patented inventions,
- 20th-century patented inventions,
- 21st-century patented inventions, and
- patentable new inventions.

This progression demonstrates that genetic programming is able to take advantage of the exponentially increasing computational power made available by iterations of Moore's law.

For over 200 years, the US Patent Office has been in the business of receiving written descriptions of human-designed inventions and judging whether the purported inventions are

- "new,"
- "improved,"
- "useful," and
- "[un]obvious . . . to a person having ordinary skill in the art to which said subject matter pertains." (35 *United States Code* 103a)

When the Patent Office passes judgment on a patent application, it generally works from written documents and operates at arms length from the inventor. When an automated method duplicates the detailed structure of a previously patented human-created invention, the fact that the human-designed version originally satisfied the Patent Office's criteria for patent worthiness means that an automatically created duplicate would also have satisfied the Patent Office's criteria for patent worthiness had it arrived at the Patent Office prior to the human inventor's submission.

When genetic programming is applied to a problem whose solution is a previously patented invention, there are three possible outcomes:

- failure of the run to solve the problem,
- creation of a solution that infringes a previously issued patent, or
- creation of a noninfringing solution that duplicates the functionality of a previously patented invention.

There are two subcases associated with the third case. First, a noninfringing solution may be a previously known solution (i.e., prior art). The previously known solution may or may not have been patented in the past. Second, a noninfringing solution may be a new solution to the problem. In this second subcase, a new, genetically evolved, noninfringing solution may be patentable if it satisfies the additional requirements of being useful, improved, and unobvious. A genetically evolved solution would generally be deemed to be useful for the same reasons that the orig-

https://doi.org/10.1017/S089006040404017X Published online by Cambridge University Press

inally patented invention was deemed to be useful. Almost every alternative solution to a particular problem usually has some attribute that can be reasonably viewed (from some standpoint) as being improved in some respect or to some degree. Because genetically evolved solutions often contain features that would never occur to human scientists or engineers, a genetically evolved alternative solution will often be unobvious to someone "having ordinary skill in the art."

US law suggests that inventions created by automated means are patentable by saying, "Patentability shall not be negatived by the manner in which the invention was made" (35 *United States Code* 103a).

8. METHODS FOR EXTENDING GENETIC PROGRAMMING TO MORE COMPLEX CIRCUITS

Sections 3 and 4 demonstrated that genetic programming can automatically synthesize analog circuits that duplicate the functionality of two particular 20th-century inventions. Section 5 demonstrated that genetic programming can automatically synthesize analog circuits that duplicate the functionality of six circuits that were patented after January 1, 2000. Section 6 presented examples of patentable new inventions that have already been produced by genetic programming. Section 7 discussed the progression of qualitatively more substantial results produced by genetic programming in relation to five order of magnitude increases in computational power. The question therefore arises of whether and how the techniques described can be extended to automatically synthesize the topology and sizing of more complex circuits.

Table 7 tallies the computer time consumed by the 11 runs of the six post-2000 patented circuits described in Section 5 of this paper. Column 2 of this table shows the product of the total population size (M) and the number of generations ($i + 1$) run before the best-of-run individual was encountered. Column 3 shows the length of the run in hours.

Table 7. *Computer time consumed by 11 runs of the six problems involving post-2000 patented inventions*

Run	$M \times (i + 1)$	Hours
Low-voltage balun circuit	490,000,000	25
Mixed analog–digital variable capacitor	198,000,000	88
High-current load circuit		
First run	230,000,000	134
Second run	432,000,000	67
Voltage–current conversion circuit	550,000,000	83
Cubic function generator		
First run	915,000,000	206
Second run	654,000,000	135
Tunable integrated active filter		
First run	142,000,000	23
Second run	102,000,000	14
Third run	78,000,000	12
Fourth run	56,000,000	6

As can be seen from Table 7, the average number of hours for runs involving each of the six post-2000 patented circuits is 25, 88, 99, 83, 170, and 14, respectively. The average of these averages is 80 h (3.3 days). (We use the average of the averages here because we happened to make four runs of the problem that took the least computer time.)

All six problems were run on a home-built parallel computer system consisting of 1000 350-MHz Pentium II processors (last row, Table 5). This system operates at an overall rate of 3.5×10^{11} Hz. A 3.3-day (80-h) run represents about 10^{17} cycles (i.e., 100 petacycles).

The relentless iteration of Moore's law promises increased availability of computational resources in future years. If available computer capacity continues to double approximately every 18 months over the next decade, a computation requiring 80 h will require only about 1% as much computer time (i.e., about 48 min) a decade from now.

Aside from the promise of increased availability of computational resources in the future, there are two reasons why we are currently not near the boundary of the current capability of genetic programming to automatically synthesize analog circuits.

One reason is that multiple runs of a probabilistic algorithm are often necessary to solve a problem. However, all 11 runs involving the post-2000 patented circuits (ignoring partial runs used for debugging purposes) produced a satisfactory solution. A success rate of 100% is unusual with a probabilistic algorithm. This high rate, over six problems involving 21st-century patented inventions, suggests that we are currently not near the boundary of the current capability of techniques used to produce those results.

A second reason is the intentional uniformity (and, hence, inefficiency) of our runs of the six problems involving 21st-century patented circuits. In approaching the six problems, our goal was to use as little problem-specific human-supplied knowledge, information, analysis, and intelligence as possible. This "clean hands" orientation is, of course, entirely irrelevant to a practicing engineer interested in extending the technique to produce more complex and more useful results.

Runs of circuit synthesis problems employing genetic programming can be accelerated in various ways.

First, many pieces of elementary knowledge helpful to the construction of useful electrical circuits were not made available to the runs of the six problems involving 21st-century patented circuits. For example, the initial population of individuals in a run of genetic programming is typically created at random. As the run proceeds, new individuals are created by probabilistic problem-independent operations (e.g., crossover, mutation). Consequently, many individuals in the population (although syntactically correct) represent unrealistic or impractical electrical circuits.

https://doi.org/10.1017/S089006040404017X Published online by Cambridge University Press

One particularly egregious characteristic of the circuits that appear in unrestricted runs of genetic programming is that the circuit draws preposterously large amounts of current. To cull circuits of this type from the population, each circuit in the population can be examined for the current drawn by the circuit's positive power supply and negative power supply. If the current exceeds a certain generous maximum (e.g., an absolute value of 250 mA), the circuit is penalized (or perhaps eliminated).

Second, a threshold requirement for a functioning circuit is that the circuit connect to all input signals, all output signals, and all necessary sources of power (e.g., the positive power supply and the negative power supply). Many randomly created circuits do not satisfy this threshold requirement. The process of generating random circuits in generation 0 (and the process of applying the genetic operations to create the new individuals for later generations) can be continued until all circuits satisfy this threshold requirement.

Third, the components that are inserted into a developing circuit need not be as primitive as a single transistor, a single resistor, or a single capacitor. Instead, the set of component-creating functions can be expanded to include numerous frequently occurring, known useful combinations of components. Examples include current mirrors, voltage gain stages, Darlington emitter–follower sections, and cascodes. Graeb et al. (2001) have identified (for a purpose entirely unrelated to evolutionary computation) a promising set of frequently occurring, combinations of transistors that are known to be useful in analog circuitry. For certain problems, the set of primitives can also be expanded (from mere one-, two-, or four-transistor combinations) to include higher level entities, such as filters, amplifiers, oscillators, voltage-controller current sources, multipliers, and phase-locked loops.

Fourth, although the runs of the six post-2000 patented circuits were intentionally done in a uniform way, a practicing engineer has no reason to enforce such uniformity. For example, we did not use automatically defined functions (subroutines) for any of the six problems involving post-2000 patented circuits. However, most practical electrical circuits are replete with reuse. A practicing engineer would recognize that reuse is highly relevant in at least two of the six problems (namely, the mixed analog–digital integrated circuit for variable capacitance and the low-voltage high-current transistor circuit for testing a voltage source). The practicing engineer would have no reason to forego the manifest benefits of reuse and automatically defined functions. The benefits of using automatically defined functions in problems having parallelism, regularity, symmetry, and modularity are considerable (Koza, 1990, 1992, 1994*a*; Koza & Rice, 1991).

Fifth, there are numerous opportunities to incorporate problem-specific knowledge into a run of genetic programming. For example, the developmental process need not start merely with a single modifiable wire. A substructure of known utility for a particular problem can be hard wired into every individual in the initial generation of the population, thereby relieving genetic programming of the need to reinvent it. For example, the search for a high-performance amplifier might begin with a balanced voltage gain stage or a differential pair as the starting point.

Sixth, it is possible to integrate general knowledge of electrical engineering into a run of genetic programming. For example, Sripramong (2001) and Sripramong and Toumazou (2002) combine current-flow analysis into runs of genetic programming for the purpose of automatically synthesizing CMOS amplifiers.

Seventh, assuming that candidate circuits are simulated by means of software (as is the case for the work described in this paper), the evolutionary process consumes a considerable amount of computer time to evaluate the a circuit's fitness. As evolutionary computation is applied to larger and more complex circuits, computer time can become a limitation on the ability to produce usable results. Fortunately, considerable work has been done in recent years to accelerate the convergence characteristics and general efficiency of circuit simulators. Although we used a version of the SPICE3 simulator (Quarles et al., 1994) that we modified in various ways (as described in Koza, Bennett, Andre, & Keane, 1999), numerous commercial simulators are now available that are considerably more efficient. Speed-ups of up to 10:1 are reportedly possible today. Moreover, ongoing progress in the field of evolvable hardware (such as field-programmable transistor arrays, as described by Stoica et al., 2001) suggest that it may be possible to replace software simulators by hardware simulations in future years.

Eighth, as pointed out by Sripramong (2001) and Sripramong and Toumazou (2002), the set of functions and the set of terminals described in Section 2.3 tend to create circuits whose connectivity does not resemble the connectivity of commonly encountered practical circuits. Sripramong (2001) and Sripramong and Toumazou (2002) have demonstrated that it is more efficient, in the context of automatically synthesizing CMOS amplifiers by means of genetic programming, to insert transistors in a way that heavily biases the insertion process so that one lead of each new transistor is connected to a power source, the circuit's input, the circuit's output, or to a point in the circuit that is not too close to the just-inserted transistor.

Looking forward, we believe that genetic programming will be increasingly used to automatically generate ever more complex human-competitive results.

9. CONCLUSIONS

This paper has argued that creative inventions are characterized by a logical discontinuity and singular moment when prevailing thinking is, in a "flash of genius," overthrown and replaced by a new approach. The paper illustrated this point by discussing the history of the invention of negative feedback by Harold S. Black. The paper showed that Black's

https://doi.org/10.1017/S089006040404017X Published online by Cambridge University Press

1927 invention can be automatically created by means of an automated design and invention process (genetic programming) and that this automated invention process is accomplished by a probabilistic search without resort to logic. The paper also showed that a 19th-century patented invention, numerous 20th-century patented inventions, other 20th-century inventions (such as the Sallen–Key filter), six 21st-century inventions, and two patentable new inventions have been discovered in a similar automated way by means of genetic programming. The paper argued that evolutionary search is a promising approach for achieving automated design because illogic, and hence creativity, is inherent in the evolutionary process.

REFERENCES

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

Andre, D., & Teller, A. (1999). Evolving team Darwin united. In *RoboCup-98: Robot Soccer World Cup II, Lecture Notes in Computer Science 1604* (Asada, M., & Kitano, H., Eds.), pp. 346–352. Berlin: Springer–Verlag.

Andre, D., Bennett III, F.H., & Koza, J.R. (1996). Discovery by genetic programming of a cellular automata rule that is better than any known rule for the majority classification problem. In *Genetic Programming 1996: Proc. First Annual Conf.* (Koza, J.R., Goldberg, D.E., Fogel, D.B., & Riolo, R.L., Eds.), pp. 3–11. Cambridge, MA: MIT Press.

Armstrong, E.H. (1914). *Wireless Receiving System*. US patent 1,113,149. Filed October 29, 1913, issued October 6, 1914.

Åström, K.J., & Hägglund, T. (1995). *PID Controllers: Theory, Design, and Tuning*, 2nd ed. Research Triangle Park, NC: Instrument Society of America.

Aytur, T.S. (2000). *Integrated Circuit with Variable Capacitor*. US patent 6,013,958. Filed July 23, 1998, issued January 11, 2000.

Balkir, S., Dundar, G., & Ogrenci, A.S. (2003). *Analog VLSI Design Automation*. Boca Raton, FL: CRC Press.

Banzhaf, W., Nordin, P., Keller, R.E., & Francone, F.D. (1998). *Genetic Programming—An Introduction*. San Francisco, CA: Morgan Kaufmann and Heidelberg.

Black, H.S. (1928). *Translating System*. US patent 1,686,792. Filed February 3, 1925, issued October 9, 1928.

Black, H.S. (1935). *Wave Translation System*. US patent 2,003,282. Filed August 8, 1928, issued June 4, 1935.

Black, H.S. (1937*a*). *Wave Translation System*. US patent 2,102,670. Filed August 8, 1928, issued December 21, 1937.

Black, H.S. (1937*b*). *Wave Translation System*. US patent 2,102,671. Filed April 22, 1932, issued December 21, 1937.

Black, H.S. (1977). Inventing the negative feedback amplifier. *IEEE Spectrum December*, 55–60.

Cipriani, S., & Takeshian, A.A. (2000). *Compact Cubic Function Generator*. US patent 6,160,427. Filed September 4, 1998, issued December 12, 2000.

Daun–Lindberg, T.C., & Miller, M.L. (2000). *Low Voltage High-Current Electronic Load*. US patent 6,211,726. Filed June 28, 1999, issued April 3, 2001.

Goddard, R. (1915). *Method of and Apparatus for Producing Electrical Impulses or Oscillations*. US patent 1,159,209. Filed August 1, 1912, issued November 2, 1915.

Graeb, H.E., Zizala, S., Eckmueller, J., & Antreich, K. (2001). The sizing rules method for analog circuit design. *Proc. Second IEEE/ACM Int. Conf. Computer Aided Design*, pp. 343–349. Piscataway, NJ: IEEE Press.

Gruau, F. (1992*a*). *Cellular Encoding of Genetic Neural Networks*. Technical Report 92-21. Lyon: Ecole Normale Supérieure de Lyon, Laboratoire de l'Informatique du Parallélisme.

Gruau, F. (1992*b*). Genetic synthesis of Boolean neural networks with a cell rewriting developmental process. *Proc. Workshop on Combinations of Genetic Algorithms and Neural Networks 1992* (Schaffer, J.D., & Whitley, D., Eds.). Los Alamitos, CA: IEEE Press.

Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. Ann Arbor, MI: University of Michigan Press.

Ikeuchi, A., & Tokuda, N. (2000). *Voltage–Current Conversion Circuit*. US patent 6,166,529. Filed February 24, 2000, issued December 26, 2000 in US; filed March 10, 1999 in Japan.

Irvine, R., & Kolb, B. (2001). *Integrated Low-Pass Filter*. US patent 6,225,859. Filed September 14, 1998, issued May 1, 2001.

Kardontchik, J.E. (1992). *Introduction to the Design of Transconductor–Capacitor Filters*. Boston: Kluwer Academic.

Karki, J. (1999). *Analysis of the Sallen–Key Architecture*. Texas Instruments Application Report SLOA024A.

Keane, M.A., Koza, J.R., & Streeter, M.J. (2002). *Improved General-Purpose Controllers*. US patent 6,847,851. Filed July 12, 2002, issued January 25, 2005.

Kitano, H. (1990). Designing neural networks using genetic algorithms with graph generation system. *Complex Systems 4*, 461–476.

Koza, J.R. (1990). *Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems*. Technical Report STAN-CS-90-1314. Stanford University, Computer Science Department.

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

Koza, J.R. (1993). Discovery of rewrite rules in Lindenmayer systems and state transition rules in cellular automata via genetic programming. *Symp. Pattern Formation (SPF-93)*, Claremont, CA, February 13.

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

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

Koza, J.R., & Rice, J.P. (1991). Genetic generation of both the weights and architecture for a neural network. *Proc. Int. Joint Conf. Neural Networks*, pp. 397–404. Los Alamitos, CA: IEEE Press.

Koza, J.R., & Rice, J.P. (1992). *Genetic Programming: The Movie*. Cambridge, MA: MIT Press.

Koza, J.R., Bennett III, F.H., Andre, D., & Keane, M.A. (1999). *Genetic Programming III: Darwinian Invention and Problem Solving*. San Francisco, CA: Morgan Kaufmann.

Koza, J.R., Bennett III, F.H., Andre, D., Keane, M.A., & Brave, S. (1999). *Genetic Programming III Videotape: Human-Competitive Machine Intelligence*. San Francisco, CA: Morgan Kaufmann.

Koza, J.R., Keane, M.A., & Streeter, M.J. (2003). Evolving inventions. *Scientific American 288(2)*, 52–59.

Koza, J.R., Keane, M.A., Streeter, M.J., Mydlowec, W., Yu, J., & Lanza, G. (2003). *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*. Boston: Kluwer Academic.

Koza, J.R., Keane, M.A., Streeter, M.J., Mydlowec, W., Yu, J., Lanza, G., & Fletcher, D. (2003). *Genetic Programming IV Video: Routine Human-Competitive Machine Intelligence*. Boston: Kluwer Academic.

Lancaster, D. (1995). *Active Filter Cookbook*. Thatcher, AZ: Synergetics Press.

Lee, S.G. (2001). *Low Voltage Balun Circuit*. US patent 6,265,908. Filed December 15, 1999, issued July 24, 2001.

Lee, T.H. (1998). *The Design of CMOS Radio-Frequency Integrated Circuits*. Cambridge, MA: Cambridge University Press.

Lipson, H. (2004). How to draw a straight line using a GP: Benchmarking evolutionary design against 19th century kinematic synthesis. In *Genetic and Evolutionary Conf. 2005 Late-Breaking Papers* (Keijzer, M., Ed.). Seattle, WA: International Society for Genetic and Evolutionary Computation. [CD]

Lohn, J.D., Hornby, G.S., & Linden, D.S. (2004). An evolved antenna for deployment on NASA's Space Technology 5 Mission. In *Genetic Programming Theory and Practice II* (O'Reilly, U.-M., Riolo, R.L., Yu, G., & Worzel, W., Eds.), Chap. 18. Boston: Kluwer Academic.

Luke, S. (1998). Genetic programming produced competitive soccer softbot teams for RoboCup97. In *Genetic Programming 1998: Proc. Third Annual Conf.* (Koza, J.R., Banzhaf, W., Chellapilla, K., Deb, K., Dorigo, M., Fogel, D.B., Garzon, M.H., Goldberg, D.E., Iba, H., & Riolo, R., Eds.), pp. 214–222, University of Wisconsin, Madison, WI, July 22–25. San Francisco, CA: Morgan Kaufmann.

Quarles, T., Newton, A.R., Pederson, D.O., & Sangiovanni–Vincentelli, A. (1994). *SPICE 3 Version 3F5 User's Manual*. Berkeley, CA: University of California, Department of Electrical Engineering and Computer Science.

Spector, L. (2004). *Automatic Quantum Computer Programming: A Genetic Programming Approach*. Boston: Kluwer Academic.
Spector, L., Barnum, H., & Bernstein, H.J. (1998). Genetic programming for quantum computers. In *Genetic Programming 1998: Proc. Third Annual Conf.* (Koza, J.R., Banzhaf, W., Chellapilla, K., Deb, K., Dorigo, M., Fogel, D.B., Garzon, M.H., Goldberg, D.E., Iba, H., & Riolo, R., Eds.), pp. 365–373. San Francisco, CA: Morgan Kaufmann.
Spector, L., Barnum, H., & Bernstein, H.J. (1999). Quantum computing applications of genetic programming. In *Advances in Genetic Programming 3* (Spector, L., Langdon, W.B., O'Reilly, U.-M., & Angeline, P., Eds.), pp. 135–160. Cambridge, MA: MIT Press.
Spector, L., Barnum, H., Bernstein, H.J., & Swamy, N. (1999). Finding a better-than-classical quantum AND/OR algorithm using genetic programming. In *Proc. 1999 Congr. Evolutionary Computation*, pp. 2239–2246. Piscataway, NJ: IEEE Press.
Sripramong, T. (2001). *The evolution of analogue CMOS circuits using genetic programming*. PhD Thesis. London: University of London and Imperial College.
Sripramong, T., & Toumazou, C. (2002). The invention of CMOS amplifiers using genetic programming and current-flow analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 21(11)*, 1237–1252.
Stoica, A., Zebulum, R., & Keymeulen, D. (2001). Polymorphic electronics. In *Evolvable Systems: From Biology to Hardware, 4th Int. Conf., Proc. ICES 2001. Lecture Notes in Computer Science 2210* (Liu, Y., Tanaka, K., Iwata, M., Higuchi, T., & Yasunaga, M., Eds.), pp. 291–302, Tokyo, October. Berlin: Springer–Verlag.
Vladimirescu, A.. (1994). *The SPICE Book*. New York: Wiley.
Wilson, S.W. (1987). The genetic algorithm and biological development. In *Genetic Algorithms and Their Applications: Proc. Second Int. Conf. Genetic Algorithms* (Grefenstette, J.J., Ed.), pp. 247—251. Hillsdale, NJ: Erlbaum.
Ziegler, J.G., & Nichols, N.B. (1942). Optimum settings for automatic controllers. *Transactions of ASME 64*, 759–768.

John R. Koza received his PhD in computer science from the University of Michigan in 1972 under the supervision of John Holland. He is currently a Consulting Professor in the Biomedical Informatics Program in the Department of Medicine and a Consulting Professor in the Department of Electrical Engineering, both at Stanford University. Dr. Koza has taught a course on genetic algorithms and genetic programming at Stanford University since 1988.

Martin A. Keane received a PhD in mathematics from Northwestern University in 1969. He worked for Applied Devices Corporation until 1972, was employed in the Mathematics Department at General Motors Laboratory until 1976, and was Vice-President for Engineering of Bally Manufacturing Corporation until 1986. Dr. Keane is currently Chief Scientist of Econometrics Inc. of Chicago and a Consultant to various computer-related and gaming-related companies.

Matthew J. Streeter received an MS in computer science from Worcester Polytechnic Institute in 2001 and worked at Genetic Programming as a Systems Programmer and Researcher from 2001 to 2003. He is currently a PhD candidate at Carnegie Mellon University.

Thomas P. Adams is a second-year MS student at Stanford University pursuing an artificial intelligence specialization. He was a Programmer and Consultant to the financial and manufacturing industry and Programmer and Researcher at Genetic Programming Inc.

Lee W. Jones received a BS degree in computational biology from Carnegie Mellon University in 1997. He has previously directed groups and consulted in the field of bioinformatics. His primary research interests lie where the fields of biology and computer science intersect. He is currently working at Genetic Programming Inc. as a Systems Programmer and Researcher.