Human-competitive Applications of Genetic Programming

John R. Koza

Stanford Medical Informatics, Department of Medicine, School of Medicine, Department of Electrical Engineering, School of Engineering, Stanford University, Stanford, California 94305 E-mail: koza@stanford.edu

Summary: Genetic programming is an automatic technique for producing a computer program that solves, or approximately solves, a problem. This chapter reviews several recent examples of human-competitive results produced by genetic programming. The examples all involve the automatic synthesis of a complex structure from a high-level statement of the requirements for the structure. The illustrative results include examples of automatic synthesis of both the topology and sizing (component values) for analog electrical circuits, automatic synthesis of placement and routing (as well as topology and sizing) for circuits, and automatic synthesis of both the topology and tuning (parameter values) of controllers.

1 Introduction

Genetic programming is an automatic technique for producing a computer program that solves, or approximately solves, a problem. Genetic programming addresses the challenge of getting a computer to solve a problem without explicitly programming it. This challenge calls for an automatic system whose input is a high-level statement of a problem's requirements and whose output is a working program that solves the problem. Paraphrasing Arthur Samuel (1959), this challenge concerns, "How can computers be made to do what needs to be done, without being told exactly how to do it?"

Since many problems can be easily recast as a search for a computer program, genetic programming can potentially solve a wide range of types of problems, including problems of control, classification, system identification, and design.

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

Section 2 describes genetic programming. Section 3 states what we mean when we say that an automatically created solution to a problem is competitive with the product of human creativity. Section 4 describes how genetic programming has been applied to problems of synthesis of both the topology and sizing (component values) for analog electrical circuits. Section 5 extends this process to include the automatic creation of the placement and routing of circuits (as well as the automatic creation of the topology and sizing). Section 6 describes the application of genetic programming to the problem of automatically synthesizing the design of both the topology and tuning (parameter values) for controllers.

2 Genetic Programming

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

On each generation of a run of genetic programming, each individual in the population of programs is evaluated as to its fitness in solving the problem at hand. The programs in generation 0 of a run almost always have exceedingly poor fitness for non-trivial problems of interest. Nonetheless, some individuals in a population will turn out to be somewhat more fit than others. These differences in performance are then exploited so as to direct the search into promising areas of the search space. The Darwinian principle of reproduction and survival of the fittest is used to probabilistically select, on the basis of fitness, individuals from the population to participate in various operations. A small percentage (e.g., 9%) of the selected individuals are reproduced (copied) from one generation to the next. A very small percentage (e.g., 1%) of the selected individuals are mutated in a random way. Mutation can be viewed as an undirected local search mechanism. The vast majority of the selected individuals participate in the genetic operation of crossover (sexual recombination) in which two offspring programs are created by recombining genetic material from two parents.

The creation of the initial random population and the creation of offspring by the genetic operations are all performed so as to create syntactically valid, executable programs. After the genetic operations are performed on the current generation of the population, the population of offspring (i.e., the new generation) replaces the old generation. The tasks of measuring fitness, Darwinian selection, and genetic operations are then iteratively repeated over many generations.

Genetic programming is an extension of the genetic algorithm (Holland 1975). Genetic programming is described in books such as Koza 1992; Koza 1994a; Koza, Bennett, Andre, and Keane 1999; Banzhaf, Nordin, Keller, and Francone 1998; Langdon 1998; Ryan 1999, Wong and Leung 2000; Langdon and Poli 2002; in edited collections of papers such as Kinnear 1994; Angeline and Kinnear 1996; and Spector, Langdon, O'Reilly, and Angeline 1999; in conference proceedings such as Koza, Goldberg, Fogel, and Riolo 1996; Koza, Deb, Dorigo, Fogel, Garzon, Iba, and Riolo 1997; Koza, Banzhaf, Chellapilla, Deb, Dorigo, Fogel, Garzon, Goldberg, Iba, and Riolo 1998; Banzhaf, Daida, Eiben, Garzon, Honavar, Jakiela, and Smith 1999; Whitley, Goldberg, Cantu-Paz, Spector, Parmee, and Beyer 2000; Spector, Goodman, Wu, Langdon, Voigt, Gen, Sen, Dorigo, Pezeshk, Garzon, and Burke 2001; Banzhaf, Poli, Schoenauer, and Fogarty 1998; Poli, Nordin, Langdon, and Fogarty 1999; Poli, Banzhaf, Langdon, Miller, Nordin, and Fogarty 2000; and Miller, Tomassini, Lanzi, Ryan, Tettamanzi, and Langdon 2001; in videotapes such as Koza and Rice 1992; Koza 1994b; and Koza, Bennett, Andre, Keane, and Brave 1999; in the new *Genetic Programming and Evolvable Machines* journal; and at web sites such as www.genetic-programming.org.

3 Human-competitive Machine Intelligence

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

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

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

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

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

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

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

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

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

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

Note that each of the above criteria are couched in terms of *producing results* and that the results are measured in terms of standards that are external to the fields of artificial intelligence and machine learning.

Using the above criteria, there are now at least 25 instances where genetic programming has produced a result that is competitive with human performance. These examples come from fields such as quantum computing, the annual Robo Cup competition, cellular automata, computational molecular biology, sorting networks, the automatic synthesis of the design of analog electrical circuits, and the automatic synthesis of the design of controllers.

Table 1 shows 25 instances of results where genetic programming has produced results that are competitive with the products of human creativity and inventiveness. Each claim is accompanied by the particular criterion (from the list above) that establishes the basis for the claim. As can be seen in the table, seven of these automatically created results infringe on previously issued patents. In addition, one of the genetically evolved results improves on a previously issued patent. Also, nine of the other genetically evolved results duplicate the functionality of previously patented inventions in a novel way. Since nature routinely uses evolution and natural selection to create designs for complex structures that are well adapted to their environments, it is not surprising that many of these examples involve the design of complex structures.

Table 1. Twenty-five instances where genetic programming has produced human-competitive results.

	Claimed instance	Basis for claim	Reference	Infringed patent
1	Creation, using genetic programming, of a better-than-classical quantum algorithm for the Deutsch-Jozsa "early promise" problem	B, F	(Spector, Barnum, and Bernstein 1998)	
2	Creation, using genetic programming, of a better-than-classical quantum algorithm for Grover's database search problem	B, F	(Spector, Barnum, and Bernstein 1999)	

3	Creation, using genetic programming, of a quantum algorithm for the depth-2 AND/OR query problem that is better than any previously published result	B, D	(Spector, Barnum, Bernstein, and Swamy 1999)	
4	Creation of soccer-playing program that ranked in the middle of the field of 34 human-written programs in the Robo Cup 1998 competition	H	(Andre and Teller 1999)	
5	Creation of four different algorithms for the transmembrane segment identification problem for proteins	B, E	(Koza, Bennett, Andre, and Keane 1999)	
6	Creation of a sorting network (O'Connor, and Nelson 1962) for seven items using only 16 steps	A, D	(Koza, Bennett, Andre, and Keane 1999)	
7	Rediscovery of the ladder topology for lowpass and highpass filters	A, F	(Koza, Bennett, Andre, and Keane 1999)	(Campbell 1917)
8	Rediscovery of "*M*-derived half section" and "constant *K*" filter sections	A, F	(Koza, Bennett, Andre, and Keane 1999)	(Zobel 1925)
9	Rediscovery of the Cauer (elliptic) topology for filters	A, F	(Koza, Bennett, Andre, and Keane 1999)	(Cauer 1934, 1935, 1936)
10	Automatic decomposition of the problem of synthesizing a crossover filter	A, F	(Koza, Bennett, Andre, and Keane 1999)	(Zobel 1925)
11	Rediscovery of a recognizable voltage gain stage and a Darlington emitter-follower section of an amplifier and other circuits	A, F	(Koza, Bennett, Andre, and Keane 1999)	(Darlington 1953)
12	Synthesis of 60 and 96 decibel amplifiers	A, F	(Koza, Bennett, Andre, and Keane 1999)	
13	Synthesis of analog computational circuits for squaring, cubing, square root, cube root, logarithm, and Gaussian functions	A, D, G	(Koza, Bennett, Andre, and Keane 1999)	

14	Synthesis of a real-time analog circuit for time-optimal control of a robot	G	(Koza, Bennett, Andre, and Keane 1999)	
15	Synthesis of an electronic thermometer	A, G	(Koza, Bennett, Andre, and Keane 1999)	
16	Synthesis of a voltage reference circuit	A, G	(Koza, Bennett, Andre, and Keane 1999)	
17	Creation of a cellular automata rule for the majority classification problem that is better than the Gacs-Kurdyumov-Levin (GKL) rule and all other known rules written by humans	D, E	(Andre, Bennett, and Koza 1996)	
18	Creation of motifs that detect the D-E-A-D box family of proteins and the manganese superoxide dismutase family	C	(Koza, Bennett, Andre, and Keane 1999)	
19	Synthesis of analog circuit equivalent to Philbrick circuit (Philbrick 1956)	A	(Koza, Bennett, Keane, Yu, Mydlowec, and Stiffelman 1999)	
20	Synthesis of NAND circuit	A	(Bennett, Koza, Keane, Yu, Mydlowec, and Stiffelman 1999)	
21	Synthesis of digital-to-analog converter (DAC) circuit	A	(Bennett, Koza, Keane, Yu, Mydlowec, and Stiffelman 1999)	
22	Synthesis of analog-to-digital (ADC) circuit	A	(Bennett, Koza, Keane, Yu, Mydlowec, and Stiffelman 1999)	

23	Synthesis of topology, sizing, placement, and routing of analog electrical circuits	G	(Koza and Bennett 1999)	
24	Synthesis of topology for a PID type of controller	A, F	(Koza, Keane, Yu, Bennett, Mydlowec 2000)	(Callender and Stevenson 1939)
25	Synthesis of topology for a controller with a second derivative	A, F	(Koza, Keane, Yu, Bennett, Mydlowec 2000)	(Jones 1942)

The fact that genetic programming can evolve entities that infringe on previously patented inventions, improve on previously patented inventions, or duplicate the functionality of previously patented inventions suggests that genetic programming can potentially be used as an "invention machine" to create new and useful patentable inventions.

4 Automatic Circuit Synthesis

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

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

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

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

"Analog designers are few and far between. In contrast to digital design, most of the analog circuits are still handcrafted by the experts or so-called 'zahs' of analog design. The design process is characterized by a combination of experience and intuition and requires a thorough knowledge of the process characteristics and the detailed specifications of the actual product."

"Analog circuit design is known to be a knowledge-intensive, multiphase, iterative task, which usually stretches over a significant period of time and is performed by designers with a large portfolio of skills. It is therefore considered by many to be a form of art rather than a science."

We use a simple filter circuit to demonstrate the automatic synthesis of analog electrical circuits using genetic programming. A *filter* is a one-input, one-output circuit that receives a signal as its input and passes the frequency components of the incoming signal that lie in a specified range (called the *passband*) while suppressing the frequency components that lie in all other frequency ranges (the *stopband*). Specifically, the goal is to design a lowpass filter composed of capacitors and inductors that passes all frequencies below 1,000 Hertz (Hz) and suppresses all frequencies above 2,000 Hz.

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

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

(1) topology-modifying functions that alter the topology of a developing circuit,

(2) component-creating functions that insert components into a developing circuit,

(3) development-controlling functions that control the development process by which the embryo and its successors become a fully developed circuit,

(4) arithmetic-performing functions that appear in subtrees as argument(s) to the component-creating functions and specify the numerical value of the component, and

(5) automatically defined functions that appear in the automatically defined functions and potentially enable certain substructures of the circuit to be reused (with parameterization).

Before applying genetic programming to a problem of circuit design, seven major preparatory steps are required: (1) identify the embryonic circuit, (2) determine the architecture of the circuit-constructing program trees, (3) identify the primitive functions of the program trees, (4) identify the terminals of the program trees, (5) create the fitness measure, (6) choose control parameters for the run, and (7) determine the termination criterion and method of result designation. A detailed discussion concerning how to apply these seven preparatory steps to a particular problem of circuit synthesis (such as a lowpass filter) is found in Koza, Bennett, Andre, and Keane 1999 (chapter 25).

4.1 Campbell 1917 Ladder Filter Patent

The best circuit (Fig. 1) of generation 49 of one run of genetic programming (Koza, Bennett, Andre, and Keane 1996) on the problem of synthesizing a lowpass filter is a 100% compliant circuit (i.e., it complies with all requirements for attenuation, passband ripple, and stopband ripple).

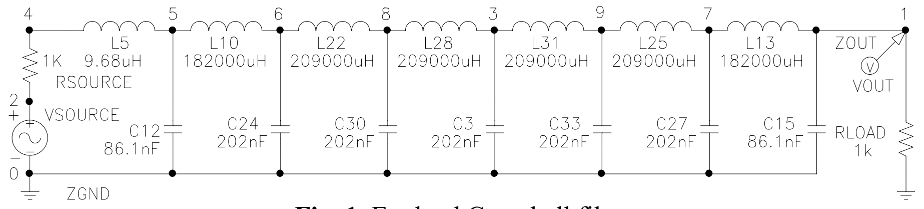

Fig. 1. Evolved Campbell filter

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

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

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

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

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

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

The fact that genetic programming rediscovered both the topology and sizing of an electrical circuit that was unobvious "to a person having ordinary skill in the art" establishes that this evolved result satisfies Arthur Samuel's criterion (1983) for artificial intelligence and machine learning, namely

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

4.2 Zobel 1925 "*M*-Derived Half Section" Patent

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

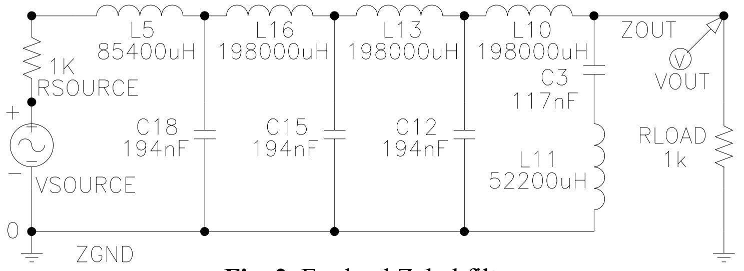

Fig. 2. Evolved Zobel filter

This evolved circuit (presented in Koza, Bennett, Andre, and Keane 1999, chapter 25) is equivalent to a cascade of three symmetric T-sections and an *M*-derived half section. Otto Zobel of American Telephone and Telegraph Company invented and received a patent for an "*M*-derived half section" used in conjunction with one or more "constant K" sections. Again, the numerical values of all the components in the evolved circuit closely approximate the numerical values specified in Zobel's 1925 patent.

4.3 Cauer 1934–1936 Elliptic Filter Patents

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

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

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

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

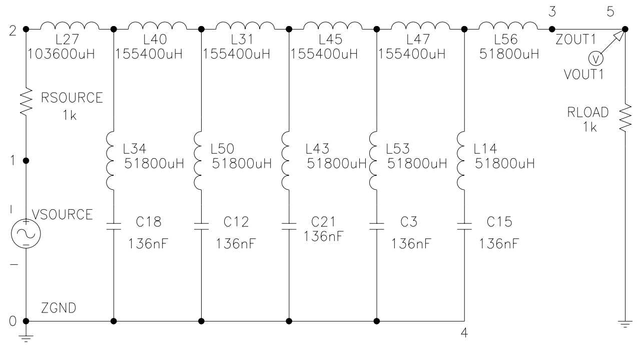

Fig. 3. Evolved Cauer (elliptic) filter topology

4.4 Other Circuits

In addition, genetic programming has also been successfully used to synthesize the design for many other types of filters, including highpass, bandpass, bandstop, crossover, comb, and asymmetric filters (Koza, Bennett, Andre, and Keane 1999; Koza, Bennett, Andre, Keane, and Brave 1999). Also, genetic programming has been applied to the problem of automatic synthesis of both the topology and sizing of many analog electrical circuits composed of transistors. These include amplifiers (evolved using multiobjective fitness measures that consider gain, distortion, bandwidth, parts count, power consumption, and power supply rejection ratio), computational circuits (square root, squaring, cube root, cubing, logarithmic, and Gaussian), time-optimal controller circuits, source identification circuits, temperature-sensing circuits, and voltage reference circuits (Koza, Bennett, Andre, and Keane 1999; Koza, Bennett, Andre, Keane, and Brave 1999).

The amplifiers, computational circuits, electronic thermometers, and voltage reference circuits were all covered by one or more patents when they were first invented. Many of these circuit include previously patented subcircuits, such as Darlington emitter-follower sections (Darlington 1953).

5 Topology, Sizing, Placement, and Routing of Circuits

Circuit *placement* involves the assignment of each of the circuit's components to a particular physical location on a printed circuit board or silicon wafer. *Routing* involves the assignment of a particular physical location to the wires between the leads of the circuit's components.

Genetic programming can simultaneously create a circuit's topology and sizing along with the placement and routing of all components as part of an integrated overall design process (Koza and Bennett 1999). It can do this while also optimizing additional other considerations (such as minimizing the circuit's area).

This is accomplished by using an initial circuit that contains information about the geographic (physical) location of components and wires and using component-inserting and topology-modifying operations that appropriately adjust the geographic (physical) location of components and associated wires. For example, the initial circuit in the developmental process complies with the requirements that wires must not cross on a particular layer of a silicon chip or on a particular side of a printed circuit board, that there must be a wire connecting 100% of the leads of all the circuit's components, and that minimum clearance distances between wires, between components, and between wires and components must be maintained. Similarly, each of the circuit-constructing functions used in preserves compliance with these requirements. Thus, every fully laid-out circuit complies with these requirements.

For example, in one run, a lowpass filter circuit was first evolved in generation 25 for a discrete-component printed circuit board. The topology and component sizing for this circuit complied with all requirements (for passband ripple, stopband ripple, and attenuation); however, this circuit contained five capacitors and 11 inductors and occupied an area of 1775.2. Later, a 100%-compliant lowpass filter was created in generation 30 containing 10 inductors and five capacitors occupying an area of 950.3. Then, in generation 138, a physically compact lowpass filter circuit (Fig. 4) containing four inductors and four capacitors and occupying an area of only 359.4 was created. As can be seen, this circuit has the Campbell topology (Campbell 1917).

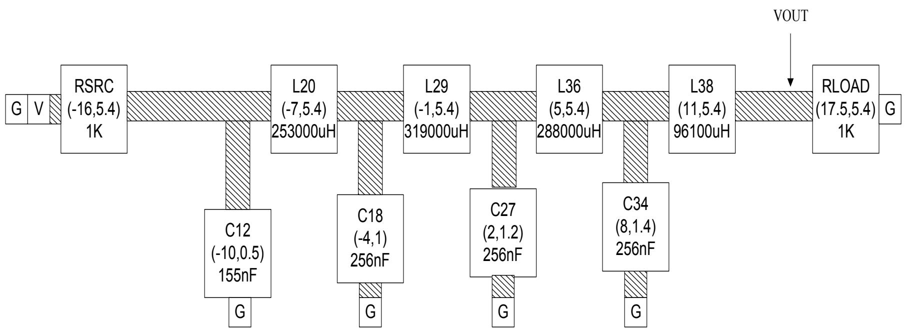

Fig. 4. Topology, sizing, placement, and routing of a lowpass filter for a printed circuit board

6 Automatic Synthesis of Controllers

The design of controllers is another area where there has been (prior to genetic programming) no previously known general technique for automatically creating the topology and tuning for a controller from a high-level statement of the design goals for the controller.

The purpose of a controller is to force, in a meritorious way, the actual response of a system (conventionally called the *plant*) to match a desired response (called the *reference signal*) (Astrom and Hagglund 1995; Boyd and Barratt 1991; Dorf and Bishop 1998).

In the PID type of controller, the controller's output is the sum of proportional (P), integrative (I), and derivative (D) terms based on the difference between the plant's output and the reference signal. The PID controller was patented in 1939 by Albert Callender and Allan Stevenson of Imperial Chemical Limited of Northwich, England.

Claim 1 of Callender and Stevenson (1939) covers what is now called the PI controller,

"A system for the automatic control of a variable characteristic comprising means proportionally responsive to deviations of the characteristic from a desired value, compensating means for adjusting the value of the characteristic, and electrical means associated with and actuated by responsive variations in said responsive means, for operating the compensating means to correct such deviations in conformity with the sum of the extent of the deviation and the summation of the deviation."

Claim 3 of Callender and Stevenson (1939) covers what is now called the PID controller,

"A system as set forth in claim 1 in which said operation is additionally controlled in conformity with the rate of such deviation."

The vast majority of automatic controllers used by industry are of the PID type. As Astrom and Hagglund (1995) observe, "Several studies … indicate the state of the art of industrial practice of control. The Japan Electric Measuring Instrument Manufacturing Association conducted a survey of the state of process control systems in 1989 … According to the survey, more than 90% of the control loops were of the PID type."

However, it is generally recognized by leading practitioners in the field of control that PID controllers are not ideal and that there are significant limitations on analytical techniques in designing controllers. As Boyd and Barratt stated in *Linear Controller Design: Limits of Performance* (Boyd and Barratt 1991),

"The challenge for controller design is to productively use the enormous computing power available. Many current methods of computer-aided controller design simply automate procedures developed in the 1930's through the 1950's …"

There is no preexisting general-purpose analytic method for automatically creating a controller for arbitrary linear and non-linear plants that can simultaneously optimize prespecified performance metrics (such as minimizing the time required to bring the plant output to the desired value as measured by, say, the integral of the time-weighted absolute error), satisfy time-domain constraints (involving, say, overshoot and disturbance rejection), satisfy frequency domain constraints (e.g., bandwidth), and satisfy additional constraints, such as constraints on the magnitude of the control variable and the plant's internal state variables.

6.1 Robust Controller for a Two-lag Plant

We employ a problem involving control of a two-lag plant to illustrate the automatic synthesis of controllers by means of genetic programming. The problem here (described by Dorf and Bishop 1998, page 707) is to create both the topology and parameter values for a controller for a two-lag plant such that plant output reaches the level of the reference signal so as to minimize the integral of the time-weighted absolute error (ITAE), such that the overshoot in response to a step input is less than 2%, and such that the controller is robust in the face of significant variation in the plant's internal gain, K, and the plant's time constant, τ.

Genetic programming routinely creates PI and PID controllers infringing on the 1942 Callender and Stevenson patent during intermediate generations of runs of genetic programming on controller problems. However, the PID controller is not the best controller for this (and many) problems.

Fig. 5 shows the block diagram for the best-of-run controller evolved on generation 32 of one run of the two-lag plant problem. In this figure, $R(s)$ is the reference signal; $Y(s)$ is the plant output; and $U(s)$ is the controller's output (control variable).

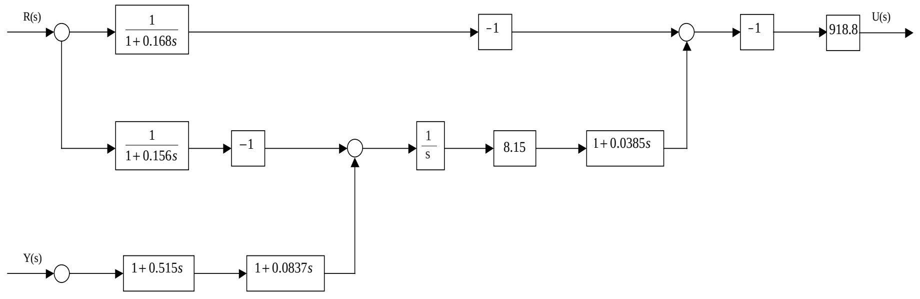

Fig. 5. Best-of-run genetically evolved controller

The controller evolved by genetic programming differs from a conventional PID controller in that the genetically evolved controller employs a second-derivative processing block. As will be seen, this evolved controller is 2.42 times better than the Dorf and Bishop (1998) controller as measured by the criterion used by Dorf and Bishop (namely, the integral of the time-weighted absolute error). In addition, this evolved controller has only 56% of the rise time in response to the reference input, has only 32% of the settling time, and is 8.97 times better in terms of suppressing the effects of disturbance at the plant input.

After applying standard manipulations to the block diagram of this evolved controller, the transfer function for the best-of-run controller from generation 32 for the two-lag plant can be expressed as a transfer function for a pre-filter and a transfer function for a compensator. The transfer function for the pre-filter, $G_{p32}(s)$, for the best-of-run individual from generation 32 is

$$G_{p32}(s)=\frac{1(1+.1262s)(1+.2029s)}{(1+.03851s)(1+.05146)(1+.08375)(1+.1561s)(1+.1680s)}$$

The transfer function for the compensator, $G_{c32}(s)$, is

$$G_{c32}(s)=\frac{7487(1+.03851s)(1+.05146s)(1+.08375s)}{s}$$
$$=\frac{7487.05+1300.63s+71.2511s^2+1.2426s^3}{s}$$

The s^3 term (in conjunction with the s in the denominator) indicates a second derivative. Thus, the compensator consists of a second derivative in addition to proportional, integrative, and derivative functions. Harry Jones of The Brown Instrument Company of Philadelphia patented this same kind of controller topology in 1942.

Claim 38 of the Jones 1942 patent (Jones 1942) states,

"In a control system, an electrical network, means to adjust said network in response to changes in a variable condition to be controlled, control means responsive to network adjustments to control said condition, reset means including a reactance in said network adapted following an adjustment of said network by said first means to initiate an additional network adjustment in the same sense, and rate control means included in said network adapted to control the effect of the first mentioned adjustment in accordance with the second or higher derivative of the magnitude of the condition with respect to time."

Note that the user of genetic programming did not preordain, prior to the run (as part of the preparatory steps for genetic programming), that a second derivative should be used in the controller (or, for that matter, that a P, I, or D block should be used). The evolutionary process discovered that these elements were helpful in producing a good controller for this problem. That is, necessity was the mother of invention. Similarly, the user did not preordain any particular topological arrangement of proportional, integrative, derivative, second derivative, or other functions within the automatically created controller. Instead, genetic programming automatically created a robust controller for the given plant without the benefit of user-supplied information concerning the total number of processing blocks to be employed in the controller, the type of each processing block, the topological interconnections between the blocks, the values of parameters for the blocks, or the existence of internal feedback (none in this instance) within the controller.

7 Conclusion

This chapter has demonstrated that genetic programming can produce human-competitive designs from complex structures. The results in this chapter (and the other recently produced human-competitive results in Table 1) suggest that genetic programming is on the threshold of routinely producing such human-competitive results. We expect that the rapidly decreasing cost of computing power will enable genetic programming to deliver additional human-competitive results on increasingly difficult problems and, in particular, that genetic programming will be routinely used as an "invention machine" for producing patentable new inventions.

References

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

Andre, David, Bennett III, Forrest H, and Koza, John R. 1996. Discovery by genetic programming of a cellular automata rule that is better than any known rule for the majority classification problem. In Koza, John R., Goldberg, David E., Fogel, David B., and Riolo, Rick L. (editors). *Genetic Programming 1996: Proceedings of the First Annual Conference, July 28-31, 1996, Stanford University*. Cambridge, MA: MIT Press. Pages 3–11.

Andre, David and Teller, Astro. 1999. Evolving team Darwin United. In Asada, Minoru and Kitano, Hiroaki (editors). *RoboCup-98: Robot Soccer World Cup II*. Lecture Notes in Computer Science. Volume 1604. Berlin: Springer-Verlag. Pages 346-352.

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

Astrom, Karl J. and Hagglund, Tore. 1995. *PID Controllers: Theory, Design, and Tuning*. Second Edition. Research Triangle Park, NC: Instrument Society of America.

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

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

Bennett III, Forrest H, Koza, John R., Keane, Martin A., Yu, Jessen, Mydlowec, William, and Stiffelman, Oscar. 1999. Evolution by means of genetic programming of analog circuits that perform digital functions. In Banzhaf, Wolfgang, Daida, Jason, Eiben, A. E., Garzon, Max H., Honavar, Vasant, Jakiela, Mark, and Smith, Robert E. (editors). 1999. *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference, July 13-17, 1999, Orlando, Florida, USA*. San Francisco, CA: Morgan Kaufmann. Pages 1477 - 1483.

Boyd, S. P. and Barratt, C. H. 1991. *Linear Controller Design: Limits of Performance*. Englewood Cliffs, NJ: Prentice Hall.

Callender, Albert and Stevenson, Allan Brown. 1939. *Automatic Control of Variable Physical Characteristics*. U.S. Patent 2,175,985. Filed February 17, 1936 in United States. Filed February 13, 1935 in Great Britain. Issued October 10, 1939 in United States.

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

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

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

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

Darlington, Sidney. 1953. *Semiconductor Signal Translating Device*. U.S. Patent 2,663,806. Filed May 9, 1952. Issued December 22, 1953.

Dorf, Richard C. and Bishop, Robert H. 1998. *Modern Control Systems*. Eighth edition. Menlo Park, CA: Addison-Wesley.

Holland, John 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. Second edition. Cambridge, MA: The MIT Press 1992.

Jones, Harry S. 1942. *Control Apparatus*. U.S. Patent 2,282,726. Filed October 25, 1939. Issued May 12, 1942.

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

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

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

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

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

Koza, John R., and Bennett III, Forrest H. 1999. Automatic Synthesis, Placement, and Routing of Electrical Circuits by Means of Genetic Programming. In Spector, Lee, Langdon, William B., O'Reilly, Una-May, and Angeline, Peter (editors). 1999. *Advances in Genetic Programming 3*. Cambridge, MA: The MIT Press. Chapter 6. Pages 105 - 134.

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

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

Koza, John R., Bennett III, Forrest H, Andre, David, and Keane, Martin A. 1996. Automated design of both the topology and sizing of analog electrical circuits using genetic programming. In Gero, John S. and Sudweeks, Fay (editors). *Artificial Intelligence in Design '96*. Dordrecht: Kluwer Academic. Pages 151-170.

Koza, John R., Bennett III, Forrest H, Keane, Martin A., Yu, Jessen, Mydlowec, William, and Stiffelman, Oscar. 1999. Searching for the impossible using genetic programming. In Banzhaf, Wolfgang, Daida, Jason, Eiben, A. E., Garzon, Max H., Honavar, Vasant, Jakiela, Mark, and Smith, Robert E. (editors). 1999. *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference, July 13-17, 1999, Orlando, Florida, USA*. San Francisco, CA: Morgan Kaufmann. Pages 1083 - 1091.

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

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

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

Koza, John R., Keane, Martin A., Yu, Jessen, Bennett III, Forrest H, and Mydlowec, William. 2000. Automatic creation of human-competitive programs and controllers by means of genetic programming. *Genetic Programming and Evolvable Machines*. 1(1-2) 121 – 164.

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

Langdon, William B. and Poli, Riccardo. 2002. *Foundations of Genetic Programming*. Springer-Verlag.

Miller, Julian, Tomassini, Marco, Lanzi, Pier Luca, Ryan, Conor, Tettamanzi, Andrea G. B., and Langdon, William B. (editors). 2001. *Genetic Programming: 4th European Conference, EuroGP 2001, Lake Como, Italy, April 2001 Proceedings*. Berlin: Springer.

Wong, Man Leung and Leung, Kwong Sak. 2000. *Data Mining Using Grammar Based Genetic Programming and Applications*. Amsterdam: Kluwer Academic Publisher. O'Connor, Daniel G. and Nelson, Raymond J. 1962. *Sorting System with N-Line Sorting Switch*. U.S. Patent 3,029,413. Issued April 10, 1962.

Philbrick, George A. 1956. *Delayed Recovery Electric Filter Network*. Filed May 18, 1951. U.S. Patent 2,730,679. Issued January 10, 1956.

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

Poli, Riccardo, Banzhaf, Wolfgang, Langdon, William B., Miller, Julian, Nordin, Peter, and Fogarty, Terence C. 2000. *Genetic Programming: European Conference, EuroGP 2000, Edinburgh, Scotland, UK, April 2000, Proceedings*. Lecture Notes in Computer Science. Volume 1802. Berlin, Germany: Springer-Verlag.

Ryan, Conor. 1999. *Automatic Re-engineering of Software Using Genetic Programming*. Amsterdam: Kluwer Academic Publisher.

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

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

Spector, Lee, Barnum, Howard, and Bernstein, Herbert J. 1998. Genetic programming for quantum computers. In Koza, John R., Banzhaf, Wolfgang, Chellapilla, Kumar, Deb, Kalyanmoy, Dorigo, Marco, Fogel, David B., Garzon, Max H., Goldberg, David E., Iba, Hitoshi, and Riolo, Rick. (editors). 1998. *Genetic Programming 1998: Proceedings of the Third Annual Conference*. San Francisco, CA: Morgan Kaufmann. Pages 365 - 373.

Spector, Lee, Barnum, Howard, and Bernstein, Herbert J. 1999. Quantum computing applications of genetic programming. In Spector, Lee, Langdon, William B., O'Reilly, Una-May, and Angeline, Peter (editors). 1999. *Advances in Genetic Programming 3*. Cambridge, MA: The MIT Press. Pages 135-160.

Spector, Lee, Barnum, Howard, Bernstein, Herbert J., and Swamy, N. 1999. Finding a better-than-classical quantum AND/OR algorithm using genetic programming. In *IEEE Proceedings of 1999 Congress on Evolutionary Computation*. Piscataway, NJ: IEEE Press. Pages 2239-2246.

Spector, Lee, Goodman, E., Wu, A., Langdon, William B., Voigt, H.-M., Gen, M., Sen, S., Dorigo, Marco, Pezeshk, S., Garzon, Max, and Burke, E. (editors). 2001. *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001*. San Francisco, CA: Morgan Kaufmann Publisher.

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

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

Whitley, Darrell, Goldberg, David, Cantu-Paz, Erick, Spector, Lee, Parmee, Ian, and Beyer, Hans-Georg (editors). 2000. *GECCO-2000: Proceedings of the Genetic and Evolutionary Computation Conference, July 10 - 12, 2000, Las Vegas, Nevada*. San Francisco: Morgan Kaufmann Publishers.

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