Routine Duplication of Post-2000 Patented Inventions by Means of Genetic Programming

Matthew J. Streeter
Genetic Programming Inc., Mountain View, California
mjs@tmolp.com

Martin A. Keane
Econometrics Inc., Chicago, Illinois
makeane@ix.netcom.com

John R. Koza
Stanford University, Stanford, California
koza@stanford.edu

ABSTRACT

Previous work has demonstrated that genetic programming can automatically create analog electrical circuits, controllers, and other devices that duplicate the functionality and, in some cases, partially or completely duplicate the exact structure of inventions that were patented between 1917 and 1962. This paper reports on a project in which we browsed patents of analog circuits issued after January 1, 2000 on the premise that recently issued patents represent current research that is considered to be of practical and scientific importance. The paper describes how we used genetic programming to automatically create circuits that duplicate the functionality or structure of five post-2000 patented inventions. This work employed four new techniques (motivated by the theory of genetic algorithms and genetic programming) that we believe increased the efficiency of the runs. When an automated method duplicates a previously patented human-designed invention, it can be argued that the automated method satisfies a Patent-Office-based variation of the Turing test.

1 Introduction

Genetic programming can automatically create both the topology and sizing (numerical component values) for a wide variety of analog electrical circuits, controllers, and other devices such as sorting networks merely by specifying the device's high-level behavior [1-3]. Seven of the analog circuits and two of the controllers that have been automatically created by genetic programming infringe patents on that were issued between 1917 and 1962 (i.e., exactly duplicate the structure of the patented invention). Other genetically evolved results duplicate the functionality of 20th Century patented inventions in a novel way. One genetically evolved device (a sorting network) was an improvement over the invention described in the patent.

This paper reports on a project in which we browsed patents of analog circuits issued after January 1, 2000 on the premise that recently issued patents represent current research that is considered to be of practical and scientific importance. The paper describes how we used genetic programming to automatically create circuits that infringe, partially infringe, or duplicate the functionality of five post-2000 inventions that were patented by major commercial and university research institutions. Table 1 shows 14 patented inventions that have been duplicated by genetic programming, including the five inventions described in this paper.

Table 1. Fourteen patented inventions duplicated by genetic programming

Invention	**Date**	**Inventor**	**Place**
PID (proportional, integrative, and derivative) controller	1939	Albert Callender and Allan Stevenson	Imperial Chemical Limited
Second-derivative controller	1942	Harry Jones	Brown Instrument Company
Darlington emitter-follower section	1953	Sidney Darlington	Bell Telephone Laboratories
Ladder filter	1917	George Campbell	American Telephone and Telegraph
Crossover filter	1925	Otto Julius Zobel	American Telephone and Telegraph
"M-derived half section" filter	1925	Otto Julius Zobel	American Telephone and Telegraph
Elliptic filter	1934 - 36	Wilhelm Cauer	Gottingen, Germany
Philbrick circuit	1956	George Philbrick	George A. Philbrick Researches
Sorting network	1962	Daniel G. O'Connor and Raymond J. Nelson	General Precision, Inc.
Mixed analog-digital integrated circuit for producing variable capacitance	2000	Turgut Sefket Aytur	Lucent Technologies Inc.
Voltage-current converter	2000	Akira Ikeuchi and Naoshi Tokuda	Mitsumi Electric Co., Ltd.
Cubic function generator	2000	Stefano Cipriani and Anthony A. Takeshian	Conexant Systems, Inc.
Low-voltage high-current transistor circuit for testing a voltage source	2001	Timothy Daun-Lindberg and Michael Miller	International Business Machines Corporation
Low-voltage balun circuit	2001	Sang Gug Lee	Information and Communications University

2 Overall Method

We used genetic programming to breed a population of circuit-constructing program trees. Each constructing program tree is converted into a circuit by means of a developmental process that starts with a simple embryo. The functions in the circuit-constructing program trees include (1) component-creating functions that insert components (i.e., resistors, capacitors, and transistors) into a developing circuit, (2) topology-modifying functions (e.g., series division, parallel division, via between nodes, via to ground, via to power) that alter the topology of a developing circuit, (3) development-controlling functions (e.g., end, safe cut) that control development.

With the exceptions described in section 4 herein, we used the methods described in [1].

The embryo used on all five problems herein consisted of a single modifiable wire.

We used the same embryo, program architecture, function set, terminal set, control parameters, termination criteria, and computing machinery for all five problems. All runs were made on a home-built Beowulf-style [4] parallel cluster computer system consisting of 1,000 350 MHz Pentium II processors (each with 64 megabytes of RAM).

The only two differences between the runs of genetic programming for the five problems were that we used

(1) different (appropriate) types (models) of transistors for each problem, and

(2) a different fitness measure (and test fixture) for each problem.

3 Fitness Measures for Five Problems

The fitness measure specifies what time-domain or frequency-domain output values are desired, given various inputs. For each specific problem, a test fixture consisting of certain fixed components (such as a source resistor, a load resistor) is connected to the desired input port(s) and the desired output port(s) to measure the output.

3.1 Voltage-Current Conversion Circuit

The purpose of the voltage-current conversion circuit patented by Ikeuchi and Tokuda (U. S. patent 6,166,529) is to take two voltages as input and to produce as output a stable current whose magnitude is proportional to the difference of the voltages [5]. As a fitness measure for this problem, we employed four time-domain input signals (fitness cases). We included a time-varying voltage source 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. The weight of each fitness case was defined as the reciprocal of the patented circuit's error for that fitness case, so that the patent circuit was defined to have a fitness of 1.0.

3.2 Balun Circuit

The purpose of a "balun" (balance/unbalance) circuit, such as that described in U. S. patent 6,265,908, is to divide an input signal into two half-amplitude signals which are 180 degrees out of phase from each other [6]. Additionally, the circuit described in the patent is noteworthy in that it operates using a power supply of only 1 Volt. Our fitness measure for this problem consisted of a (1) frequency sweep analysis designed to ensure the correct magnitude and phase at the two outputs of the circuit and (2) a Fourier analysis designed to penalize harmonic distortion.

3.3 Cubic Signal Generator

U. S. patent 6,160,427 covers a "Compact cubic function generator" [7]. This is a computational circuit designed to produce as output the cube of an input signal. The patented circuit is "compact" in the sense that it requires a voltage drop across no more than two transistors at any point in the circuit. Our 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 allowing the evolutionary process access to only a 2-Volt power supply.

3.4 Register-Controlled Variable Capacitor

U. S. patent 6,013,958 covers a circuit whose behavior is equivalent to that of a capacitor whose capacitance is controlled by the value stored in a digital register [8]. For this problem, we used 16 time-domain fitness cases. The 16 fitness cases ranged over all eight possible values of a 3-bit digital register for two different input signals.

3.5 High-Current Load Circuit

U. S. patent 6,211,726 covers a circuit designed to sink a time-varying amount of current in response to a control signal. Toward this end, Daun-Lindberg and Miller of IBM employed a number of FET transistors arranged in a parallel structure, each of which sinks a small amount of the desired current [9]. Our fitness measure for this problem consisted of two time-domain simulations, each representing a different control signal. Each fitness case was weighted by the reciprocal of the patented circuit's error on that fitness case, so that the patent circuit was defined to have a fitness of 1.0.

4 Four New Techniques Employed in This Work

Broadly speaking, we used the methods for the automatic synthesis of circuits described in [1]. However, we employed four new techniques in these runs. Two of the new techniques were designed to increase the degree to which local substructures are preserved during the developmental process.

4.1 New Function for Connecting Distant Points

Most electrical circuits cannot be laid out in a plane. Instead, practical circuits require connections between distant points. The connections typically cannot be achieved in a totally planar circuit. Our previous work with the automatic synthesis of electrical circuits employed functions such as `VIA` and `PAIR_CONNECT` (described in [1]) to connect distant points in a developing circuit.

The premise of the crossover operation in genetic algorithms and genetic programming is that individuals that have comparatively high fitness are likely to have useful substructures. The `VIA` and `PAIR_CONNECT` functions have the disadvantage that when a subtree of one circuit-constructing program tree is swapped with a subtree of another circuit-constructing program tree, the connectivity of points within both the crossover fragment and the remainder is, almost always, very dramatically altered in a rather arbitrary and unpredictable way. That is, crossover usually significantly disrupts preexisting connections within a local area of the developing circuit (thereby disrupting the very local structures that probably contributed to the individual's comparatively high fitness and to the individual's selection to participate in the genetic operation in the first place).

To the extent that crossover dramatically alters the characteristics of the swapped genetic material, it acquires the characteristics of the mutation operation and its effectiveness in solving the problem approaches that of blind random search.

We addressed this problem concerning the VIA and PAIR_CONNECT functions in the runs described herein by employing a new two-argument function. The new NODE function replaces one modifiable wire (or component) with a series composition consisting of one modifiable wire, a port that can potentially be connected to other point(s) in the circuit, and a second modifiable wire. Prior to the execution of the developmental process, the circuit-constructing program tree is examined to identify the set of NODE functions that are not ancestors of any NODE function higher in the program tree. The NODE functions in this set are called "top-most NODE functions." For each such top-most NODE function, all the ports (if any) associated with NODE functions that are ancestors of the top-most NODE function are connected together. When the crossover operation moves any subtree within the subtree rooted by a particular top-most NODE function into another circuit-constructing program tree, all the ports of the moved subtree remain connected together. Moreover, all the ports in the unmoved remainder of the original remain connected together. We believe that this new approach based on local information encourages the preservation of building blocks and thereby increases the efficiency of the crossover operation.

4.2 New Symmetry-Breaking Procedure using Geometric Coordinates

Parts in conventional schematic diagrams of electronic circuits carry unique (and consecutive) parts numbers. In earlier work, we used the unique part number (created when a component is first inserted into the developing circuit during the developmental process) to break symmetries and thereby determine the behavior of certain circuit-constructing functions. For example, the way in which the PARALLEL_0 and PARALLEL_1 topology-modifying functions carry out the parallel division was determined by referring to the unique parts numbers of neighboring components. Similarly, when a transistor was inserted into a developing circuit, its base, collector, and emitter were permuted by referring to the parts numbers of neighboring components.

The overall circuit-constructing program tree, of course, changes throughout the run of genetic programming. Thus, when neighboring parts change, the behavior of various circuit-constructing functions is dramatically altered in a rather arbitrary and unpredictable way (thereby disrupting the very local structures that probably contributed to the individual's comparatively high fitness and to the individual's selection to participate in the genetic operation in the first place).

To the extent that the genetic operations do not preserve locality, they acquire the characteristics of the mutation operation and their effectiveness in solving the problem begins to approach that of blind random search.

We addressed this problem in the runs described herein by breaking symmetry using the geometric coordinates of each node in the developing circuit (instead of the unique consecutive parts number). Specifically, the positive end of the single embryonic modifiable wire is defined to be at coordinate location (0,0), and its negative end is defined to be at coordinate location (1,0). The coordinate locations of new nodes that are created by functions that entail symmetry breaking (and those created by all other functions) are defined in terms of the coordinate locations of the

existing nodes by recursively dividing the pre-existing modifiable wires into smaller and smaller new wires. In this way, the behavior of the functions that previously relied on the unique consecutive parts number can be defined in terms of information that is local to the region of the circuit where the symmetry-breaking is performed (instead of in terms of the circuit-wide information represented by component numbers). Again, we believe that this new approach based on local information encourages the preservation of building blocks and thereby increases the efficiency of the crossover operation.

4.3 New Function for Inserting Two-Leaded Components

The three-argument `TWO_LEAD` function replaces one modifiable wire (or component) with a series composition consisting of one modifiable wire, a two-leaded component (capacitor or resistor here), and a second modifiable wire. The third argument of the `TWO_LEAD` function consists of a subtree that contains a one-argument capacitor-inserting function or a one-argument resistor-creating function (both of which possess a single numerical parameter indicating the component value). The possibilities for the third argument are under control of a constrained syntactic structure which limits the choice to a resistor or capacitor. The remaining two arguments are the construction-continuing subtrees for each of the two modifiable wires created by this function.

4.4 New Transistor-Inserting Function

The six-argument `Q` function inserts a transistor into a developing circuit, with both the model and orientation of the transistor specified as parameters to the function. The first argument to the function specifies which transistor model is to be used. The set of available transistor models is specific to the problem at hand. The second argument establishes which end (polarity) of the preexisting modifiable wire will be bifurcated (if necessary) in inserting the transistor. It can take on the values `BIFURCATE_POSITIVE` or `BIFURCATE_NEGATIVE`. The third argument specifies which of six possible permutations of the transistor's three leads (base, collector, and emitter) are to be used. It can take on the values `B_C_E`, `B_E_C`, . . . `E_C_B`. That is, together, there are 12 possible ways of inserting a transistor. The remaining three arguments are the construction-continuing subtrees for each of the three modifiable wires created by this function.

5 Results

5.1 Voltage-Current Conversion Circuit

A circuit (Figure 1) emerged on generation 109 of our run of this problem with a fitness of 0.619. 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 which were not part of the fitness measure, and outperformed the patented circuit on these new fitness cases.

5.2 Balun circuit

The best-of-run evolved circuit (Figure 2) was produced in generation 84 and has a fitness of 0.429. The patent circuit had a total fitness of 1.72. That is, the evolved circuit achieves roughly a fourfold improvement over the patented circuit in terms of our fitness measure. The evolved circuit is superior to the patented circuit both in terms of its frequency response and its harmonic distortion.

5.3 Cubic Signal Generator

The best-of-run evolved circuit (Figure 3) was produced generation 182 and has an average error of 4.02 mV. The patented circuit had an average error of 6.76 mV. That, the evolved circuit has approximately 59% of the error of the patented circuit over our four fitness cases.

5.4 Register-Controlled Variable Capacitor

Over our 16 fitness cases, the patented circuit had 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 (deme) which were compliant with a certain maximum level of error. Examination of these harvested individuals revealed a circuit created in generation 98 which matched the topology of the patented circuit. This circuit is presented in Figure 4.

5.5 High-Current Load Circuit

Our run for this problem eventually reached a plateau and produced a circuit that sunk the desired current into one of the negative power supplies (rather than to ground). This "cheating" circuit was not in the spirit of the patented invention. However, on generation 114 of this run (before the cheating solution appeared), a circuit emerged that duplicated Daun-Lindberg and Miller's parallel FET transistor structure. This circuit had a fitness (weighted error) of 1.82, or 182% of the weighted error for the patented circuit. This circuit is presented in Figure 5.

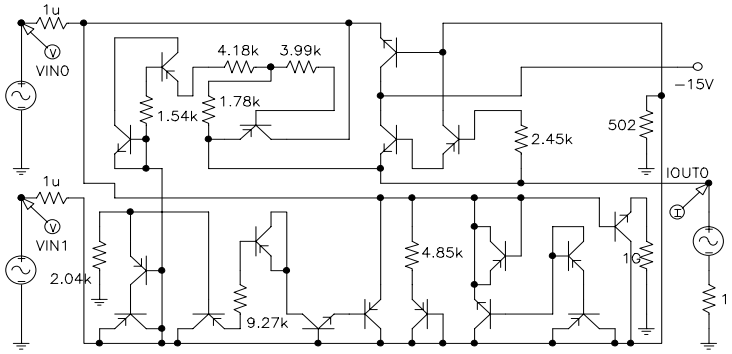

Figure 1. Best-of-run voltage-current-conversion circuit from generation 109

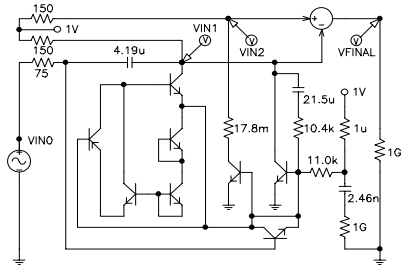

Figure 2. Best evolved balun circuit from generation 84

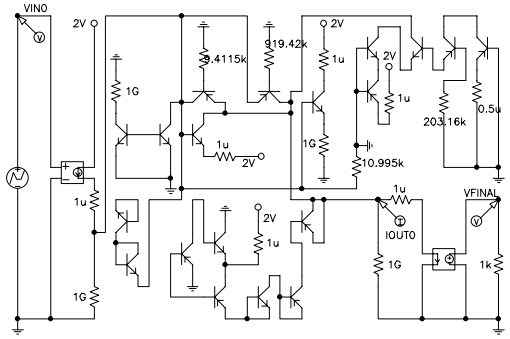

Figure 3. Best-of-run cubic signal generation circuit from generation 182

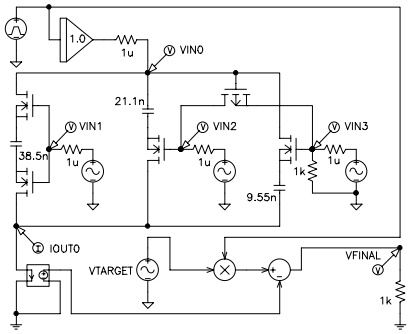

Figure 4. Smallest compliant register-controlled capacitor circuit from generation 98

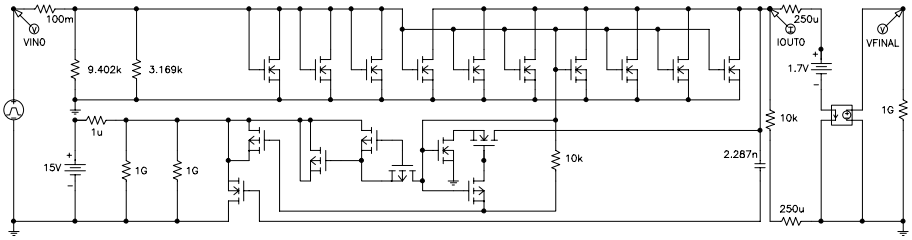

Figure 5. Best-of-run high current load circuit from generation 114

6 Genetic Programming's High-Yield and Routineness

Methods for getting computers to solve problems automatically can be ranked by their "AI ratio." The *AI ratio* is the ratio of that which is delivered by the *artificial* system to the amount of *intelligence* that is supplied by the humans employing the method. The aim, of course, is to get computers to solve problems automatically with a high A-to-I ratio.

The defeat of the then-reigning human world chess champion by the Deep Blue system is an outstanding human-competitive result in the field of machine intelligence. However, there was a enormous of "I" provided by a team that worked on developing Deep Blue's software and hardware for many years in relation to the relatively small amount of "A" delivered by the system. Thus, Deep Blue has a very low A-to-I ratio. Most results produced in the fields of artificial intelligence, machine

learning, and automated reasoning also have low AI ratios (regardless of whether the results are human-competitive).

Similarly, methods for getting computers to solve problems automatically can be ranked by their routineness. The word “routine” includes the connotation of “general purpose.” However, “routine” is more demanding. When we use the term "routine" to describe a method for getting computers to solve problems automatically, we mean that there is a relatively effortless transition required to get the method to successfully handle additional problems within a particular domain and additional problems from a different domain.

Would the transition required to apply Deep Blue’s methods to bridge be routine? Would the transition required to apply Deep Blue’s methods to the problem of getting a robot to mop the floor of an obstacle-laden room be routine? What fraction of Deep Blue’s methods could be brought to bear on programming a cellular automaton to perform a specific calculation? Classifying protein sequences? Designing an amplifier circuit? Devising a algorithm to solve a mathematical problem?

As can be seen in this paper, there was a relatively effortless transition required to get genetic programming to successfully handle additional problems within a particular domain. As we moved from problem to problem, we changed the specification of “what needs to be done” based on the inventor’s statement of desired performance as stated in each patent.

As is apparent from much previous work in the field of genetic programming, a relatively effortless transition is required to get genetic programming to successfully handle additional problems from entirely different domains. Thus, we think it is fair to say that genetic programming is now capable of routinely delivering high-yield human-competitive machine intelligence.

7 Patent-Office-Based Variation of the Turing Test

In 1950, Turing proposed a three-person "imitation game" that might be used to determine whether machine intelligence had been achieved [10]. In the "imitation game," a judge tries to decide whether typewritten replies to questions came from a man or a woman. Turing's original test has been paraphrased in various ways over the years. One popular restatement of Turing's original test for machine intelligence is a two-person game in which a judge receives messages "over a wall" and tries to decide whether the messages came from a human or a machine.

Patent Offices in various countries have been in the business of performing a similar kind of "over the wall" test for over 200 years. For example, the U. S. Patent Office receives written descriptions of human-designed inventions and judges whether they satisfy the statutory requirement of being "[un]obvious … to a person having ordinary skill in the art to which said subject matter pertains."

The Patent Office operates at arms-length and does not know who (or what) actually conceived the proposed invention when it passes judgment on a patent application. The inventor could be an exceptionally creative human or it could be something else (e.g., an automated process). If an automated method were able to duplicate a previously patented human-created invention, the fact that the original human-designed version satisfied the Patent Office's criteria of patent-worthiness means that the automatically created duplicate would also have satisifed the Patent Office's criteria. Thus, whenever an automated method duplicates a previously

patented human-designed invention, the automated method can be viewed as satisfying a Patent-Office-based variation of the Turing test.

The original Turing test (and many of the popular restatements of it) deal with inconsequential chit chat. When an institution or individual allocates time and money to invent something and then also embarks on the time-consuming and expensive process of obtaining a patent, it has made a judgment that the work is of some practical or scientific importance. Moreover, the Patent Office also applies a statutory test of utility as a precondition to issuing a patent. Thus, the above Patent-Office-based variation of the Turing test differs from the original Turing test in that patented inventions represent non-trivial work by exceptionally creative humans.

8 Conclusions

Genetic programming was used to automatically create analog circuits that infringe, partially infringe, or duplicate the functionality of five post-2000 patented inventions. This work employed several new techniques (motivated by the theory of genetic algorithms and genetic programming). The paper also argued that when an automated method duplicates a previously patented human-designed invention, the automated method can be viewed as satisfying a Patent-Office-based variation of the Turing test.

References

1. 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.
2. 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.
3. Koza, John R., Keane, Martin A., Yu, Jessen, Bennett, Forrest H III, and Mydlowec, William. 2000. Automatic creation of human-competitive programs and controllers by means of genetic programming. *Genetic Programming and Evolvable Machines*. (1) 121 - 164.
4. Sterling, Thomas L., Salmon, John, Becker, Donald J., and Savarese, Daniel F. 1999. *How to Build a Beowulf: A Guide to Implementation and Application of PC Clusters*. Cambridge, MA: MIT Press.
5. Ikeuchi, Akira and Tokuda, Naoshi. 2000. *Voltage-Current Conversion Circuit*. U. S. patent 6,166,529. Filed February 24, 2000 in U. S.. Issued December 26, 2000 in U. S.. Filed March 10, 1999 in Japan.
6. Lee, Sang Gug. 2001. *Low Voltage Balun Circuit*. U. S. patent 6,265,908. Filed December 15, 1999. Issued July 24, 2001.
7. Cipriani, Stefano and Takeshian, Anthony A. 2000. *Compact cubic function generator*. U. S. patent 6,160,427. Filed September 4, 1998. Issued December 12, 2000.
8. Aytur; Turgut Sefket. 2000. *Integrated Circuit with Variable Capacitor*. U. S. patent 6,013,958. Filed July 23, 1998. Issued January 11, 2000.
9. Daun-Lindberg, Timothy Charles and Miller; Michael Lee. 2001. *Low Voltage High-Current Electronic Load*. U. S. patent 6,211,726. Filed June 28, 1999. Issued April 3, 2001.
10. Turing, Alan M. 1950. Computing machinery and intelligence. *Mind*. 59(236) 433 – 460. Reprinted in Ince, D. C. (editor). 1992. *Mechanical Intelligence: Collected Works of A. M. Turing*. Amsterdam: North Holland. Pages 133 – 160.