Automatic Synthesis of a Wire Antenna Using Genetic Programming

William Comisky
Genetic Programming Inc.
Los Altos, California
bcomisky@pobox.com

Jessen Yu
Genetic Programming Inc.
Los Altos, California
jyu@cs.stanford.edu

John R. Koza
Stanford University
Los Altos, CA 94023
koza@stanford.edu

Abstract

This paper demonstrates the use of genetic programming to automatically synthesize the design of a wire antenna for an illustrative problem that has been previously solved by both conventional antenna design techniques and the genetic algorithm operating on fixed-length character strings. When the genetic algorithm was used, the human user prespecified many characteristics of the size and shape of the solution. The run of genetic programming also produced a satisfactory result for the illustrative problem. However, it did not require the human user to prespecify the size and shape of the solution. Functions from the Logo programming language and Lindenmayer systems enable genetic programming to draw the antenna. The solution evolved by genetic programming possesses the essential characteristics of the Yagi-Uda type of antenna. The rediscovery by genetic programming of the essential characteristics of the Yagi-Uda antenna is an instance where genetic programming has produced a result that is competitive with a result produced by creative and inventive humans.

1 Introduction

An antenna is a device for receiving or transmitting electromagnetic waves. An antenna may receive an electromagnetic wave and transform it into a signal on a transmission line. Alternately, an antenna may transform a signal from a transmission line into an electromagnetic wave that is then propagated in free space.

Some antennas are directional in the sense that they send and receive a signal primarily in a specified direction. Some operate only over a narrow band of frequencies, while others operate over a wide band of frequencies. Antennas vary as to their impedance at their feed points and the polarization of the waves that they transmit or receive. The antenna's gain is usually a major consideration.

Maxwell's equations govern the electromagnetic waves generated and received by antennas. The task of analyzing the characteristics of a given antenna is difficult. The task of synthesizing the design of an antenna with specified characteristics is even more difficult and typically calls for considerable creativity on the part of the antenna engineer (Balanis 1982; Stutzman and Thiele 1998; Linden 1997).

1.1 Numerical Electromagnetics Code

The behavior and characteristics of many antennas can be determined by simulation. For example, the *Numerical Electromagnetics Code* (NEC) is a method-of-moments (MoM) simulator for wire antennas that was developed at the Lawrence Livermore National Laboratory (Burke 1992).

The NEC simulator is very general and reasonably fast. It works from relatively simple text input and produces output in a well-defined (text) format. The NEC simulator is widely used in the antenna community and is considered to be reasonably accurate and reliable for a broad range of structures (Linden 1997). The availability of the source code (in FORTRAN) for the NEC simulator (subject to U. S. export controls for version 4) enables the simulator to be efficiently embedded inside a run of an algorithm for genetic and evolutionary computation.

In regard to the above characteristics, the NEC simulator is similar to the SPICE simulator (Quarles, Newton, Pederson, and Sangiovanni-Vincentelli 1994) that we have previously used for simulating analog electrical circuits (Koza, Bennett, Andre, and Keane 1999) and controllers (Koza, Keane, Yu, Bennett, and Mydlowec 2000).

1.2 Previous Work on Antenna Design using Evolutionary Computation

Genetic algorithms have been successfully applied to the design of antennas, including the design of thinned arrays (Haupt 1994), wire antennas (Linden 1997; Altshuler and Linden 1997, 1998), patch antenna (Johnson and Rahmat-Samii 1999), and linear and planar arrays (Marcano and Duran 1999). Jones (1999) applied genetic programming to antenna design. The book *Electromagnetic Optimization by Genetic Algorithms* (Rahmat-Samii and Michielssen 1999) describes numerous applications of the genetic algorithm to antenna design.

1.3 Genetic Programming

Genetic programming (Koza 1992; Koza and Rice 1992; 1994a, 1994b) is a technique for automatically creating computer programs to solve, or approximately solve, problems. Genetic programming is an extension of the genetic algorithm (Holland 1975). Genetic programming is capable of synthesizing the design of complex structures from a high-level statement of the structure's desired behavior and characteristics. The complex structures include analog electrical circuits (Koza, Bennett, Andre, and Keane 1999; Koza, Bennett, Andre, Keane, and Brave 1999) and controllers (Koza, Keane, Yu, Bennett, and Mydlowec 2000).

Additional information on genetic programming can be found in books such as Banzhaf, Nordin, Keller, and Francone 1998; books such as Langdon 1998, Ryan 1999, and Wong and Leung 2000 in the series on genetic programming from Kluwer Academic Publishers; in edited collections of papers such as the *Advances in Genetic Programming* series of books from the MIT Press (Spector, Langdon, O'Reilly, and Angeline 1999); in the proceedings of the Genetic Programming Conference (Koza, Banzhaf, Chellapilla, Deb, Dorigo, Fogel, Garzon, Goldberg, Iba, and Riolo 1998); in the proceedings of the Euro-GP conference (Poli, Banzhaf, Langdon, Miller, Nordin, and Fogarty 2000); in the proceedings of the Genetic and Evolutionary Computation Conference (Banzhaf, Daida, Eiben, Garzon, Honavar, Jakiela, and Smith 1999); at web sites such as `www.genetic-programming.org`; and in the *Genetic Programming and Evolvable Machines* journal (from Kluwer Academic Publishers).

1.4 Roadmap for this Paper

Section 2 describes an illustrative problem of antenna design. Section 3 describes one way by which genetic programming may be applied to the design of antennas, including a repertoire of functions and terminals based on the Logo programming language and Lindenmayer systems. Section 4 itemizes the preparatory steps necessary to apply genetic programming to an illustrative problem. Section 5 presents the results.

2 Illustrative Problem

We illustrate the use of genetic programming for the design of antennas with an illustrative problem that has been previously solved by both conventional antenna design techniques and the genetic algorithm operating on fixed-length character strings (Linden 1997; Altshuler and Linden 1999).

The problem is to synthesize the design of a planar symmetric antenna composed of wires of a half millimeter radius that

- has maximum gain in a preferred direction (specifically, along the positive X-axis) over a range of frequencies from 424 MHz to 440 MHz,
- has reasonable value (specifically ≤ 3.0) for voltage standing wave ratio (VSWR) when the antenna is fed by a transmission line whose characteristic impedance, Z_0, is 50 Ω,
- fits into a bounding rectangle whose height is 0.4 meters and whose width is 2.65 meters, and
- is excited by a single voltage source. One end of the transmission line is connected to the source and the other end is connected (at the origin of a coordinate system) to the antenna's driven element. The lower left corner of the bounding rectangle is positioned at (-250, -200) and the upper right corner is at (2400, 200).

The above requirements can be satisifed by a Yagi-Uda antenna (Uda 1926, 1927; Yagi 1928). A Yagi-Uda antenna (figure 1) is a planar symmetric wire antenna consisting of a number of parallel linear elements. The Yagi-Uda antenna is widely used for home TV antennas. The *driven element* is the vertical linear element positioned along the Y-axis. The midpoint of the driven point is connected to a transmission line (at the origin). The other elements of the antenna are not connected to the transmission line. Instead, their currents are induced parasitically by mutual coupling (Balanis 1982). All of the antenna's elements are symmetric (about the X-axis). The elements to the left of the driven element act as *reflectors*. Usually (but not necessarily) there is just one reflector. The (numerous) elements to the right of the driven element act as *directors*. The antenna in the figure has four directors. The directors are typically spaced unequally, shorter than the driven element, and of different lengths. A well-designed Yagi-Uda antenna is an *endfire* antenna in that it directs most of its energy toward a point in the farfield of the antenna (along the positive X-axis here). The driven element, directors, and reflector(s) of a Yagi-Uda antenna are attached to a non-conducting physical support (not shown).

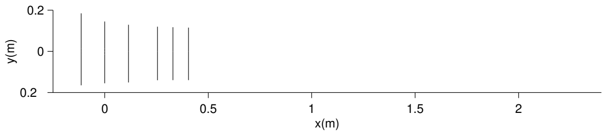

Figure 1 Example Yagi-Uda antenna.

Before applying the genetic algorithm operating on fixed-length character strings to a particular problem, the human user of the algorithm must perform certain preparatory steps. These preparatory steps include choosing the representation scheme by which the to-be-discovered parameters are mapped onto the chromosome string. Typically, the representation scheme for the genetic algorithm involves choosing the number of characters in the alphabet for the chromosome string, the number of characters that are allocated to each variable, and the location of each variable on the chromosome string (Goldberg 1989).

When the genetic algorithm was used on the above illustrative problem, the human user (Linden 1997; Altshuler and Linden 1999) prespecified the

(1) the number of directors,

(2) the number of reflectors (one),

(3) the fact that the driven element, the directors, and the reflector are all single straight wires,

(4) the fact that the driven element, the directors, and the reflector are all arranged in parallel,

(5) the fact that the energy source (via the transmission line) is connected only to the driven element — that is, the directors and reflectors are parasitically coupled.

After preordaining the above five characteristics of the size and shape of the solution, a chromosome string was constructed containing the length of the wires and spacing between the wires. In contrast, when genetic programming is applied to this problem, the human user does not prespecify any of the above characteristics.

3 Designing Wire Antennas using Genetic Programming

Our approach to the synthesis of a wire antenna involves creating a two-dimensional drawing of the antenna. This can be accomplished using turtle geometry (Abelson and diSessa 1980), including certain features of the Logo programming language, certain features of Lindenmayer systems (Lindenmayer 1968; Prusinkiewicz and Hanan 1980; Prusinkiewicz and Lindenmayer 1990), and certain of functions of our own design. The turtle may (or may not) deposit ink (metal) as it moves. The Logo programming language is a dialect of LISP that was created in 1967 by Seymour Papert, Wallace Feurzeig, and his group at Bolt, Beranek and Newman. We use functions patterned after functions from Logo that turn and move the turtle and that iteratively execute a group of functions. We also use a function that rubber-bands the turtle in the same manner as the brackets of Lindenmayer systems. See also Koza 1993.

3.1 Repertoire of Functions

Eight functions are used in the program trees.

The one-argument `TURN-RIGHT` function changes the facing direction of the turtle by turning the turtle clockwise by the amount specified by its argument. The argument is a floating-point number between 0.0 and 1.0. The argument is multiplied by 2π so that it represents an angle in radians.

The two-argument `DRAW` function moves the turtle in the direction that it is currently facing by an amount specified by its first argument. The turtle may or may not deposit a wire as it is moving, depending on its second argument. Drawing with `HALF-MM-WIRE` is equivalent to depositing a half millimeter wire while moving the turtle. Drawing with `NOWIRE` is equivalent to moving the turtle without depositing metal. The first argument is a floating-point number between 0.0 and 1.0. This number is scaled between two bounds (16 to 200 millimeters for this problem) to avoid unreasonably long wires and unreasonably short wires (which sometimes lead to simulator errors or represent an implausible antenna geometry).

The two-argument `REPEAT` function causes its second argument subtree to be executed for the number of times specified by its first argument. The first argument is an integer modulo 100 (to avoid infinite loops).

The one-argument `LANDMARK` function causes the turtle to execute its single argument. The turtle is then restored to the position and facing direction that it had at the start of the evaluation of the `LANDMARK` function. The `LANDMARK` function operates in the same manner as brackets of Lindenmayer systems (Lindenmayer 1968; Prusinkiewicz and Hanan 1980; Prusinkiewicz and Lindenmayer 1990).

The two-argument `TRANSLATE-RIGHT` function does three things. First, this function temporarily turns the turtle clockwise by the amount specified by its first argument (in the same manner as the argument of the `TURN-RIGHT` function). Second, this function moves the turtle forward (in the direction in which the turtle has been temporarily turned) by the amount specified by its second argument (in the same manner as the `DRAW` function with `NOWIRE` as its second argument). Third, this function restores the turtle to its original facing direction.

The connective functions `PROGN2`, `PROGN3`, `PROGN4` sequentially execute their 2, 3, and 4 arguments (respectively).

3.2 Repertoire of Terminals

Four terminals are used in the program trees. None of the functions return any value.

The terminal `HALF-MM-WIRE` denotes a half millimeter wire. The terminal `NOWIRE` represents the absence of a wire. A constrained syntactic structure restricts the location of these two terminals to the second argument of the `DRAW` function.

$\Re_{real}$ denotes floating-point numbers between 0.0 and 1.0. These terminals can appear only as the first argument of the `TURN-RIGHT` and `DRAW` functions.

$\Re_{integer}$ denotes integers between 0 and 99. These terminals can appear only as the first argument of a `REPEAT` function. These terminals can appear only as the first argument of the `REPEAT` function.

The `END` terminal appears at all other leaves (endpoints) of the program tree.

4 Preparatory Steps

Before applying genetic programming to a particular problem, the user must perform six major preparatory steps, as itemized below.

4.1 Program Architecture

Each individual program in the population has one result-producing branch. Automatically defined functions are not used.

4.2 Function Set

The function set for the result-producing branch is

F_{rpb} = {`TURN-RIGHT`, `DRAW`, `REPEAT`, `LANDMARK`, `TRANSLATE-RIGHT`, `PROGN2`, `PROGN3`, `PROGN4`}.

4.3 Terminal Set

As previously mentioned, a constrained syntactic structure is used to restrict certain terminals (`WIRE`, `NOWIRE`, $\Re_{real}$, and $\Re_{integer}$) to be certain arguments of certain functions. For all other parts of the program tree, the terminal set is simply

T_{rpb} = {`END`}.

4.4 Fitness Measure

Each antenna begins with a 7.5-millimeter straight piece of wire beginning at the origin and lying along the positive Y-axis. This stub becomes part of the antenna's driven element. The driven element is excited by a voltage source applied at (0,0). The turtle starts at position (0, 7.5) with a facing direction of north (i.e., along the positive Y-axis).

Each program tree in the population is a composition of the above functions and terminals. The program tree is executed in the usual depth-first order of evaluation (from the left). The number, location, and shape of the antenna's elements are determined by the program tree. As the turtle moves, it may (or may not) deposit metal in the form of straight pieces of wire. Separate elements can be created by the `DRAW` function (when it is executed with the `NOWIRE` argument) and by the `TRANSLATE-RIGHT` function.

Because the statement of this problem calls for the antenna to fit inside a specified area, after execution of the entire program tree, any metal that has been deposited outside the boundary of a clipping rectangle is deleted. The lower left corner of the clipping rectangle for this problem is positioned at (-250, 0) and the upper right corner is positioned at (2400, 200).

Because the statement of this problem calls for a symmetric antenna, all metal deposited above the X-axis is, after clipping, duplicated by reflecting it across the X-axis. The result is the desired symmetric antenna lying inside the bounding rectangle (which is twice the height, but equal in width, to the clipping rectangle).

The geometry of the antenna is specified by a data structure containing the coordinates of each wire and the radius of each wire (one half millimeter here).

We embedded version 4 of the *Numerical Electromagnetics Code* antenna simulator in our genetic programming software. The input to NEC consists of text information that describes the geometry of the antenna, the number of segments into which each wire is partitioned for the method of moment calculation, information about the means of excitation, the output, information about the ground plane (or lack thereof), and various commands for controlling the simulation.

Fitness is a linear combination of VSWR and gain.

The VSWR is a measure of how much of the input energy from the source is reflected back down the transmission line from the antenna (rather than radiated by the antenna). The NEC code calculates the complex input impedance, Z, at the voltage source.

$$VSWR = \frac{(1+|R|)}{(1-|R|)}$$

Here R is the reflection coefficient computed by

$$R = \frac{(Z - Z_0)}{(Z + Z_0)}$$

where Z_0 is the characteristic impedance of the transmission line feeding the antenna.

The value of VSWR ranges from 1 (representing no reflection — that is, all energy is radiated), to infinity (i.e., all energy reflected and there is no radiation). In this optimization the maximum value of the VSWR is limited to 2×10^8.

The NEC simulator was instructed to compute the farfield radiation pattern at θ = 90 and ϕ = 0. θ is measured from the positive Z-axis to the X-Y-plane, while ϕ is measured from the positive X-axis to the positive Y-axis. The value for the antenna gain is the magnitude (in decibels relative to an isotropic radiator) of the farfield radiation pattern at θ = 90 and ϕ = 0.

The accuracy of the simulation can be significantly affected by the number of segments into which each straight wire in the antenna is divided for purpose of simulation. There is no a priori way to compute the ideal number of segments to achieve an accurate simulation. Moreover, contrary to intuition, the largest number of segments does not necessarily produce the most accurate simulation. Therefore, we performed the NEC simulation several ways for each antenna and used the worst outcome. Specifically, we performed the simulation using both 15 and 25 as the number of segments per wavelength. In addition, we performed the simulation using both 1 and 2 as the minimum number of segments per straight wire section. (To save computer time, we omitted any simulation with a duplicated segmentation). In addition, each of these four simulations was performed at 3 frequencies (424, 432, 440 MHz). The simulator calculates the current at the center of each segment. The value of the VSWR that we used in the fitness calculation is the maximum over all of the above 12 cases. The value of the gain, *G*, is the minimum value over all of the above 12 cases.

Fitness is $-G + C * VSWR$. A smaller fitness is better. The constant *C* is 0.1 when $VSWR \leq 3.0$ and is 10 when $VSWR > 3.0$. This is the same fitness measure used by Altshuler and Linden (1999), except that it has a slightly heavier penalty for poor *VSWR*. We increased the penalty because an antenna with poor *VSWR* can produce an artificially high gain (offsetting the VSWR in the overall fitness). *G* appears with a negative sign above because greater gain is more desirable.

As required by the NEC simulator, each pair of intersecting wires is replaced with four new wires (each with an endpoint at the intersection point) prior to running the simulator. In addition, each antenna geometry is pre-checked by a various rules. If the length of each segment divided by its radius is not greater than 2.0 or the center of each segment is not at least 4 wire radii from the axis of every other wire, then the individual is either replaced (for generation 0) or assigned a high penalty value of fitness (10^8).

If more than 10,000 functions are executed, execution of the individual program tree is terminated and the individual is assigned a high penalty value of fitness (10^8).

Antennas that cannot be simulated receive a high penalty value of fitness (10^8).

4.5 Control Parameters

The population size, *M*, was 500,000. A (generous) maximum size of 500 points (i.e., total number of functions and terminals) was established for the result-producing branch. The percentages of the genetic operations are 60% one-offspring crossover on internal points of the program tree other than numerical constant terminals, 10% one-offspring crossover on points of the program tree other than numerical constant terminals, 1% mutation on points of the program tree other than numerical constant terminals, 20% perturbation on numerical constant terminals, and 9% reproduction. The other parameters are the same default values that we have used previously on a broad range of problems (Koza, Bennett, Andre, Keane 1999).

4.6 Termination

The run was manually terminated when the values of fitness for successive best-of-generation individuals appeared to have reached a plateau.

4.7 Parallel Implementation

After code development using a single Pentium computer, this problem was run on a home-built Beowulf-style (Sterling, Salmon, Becker, and Savarese 1999) parallel cluster computer system consisting of 1,000 350 MHz Pentium II processors (each accompanied by 64 megabytes of RAM). The system has a 350 MHz Pentium II computer as host. The processing nodes are connected with a 100 megabit-per-second Ethernet. The processing nodes and the host use the Linux operating system. The distributed genetic algorithm with unsynchronized generations and semi-isolated subpopulations was used with a subpopulation size of Q = 500 at each of D = 1,000 demes. As each processor (asynchronously) completes a generation, four boatloads of emigrants from each subpopulation are dispatched to each of the four toroidally adjacent processors. The 1,000 processors are hierarchically organized. There are $5 \times 5 = 25$ high-level groups (each containing 40 processors). If the adjacent node belongs to a different group, the migration rate is 2% and emigrants are selected based on fitness. If the adjacent node belongs to the same group, emigrants are selected randomly and the migration rate is 5% (10% if the adjacent node is in the same physical box).

5 Results

Figure 2 shows a best-of-node antenna from generation 0 of our one and only run of this problem. . It consists of only a driven element and has a fitness of -2.03.

Figure 3 shows another best-of-node antenna from generation 0. It consists of driven element and a V-shaped element. It has fitness of -3.82.

Figure 4 shows yet another best-of-node antenna from generation 0. It consists of numerous straight wires and V-shaped elements. It has fitness of -4.43.

Figure 5 shows the best-of-generation antenna from generation 2 (resembling a star burst with both straight wires and V-shaped elements). It has fitness of -5.18.

The best-of-generation antenna from generation 9 (figure 6) has fitness of -7.58.

The best-of-generation antenna from generation 47 (figure 7) has fitness of -14.13.

The best-of-run antenna emerged in generation 90 (figure 8). It has fitness of -16.04.

Table 1 compares the conventional Yagi-Uda antenna, the antenna created by the genetic algorithm (Linden 1997; Altshuler and Linden 1997, 1998), and the antenna created by genetic programming. All three are satisfactory solutions to the problem here.

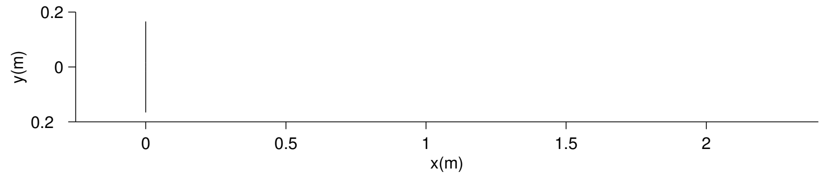

Figure 2 First example of a best-of-node antenna from generation 0.

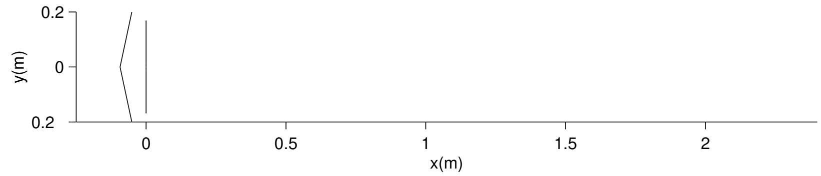

Figure 3 Second example of a best-of-node antenna from generation 0.

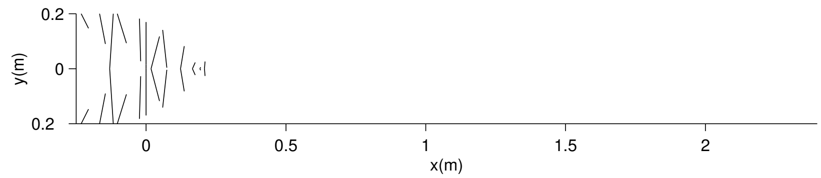

Figure 4 Third example of a best-of-node antenna from generation 0.

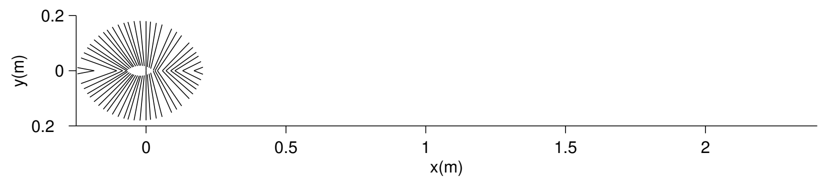

Figure 5 Best-of-generation antenna from generation 2.

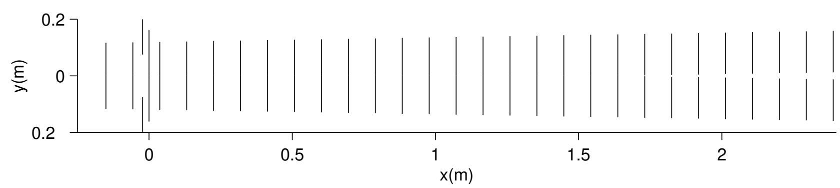

Figure 6 Best-of-generation antenna from generation 9.

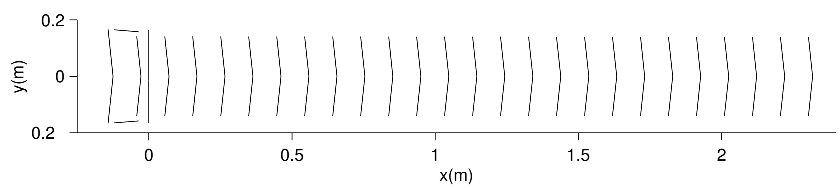

Figure 7 Best-of-generation antenna from generation 47.

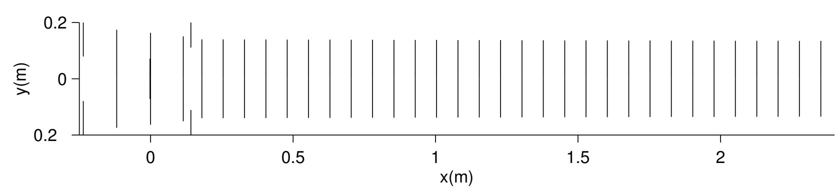

Figure 8 Best-of-run antenna from generation 90.

Table 1 Comparison of Yagi-Uda, Altshuler-Linden, and genetic programming.

Frequency (MHz)	Yagi-Uda		Altshuler-Linden		GP	
	Gain (dBi)	VSWR	Gain (dBi)	VSWR	Gain (dBi)	VSWR
424	15.5	1.41	15.4	1.88	16.3	2.19
428	15.8	1.11	16.0	1.80	16.4	2.00
432	15.9	1.23	16.3	1.09	16.4	2.02
436	15.7	1.60	16.1	9.50	16.5	2.16
440	15.5	1.85	9.4	39.00	16.3	2.36

The conventional Yagi-Uda antenna and the antennas created by the genetic algorithm and genetic programming are all approximately the same length (i.e., about 3.6 wavelengths along the X-axis).

Note that all five of the characteristics enumerated in section 2 above were automatically created during the run of genetic programming. In contrast, all five of these characteristics were prespecified by the human user prior to run of the genetic algorithm.

Characteristics (3), (4), and (5) are essential characteristics of the Yagi-Uda antenna (Uda 1926, 1927, Yagi 1928), namely an antenna with multiple parallel parasitically coupled straight-line directors, a single parallel parasitically coupled straight-line reflector, and a straight-line driven element.

The Yagi-Uda antenna was considered an achievement in its field at the time it was first invented. As Balanis (1982) observed,

> "Although the work of Uda and Yagi was done in the early 1920s and published in the middle 1920s, full acclaim in the United States was not received until 1928 when Yagi visited the United States and presented papers at meetings of the Institute of Radio Engineers (IRE) in New York, Washington, and Hartford. In addition, his work was published in the *Proceedings of the IRE*, June 1928, where J. H. Dellinger, Chief of Radio Division, Bureau of Standards, Washington, D. C., and himself a pioneer of radio waves, wrote 'I have never listened to a paper that I left so sure was destined to be a classic.' So true!!"

We claim that the reinvention by genetic programming of the essential characteristics of the Yagi-Uda antenna in solving the illustrative problem herein is an instance where genetic programming has produced a result that is competitive with a result produced by creative and inventive humans.

5.1 Computer Time

The best-of-run individual from generation 90 was produced after evaluating 4.55×10^6 individuals (500,000 times 91). This required 22 hours (7.92×10^4 seconds) on our 1,000-node parallel computer system — that is, the expenditure of 2.772×10^{16} computer cycles (about 28 peta-cycles of computer time).

References

Abelson, Harold and diSessa, Andrea. 1980. *Turtle Geometry*. Cambridge, MA: The MIT Press.

Altshuler; Edward E. and Linden; Derek S. 1998. *Process for the Design of Antennas using Genetic Algorithm*. United States Patent 5,719,794. Applied for on July 19, 1995. Issued on February 17, 1998.

Altshuler; Edward E. and Linden; Derek S. 1999. Design of wire antennas using genetic algorithms. In Rahmat-Samii, Yahya and Michielssen, Eric (editors). *Electromagnetic Optimization by Genetic Algorithms*. New York, NY: John Wiley & Sons. Chapter 8. Pages 211 - 248.

Balanis, Constantine A. 1982. *Antenna Theory: Analysis and Design*. New York, NY: John Wiley.

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.

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

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

Burke, Gerald J. 1992. *Numerical Electromagnetics Code — NEC-4: Method of Moments — User's Manual*. Lawrence Livermore National Laboratory report UCRL-MA-109338. Livermore, CA: Lawrence Livermore National Laboratory.

Goldberg, David E. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley 1989.

Haupt, Randy L. Thinned arrays using genetic algorithms. 1994. *IEEE Transactions on Antennas and Propagation*. 42: 993 - 999.

Holland, John H. 1975. *Adaptation in Natural and Artificial Systems*. 2nd. Ed. Cambridge, MA: MIT Press.

Johnson, J. Michael and Rahmat-Samii, Yahya. 1999. Genetic algorithms and method of moments (GA/MOM) for the design of integrated antennas. *IEEE Transactions on Antennas and Propagation*. 47(10) 1606 - 1614. October 1999.

Jones, Eric A. 1999. *Genetic Design of Antennas and Electronic Circuits*. PhD Thesis. Department of Electrical and Computer Engineering. Duke University.

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

Koza, John R. 1993. Discovery of rewrite rules in Lindenmayer systems and state transition rules in cellular automata via genetic programming. Symposium on Pattern Formation (SPF-93), Claremont, California. February 13, 1993.

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., 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., 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.

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

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

Linden, Derek S. 1997. *Automated Design and Optimization of Wire Antennas Using Genetic Algorithms*. Ph.D. thesis. Department of Electrical Engineering and Computer Science. Massachusetts Institute of Technology.

Lindenmayer, Aristid. Mathematical models for cellular interactions in development, I & II. *Journal of Theoretical Biology*. 18: 280–315. 1968.

Marcano, Diogenes and Duran, Filinto. 1999. Synthesis of linear and planar arrays using genetic algorithms. In Rahmat-Samii, Yahya and Michielssen, Eric (editors). *Electromagnetic Optimization by Genetic Algorithms*. New York, NY: John Wiley & Sons. Chapter 6. Pages 157 - 179.

Prusinkiewicz, Przemyslaw and Hanan, James. 1980. *Lindenmayer Systems, Fractals, and Plants*. New York: Springer-Verlag.

Prusinkiewicz, Przemyslaw, and Lindenmayer, Aristid. 1990. *The Algorithmic Beauty of Plants*. New York: 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.

Quarles, Thomas, Newton, A. R., Pederson, D. O., and Sangiovanni-Vincentelli, A. 1994. *SPICE 3 Version 3F5 User's Manual*. Department of Electrical Engineering and Computer Science, University of California. Berkeley, CA. March 1994.

Rahmat-Samii, Yahya and Michielssen, Eric (editors). 1999. *Electromagnetic Optimization by Genetic Algorithms*. New York, NY: John Wiley & Sons.

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

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

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.

Stutzman, Warren. L. and Thiele, Gary A. 1998. *Antenna Theory and Design*. Second edition. New York, NY: John Wiley.

Uda, S. 1926. Wireless beam of short electric waves. *Journal of the IEE (Japan)*. March 1926. 273 - 282.

Uda, S. 1927. Wireless beam of short electric waves. *Journal of the IEE (Japan)*. March 1927.1209-1219.

Wong, Man Leung and Leung, Kwong Sak. 2000. *Data Mining Using Grammar Based Genetic Programming and Applications*. Amsterdam: Kluwer Academic Publishers.

Yagi, H. 1928. Beam transmission of ultra short waves. *Proceedings of the IRE*. 26: 714-741. June 1928.