Automatic Synthesis of an 802.11a Wireless LAN Antenna using Genetic Programming

Rian Sanderson
CS426 Stanford University
Texas Instruments Wireless LAN Business unit
Santa Rosa California
rsanders@alumni.calpoly.edu

ABSTRACT

Describes the application of genetic programming to synthesize a small form factor, 2 dimensional wire antenna for a 5 GHz 802.11a application. Discusses a novel approach to quantizing the search space which leads to a more simple representation. Discusses implementation details and demonstrates promising results.

1. Introduction

Many people have used evolutionary algorithms to synthesize antennas. Evolutionary algorithms are well suited to this problem domain due to the ready availability of simulators and supporting tools, the large search space, and the nature of the art of RF engineering. This fit to the problem domain can be taken a step further by representing antenna geometries with an implementation of the LOGO drawing language. LOGO instructions are easy to draw, easy to parse, and the physicality of its representation has an appealing intuitive feel.

While other literature has focused on 440Mhz Yagi antennas, this paper looks at a 5220 MHz antenna for an 802.11a wireless LAN application. An efficient antenna is an important part of an 802.11 system; it can extend the reach of signals, or extend battery life by requiring less transmit power. The Pringles can and biquad are popular do-it-yourself antenna designs for 802.11b in the 2.4 GHz spectrum. However, their large form factors are inconvenient for mobile use, and these popularized designs don't work well in the 5GHz spectrum. An ideal antenna could be easily carried around in a computer bag and perched on top of a notebook computer.

2. Statement of the problem

The problem is to synthesize a two dimensional continuous wire antenna for the Lower UNI band (frequency range 5180 – 5240 MHz) which is no larger than 6 cm square, and made of American wire gauge (AWG) #19 wire. The antenna is simulated using the freely available NEC2 simulator at a single representative frequency of 5220 MHz, in free space, assuming a lossless conductor.

Rather than using real valued quantities for wire sections and angles, the 6 cm square problem space is discretized into a matrix of points. As Figure 1 shows, a continuous wire antenna can be viewed as a collection of small wires each joined at a point in the matrix.

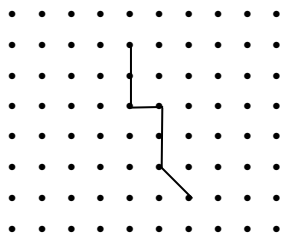

Figure 1: An antenna viewed as the connection of six individual segments.

While NEC has many conditions regarding allowable geometries with respect to wavelength, wire radii, and segment length, by spacing the matrix points appropriately the number of unsimulatable individuals can be minimized. For this problem, with its 5.7 cm wavelength signal on AWG #19 wire, 4 mm is the minimum spacing. The only simulator

precondition that may be violated is the crossing of two 45 degree wires. However, this precondition is not a hard rule; most of the time the simulator cannot calculate the current on overlapping segments due to an indeterminate matrix (NEC manual pg 9), but sometimes it can. When it cannot calculate due to an intersection the simulation fails and writes no output.

The exact size of the problem space is difficult to calculate, but a conservative estimate is 4! * (14*14)!, or 1.2e+367. This is the number of combinations of four wire connections in each square formed by four points. It leaves out two possible wire connections in the bottom right corner, one wire connection for each bottom squares, and one wire connection for each of the right hand squares. However, these omissions seem insignificant at this order of magnitude.

3. Methods

Objective:	The objective is to synthesize a two dimensional continuous wire antenna for the Lower UNI band (frequency range 5180 – 5240 MHz) which is no larger than 6 cm square, and made of American wire gauge (AWG) #19 wire.
Architecture:	One result producing branch
Functions:	progn2(), progn3()
Terminals:	move, rot 45, rot 90, rot 315, rot 270
Fitness Case:	Simulation with NEC2 at a single frequency of 5220 MHz, generating radiation data for 360 degrees of phi and theta at 5 degree increments.
Raw Fitness:	-maxgain + V(vswr) ; smaller fitness is better. where V(vswr)= vswr *C, with C= .1 for vswr <= 2, C= 1 for vswr <=3, C=10 for vswr > 3
Standardized Fitness:	12 + Raw Fitness
Selection:	separate runs: roulette wheel with .6 p cross over and .1 p mutation; tournament with size of 7 .9 p cross over, .1 p mutation
Hits:	not used
Parameters:	typical: M= 1000; G=108
Success Predicate:	Raw Fitness <= -12

Table 1: Tableau

The simplicity of the genetic programming architecture left time for other complications of implementation. While not strictly necessary, the four rotate terminals seem appropriate when looking at the problem from a human's point of view. In the actual results, rotate operators are often stacked five or more deep, causing the turtle to spin several revolutions to change perhaps 45 degrees.

The fitness case is far from rigorous, but lends itself to quick computation. Due to the representation of the problem the segment size cannot be changed without increasing the bounding area for the antenna or decreasing the wire size. It remains to been seen how well these antennae would perform when physically built.

The fitness measure is based on Koza's work, though an even higher penalty for poor VSWR was assessed. In 802.11 systems an antenna with good VSWR is critical. Standardized fitness was achieved by using twelve as an estimate of the best possible fitness. This proved to be too low of a number and was modified in later runs.

Roulette wheel and tournament selection are used in separate runs and are discussed in the results section.

The parameters for M and G are chosen with the length of the run in mind. Evaluation of a single individual averaged .4 seconds and evaluating one thousand individuals for 108 generations took about 12 hours.

4. Structures Undergoing Adaptation

The trees of LOGO instructions that actually undergo adaptation are much easier to view in evaluated antenna forms. Figure 2 shows a portion of a the program tree for a random individual of generation 0.

```
(progn2 (progn2 (progn3 (progn2 (progn2 rotxy 90 rotxy 90)
(progn3 rotxy 90 rotxy 45 move))
(progn3 (progn3 rotxy 270  rotxy 45 rotxy 270 )
(progn2 rotxy 315  rotxy 90)
(progn3 move move rotxy 315 ))
(progn2 (progn3 rotxy 270  rotxy 270  move)
```

```
(progn3 rotxy 270  move rotxy 270 )))
(progn3 (progn2 (progn3 rotxy 315  rotxy 270  move)
(progn2 rotxy 90 move))
(progn3 (progn2 move move)
(progn3 rotxy 90 move rotxy 270 )
(progn3 rotxy 90 rotxy 270  rotxy 45))
```

Figure 2: A portion of a program tree from generation 0

The individual in Figure 2, like most others, wastes time spinning around in circles. Figure 3 shows a different randomly evaluated individual from generation 0.

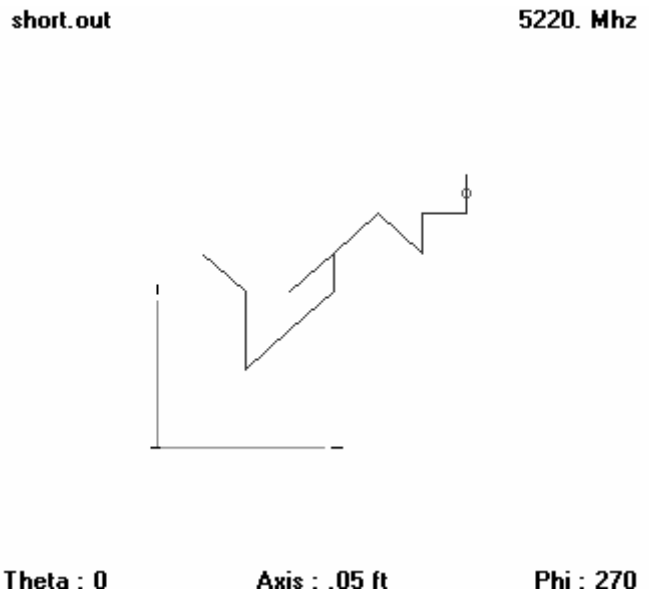

Figure 3: Random individual from generation 0 with raw fitness of 2395.21

A wire is created between two points upon the execution of a move instruction. Rotate instructions don't actually create wires, they only select which point the next move instruction will connect to. The starting wire is where the excitation signal is fed to the antenna, and is represented with a small circle.

The individual in Figure 3 has reasonable gain of 4.79 db, but its dismal VSWR of 240 contributes most to its raw fitness of 2395.21. It is generated from a tree that condenses into: rot 90, move, rot 270, move, rot 90, rot 45, move, rot 270, move, move, rot 90, rot90, move,rot 90, rot 45, move, rot 270, move, move, rot 90, rot 45, move, move, rot 315, move. Note the double back at the Y in the middle of the antenna. The simulator ignores two wires with the same coordinates.

An individual at generation 11 of the same run, Figure 4, performs quite well despite its strange structure. The removal of just one section at the tail of the arrow created the best of run individual shown in Figure 5.

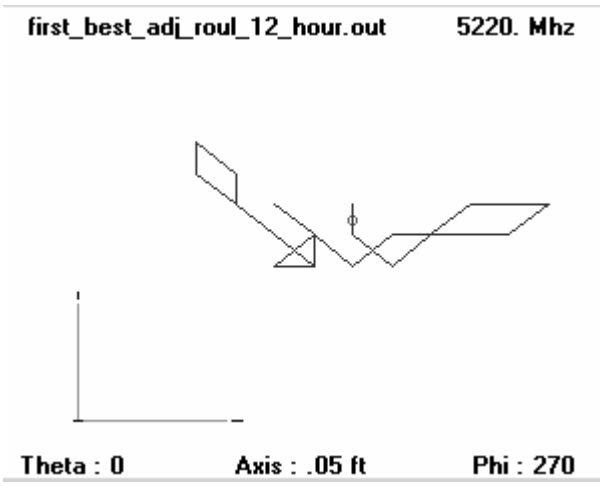

Figure 4: The Arrow, at gen. 11, has a gain of 13.4 db and VSWR of 2.3 for a raw fitness of -11.1

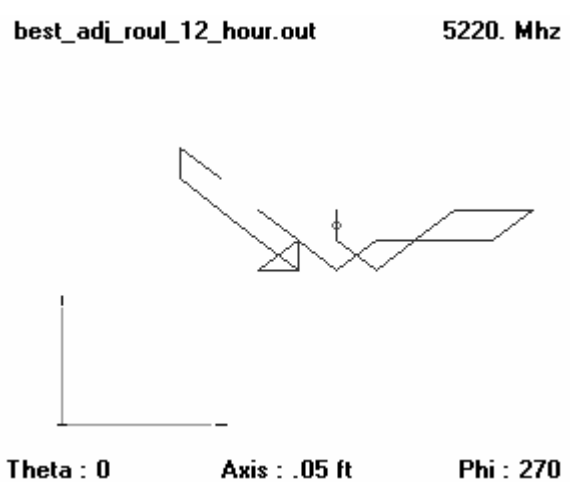

Figure 5: Best of run, the Arrow from gen. 14, has a gain of 12.9 db but a much improved VSWR of 1.65 for a raw fitness of -12.7

The best individual from this run, Figure 5, has 300 nodes and a depth of 7 and raw fitness of -12.7. The improved VSWR more than offsets its slightly poorer gain. This is certainly not the kind of antenna a human would design, but it works quite well.

5. Implementation

A foundation of this project is the quantizing of the problem space. In conjunction with the scaling feature of NEC, direct matrix points can be written to the NEC input file creating a one to one mapping from the theoretical points of the matrix and their actual mapping in the input file. Not having to scan the simulator input for proper semantics saves considerable implementation time. The discretization also fixes the two variables of wire size and segment length, further narrowing the dimensionality of the problem.

The evaluation of an individual is staged over several steps. The data flow starts with evaluating the GP tree to generate a LOGO instruction file as show in Figure 6.

```
rotxy 315
rotxy 90
rotxy 45
rotxy 45
rotxy 315
rotxy 315
move
rotxy 270
rotxy 270
move
rotxy 90
rotxy 45
move
rotxy 270
```

Figure 6: LOGO instructions generated from the GP tree.

The parser converts from LOGO to NEC input, shown in Figure 7. The first wire is a stub for the excitation connection. It allows for an individual which is made up of only rotate instructions.

```
CM GP Generated nec input
CM Using screen coords.  Excite at first wire
CM dimensions in mm, scaled to meters with GS
```

```
CM
CM Rian Sanderson Copyright (c) 2003
CM CS426 Autumn 2003
CM
CM  TAG   #Segs    x1      y1      z1 x2     y2      z2  wire radius
CE
GW  1       1       7       7       0 7       6       0 .120
GW    2    1    7    6    0    6    6    0    .120
GW    3    1    6    6    0    6    5    0    .120
GW    4    1    6    5    0    5    6    0    .120
GW    5    1    5    6    0    4    5    0    .120
GW    6    1    4    5    0    3    4    0    .120
GW    7    1    3    4    0    4    5    0    .120
GW    8    1    4    5    0    4    4    0    .120
GW    9    1    4    4    0    3    3    0    .120
GW    10    1    3    3    0    2    2    0    .120
GW    11    1    2    2    0    2    3    0    .120
GW    12    1    2    3    0    2    4    0    .120
GW    13    1    2    4    0    1    5    0    .120
GS  0   0   .004
GE
EX  0       1       1       0       1       0
FR  0       1       0       0       5220
RP 0 73 73 1000 -180.0 0.0 5.0 5.0
EN
```

Figure 7: GP generated NEC input file

The simulator uses the NEC input to generates a .out file, which, finally, the fitness evaluator parses to compute a raw fitness. The data flow is disk intensive, but writing to a file at each stage made modular development and discrete unit testing possible.

Error checking on file opening and closing, programming niceties usually shirked by most programmers, turned out to be important implementation details. Early runs failed due to a filled user disk quota and segmentation faults after opening too many file handles.

The most egregious implementation error was not explicitly calculating standardized fitness. When 228 hours of computer time, in ten different runs, yielded only mediocre results I became suspicious. On closer examination all of the raw fitnesses hovered right around -.9, yet never broke past. I was concerned only with raw fitness, but Lil-gp uses standardized fitness when choosing individuals to breed. These early runs calculated only raw fitness, leaving some copy/pasted calculation, from another problem, for standardized and adjusted fitness. After calculating standardized fitness selection occurred more appropriately and it bred individuals with fitness fifteen times better.

Other implementation details:
Hardware: eight Pentium III PCs running Debian Linux
Computer use: one run per computer. No clustering. No farming out of simulation runs.
Simulator: nec2
Genetic Programming Software: lil-gp version 1.1
individual evaluation time: .4 seconds
evaluating 1000 individuals for 108 generations ~= 12 hours of wall clock time

5. Results

All best of run individuals show convergence and improvement, though one run generated the particularly promising antenna shown in Figure 5. This antenna, the Arrow, has fitness -12.7, a VSWR of 1.65, and maximum gain of 12.9. It's low VSWR makes it a match for any 802.11 design. Compared to an antenna with

a gain of 1, it could transmit a signal equally far with ¼ the power. Its full gain pattern, Figure 8, shows its wide beam width with narrow null points.

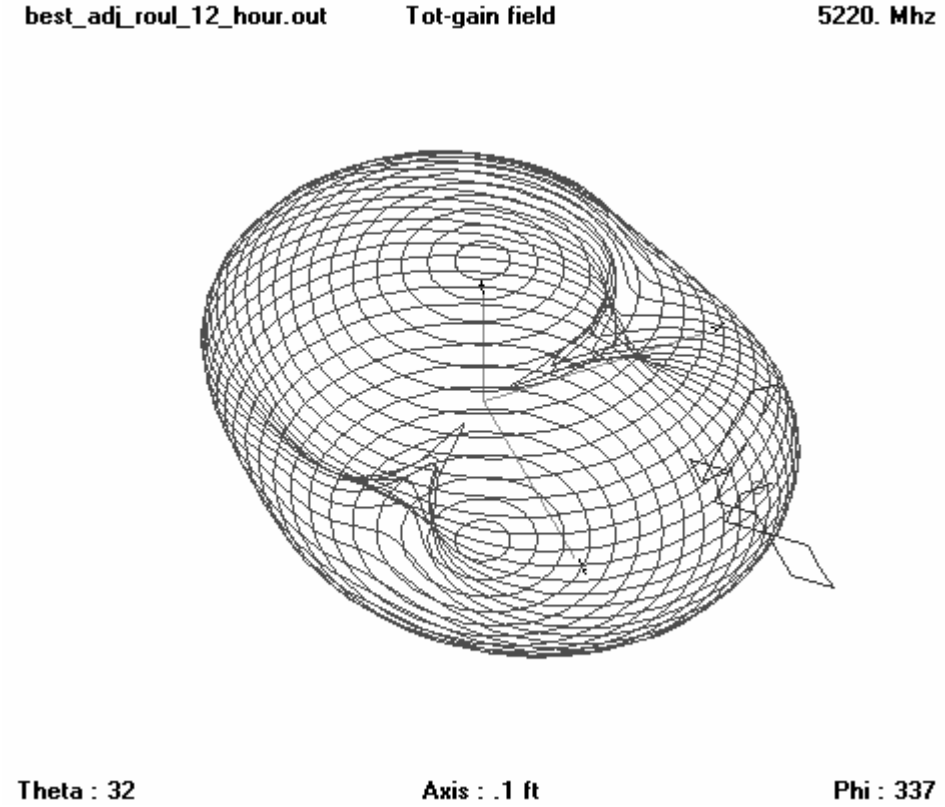

Figure 8: Full gain pattern for best of run 1 (offset of pattern from antenna is an artifact of the diagram)

This antenna also shows good out of sample correlation. When simulated at higher frequencies in this band and into the UNI MID band, its VSWR actually decreases to a minimum of 1.2 at 5250 MHz.

The run that generated the arrow antenna illustrates good general progress. While the mean standardized fitness of .048 at generation 14 shows that most antennae in the generation are totally unusable, Figure 9 shows the mean standard fitness increases six times from even more unusable antennae with fitness of .008.

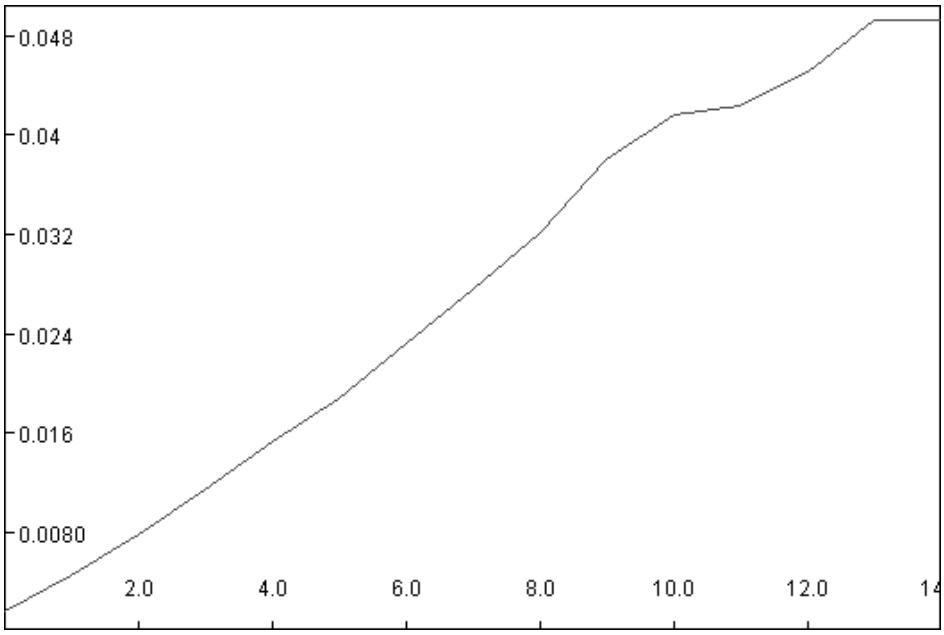

Figure 9: Mean standardized fitness of each generation increased six fold

Tree size is an important measure- the size of the tree directly correlates to the size of the antenna and the size of the antenna directly affects its fitness. Figure 10 shows the best of run individuals find a steady tree size, while the worst never converge.

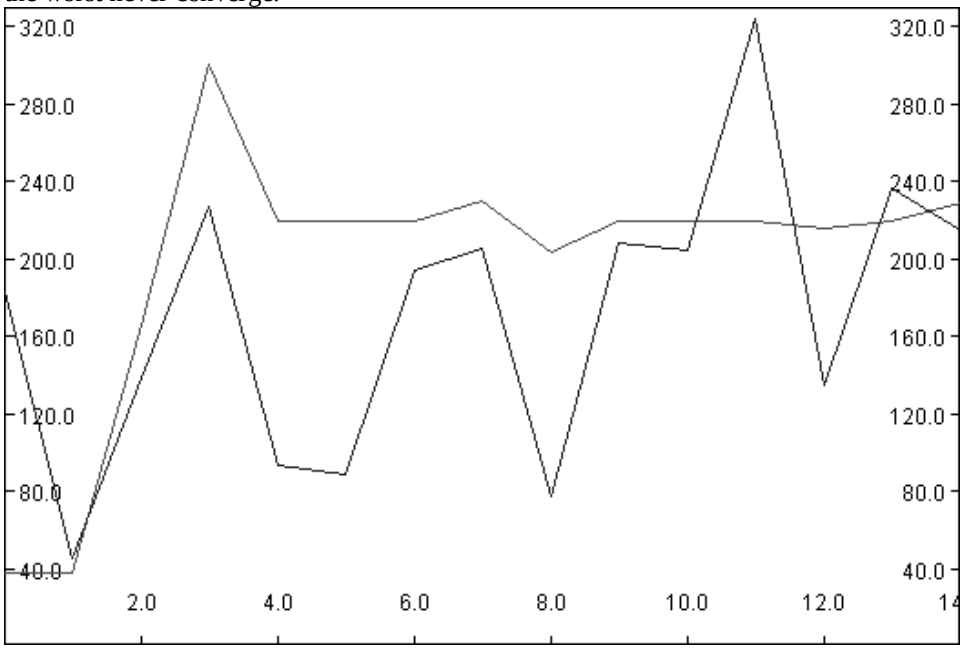

Figure 10: The best of generation individuals find an optimum tree size, the worst of generation never converge

The other runs aren't as exciting, but are valuable for showing areas for improvement. I started the first four runs simultaneously on separate computers. All of them converged more quickly and to better fitness values than I had expected, so I sent off a fifth run with a more stringent exit criteria of -20. Details are summarized in Table 2.

Seed	Selection	M	G	Fitness	Exit at Gen	VSWR	Max Gain	Comment
52376	roulette	1000	108	-12.7	14	1.65	12.9	med manufacturability
52376	tournament	1000	108	-12.59	23	1.7	12.7	difficult manufacturability
3459821	roulette	3000	45	-12.67	19	-200	-99	simulator error
3459821	tournament	3000	45	-12.59	18	1.4	12	difficult manufacturability
20978	roulette	1000	108	-21.8*	97	1.6	22.2	geometry error

Table 2: Best of run individuals and their parameters.

Verifying best of run individuals showed that some of them had either non fatal simulation or geometry errors that make them unacceptable. More data is needed to compare the effects of roulette wheel versus tournament selection.

7. Conclusion

This project shows promising results, but still needs more work. The Arrow antenna with its high gain and exceptional VSWR shows the power of genetic programming and the validity of discretizing the problem space. Investigating the two error conditions that made themselves top individuals is first priority. The discretization does not prevent all errors and perhaps a geometry check would be helpful. The fitness measure needs more work as well: after looking at the results I found myself interested more in a good VSWR than high gain. The difficulty of physically creating these antennae for live testing now seems a pressing problem. Adding a fitness term for ease of construction and sending off a new batch of runs might be easier than trying to create any of these antennae.

8. Future Work

There is much more work that could expand upon this project:

- catch geomentry and simulator input errors earlier
- More runs with an even higher maximum factor in the standardized fitness
- Add fitness term for manufacturability
- Add fitness term for beam width
- Run fitness measure over range of frequencies
- Create antennas in three dimensions
- Smaller form factor
- Full 802.11a spectrum 5180 MHz – 5805 MHz
- Dual band 802.11g /a: 2412 MHz – 2484 MHz, 5180 MHz – 5805 MHz
- Use island model
- Make use of computer cluster
- Add terminals for pen up / pen down
- Use only one rotate terminal
- Add terminals for dropping capacitors and inductors

Acknowledgements

Thanks to Dr. John Koza for his personal interest and assistance with this project, and Brad Marsh from Texas Instruments for making time in a demanding release schedule for this project. Thanks also to L.B. Cebik for his invaluable NEC tutorials and to Pieter-Tjerk de Boer for his example of parsing NEC output files in x-necview.

Bibliography

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