Hindawi Publishing Corporation
International Journal of Antennas and Propagation
Volume 2008, Article ID 197849, 6 pages
doi:10.1155/2008/197849

Research Article

A Comparison of Genetic Programming with Genetic Algorithms for Wire Antenna Design

P. J. Williams and T. C. A. Molteno

Department of Physics, University of Otago, P.O. Box 56, 9016 Dunedin, New Zealand

Correspondence should be addressed to T. C. A. Molteno, tim@physics.otago.ac.nz

Received 21 October 2007; Accepted 18 March 2008

Recommended by Lance Griffiths

This work compares the performance of genetic programming (GP) against traditional fixed-length genome GA approaches on the optimization of wire antenna designs. We describe the implementation of a GP electromagnetic optimization system for wire structures. The results are compared with the traditional GA approach. Although the dimensionality of the search space is much higher for GP than GA, we find that the GP approach gives better results than GA for the same computational effort. In addition, we find that a more expressive antenna structure grammar, dramatically, improves the performance of the GP approach.

1. INTRODUCTION

Genetic programming (GP) [1] applies the genetic algorithm to a population of computer programs. The aim of the optimization is to evolve a program that, when evaluated, produces behavior as close as possible to some desired goal. This approach has been shown to have advantages over traditional fixed-length chromosome genetic algorithm (GA) approaches [1]. Both GP and GA are examples of evolutionary algorithms (EA-s). GP has been applied to a wide variety of optimization problems including the automated design of analog electric circuits [2], however the use of GP for the design of radiating structures has received comparatively less attention.

A GP approach (termed an "open-ended, constructive EA") is described by Lohn et al. [3, 4]. In this work, wire antenna is represented by nodes in a tree structure. Antennas are created by executing operators at each node in the representation. Their work describes the optimization of two wire antennas, one of them allows branching structures and the other does not. Their work, however, uses different population sizes for the two problems and does not provide a direct comparison of the effectiveness of the two approaches for a given computational effort.

In contrast to GP, the traditional fixed-length genetic algorithm (GA) has been widely applied to electromagnetic optimization of structures [5–10]. These approaches typically use a fixed-length chromosome, where an antenna geometry is chosen, and the GA optimizes parameters of this geometry. For example, an optimization of a Yagi-Uda antenna [11] would typically fix the number of elements, and the optimization search space would have a degree of freedom for the position and length of each antenna element. The search space for a nine-element antenna would then have eighteen degrees of freedom.

In this paper, we consider optimization of wire antenna designs and directly compare the performance of GP-based optimization strategies against the traditional fixed-length genome GA approach. We begin with a description of our GP algorithm, and then we apply this to a standard GA structure optimization problem, the crooked wire antenna of Linden and Altshuler [7]. We conclude with a comparison of our GP results with the traditional GA approach.

2. STRUCTURE GENERATOR FUNCTIONS FOR WIRE ANTENNAE

In the GP approach, we represent the antenna structure as a function that *generates* the appropriate structure geometry when evaluated. The GP optimization then evolves a population of *structure generator functions*.

Structure generator functions are expressed as programs in a language. This language consists of four functions that each perform a different task in structure generation. These are "start" S, "wire" W, "join" J, and "end" E. Each of these functions adds extra structure to the individual using relative coordinates. Relative coordinates allow GA operations to specify substructures that can easily be swapped between individuals. The grammar for the structure geometry language we use is described in extended Backus-Naur form [12] below:

```
individual =
          start;
structure =
      {wire|join|end};
start =
         'S' (vector : x, structure : child);
wire =
         'W' (vector : x, structure : child);      (1)
join =
         'J' (structure : a, structure : b);
end =
         'E';
vector =
         '[' number : x, number : y, number : z ']'
          ;
```

Each individual member of the GP population begins with a start function. This has two parameters, the first is a position vector that represents the starting position of the antenna. The second parameter is a child structure that extends from this starting position. Each structure consists of either a wire, join, or end function.

An example of an individual starting at the origin $[0, 0, 0]$ and consisting of two wires finishing at $[1, 1, 1]$ and $[1, 3, 2]$ would be $S([0, 0, 0], J(W([1, 1, 1], E), W([1, 3, 2], E)))$. The meaning of each term in the structure grammar is described below.

Start, S

The start function $S(x, c)$ takes exactly two parameters. The first parameter is the position x of the start of the structure and the second, c, is a child structure element (either a wire, join, or end) attached to the start.

Wire, W

$W(x, c)$ represents a straight wire extending from the current position by a displacement vector x and connecting to the child structure c.

Join, J

$J(a, b)$ joins its two child structure parameters a and b at the current position.

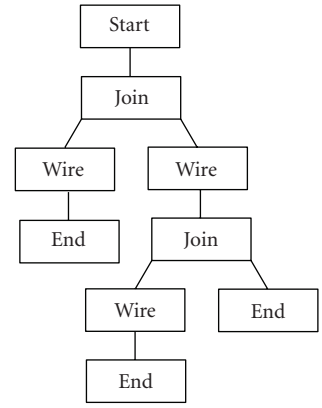

Figure 1: The tree structure representing the function $S(\mathbf{x}_0, J(W(\mathbf{x}_1, E), W(\mathbf{x}_2, J(W(\mathbf{x}_3, E), E)))$. The $\mathbf{x}_i$ are coordinate vectors for the relative positions of the structure elements.

End, E

This is a terminal symbol for structures. E is a function that takes no parameters and indicates that no further structure is present.

In this *functional form*, individuals are naturally represented as tree structures. Figure 1 shows the tree representation of a three-wire individual structure.

2.1. Genetic operations

The tree structures that are compositions of structure functions described in Section 2 are then optimized using the usual genetic programming approach (see, e.g., Koza [1]). Subsequent generations are evolved from the current generation by selection, crossover, and mutation.

2.1.1. Initial population

An initial population of individuals is randomly created. Each individual begins with a "start" function, with a random position parameter and a random child structure. In this work, the random child structure is created by selecting with equal probability a "wire," "join," or "end" function, each with randomly chosen parameters as appropriate (positions and further child structures). Different choices of probability might yield improved results but were not explored.

2.1.2. Selection

The selection of individuals for reproduction is carried out by first evaluating the fitness, F, of each individual in the population. This fitness is then used to weight the probability that an individual will be selected for direct copying (elitism), crossover, and mutation.

2.1.3. Elitism

A fraction of the next generation is created by direct copying. This process is also called elitism. In the work presented

here, the fittest 1% of the current generation are selected for direct copying to the next generation. Another 18% of the next generation are copied from the previous generation by proportionate selection [6], where the probability of an individual being selected is proportional to its fitness. The remaining individuals are selected by crossover and mutation as described below.

2.1.4. Crossover

Individuals are selected for crossover by proportionate selection. Two individuals from the current generation, when selected for crossover, will generate two new individuals in the next generation. A node on each tree is selected at random and the child nodes are swapped. Thus, two individuals selected for crossover will produce two individuals in the next generation. This method of crossover between two individuals can grow or reduce the size of the individuals generated in subsequent generations.

We tested crossover rates between 20% and 90%, and found that using a crossover rate of 80% yielded the highest fitness after 50 generations. In the work presented here, 80% of the next generation are generated by crossover.

2.1.5. Mutation

A small percentage of individuals are selected for mutation. Mutation introduces new structures into the next generation. A mutation was carried out by choosing a random node in the tree structure of an individual, deleting that node and its children and substituting a randomly generated structure in its place. In the work presented here, 1% of the next generation are generated by mutation.

2.1.6. Fitness

After each generation has been selected each individual's fitness is evaluated using a cost function, and this fitness drives the selection process for the next generation. If, for example, an individual is generated with no wire elements, that is, $S(E)$, then it will not radiate and the resulting high cost (low fitness) will make it extremely unlikely to survive into the next generation.

2.2. Implementation

The GP code was written in C++ and designed from the ground up to operate efficiently in a cluster computing environment. A master node executes the genetic programming code, while the evaluation of individuals is performed on all the other nodes in the cluster using the message passing interface (MPI) [13]. This architecture allows the optimization system to operate effectively in a heterogeneous parallel computing environment.

To improve speed, particularly on small geometries, we use Nec2++ [14] to evaluate the radiation patterns of individual structures. Nec2++ is an open-source (released under the GPL open-source license), high-performance, NEC-2 compatible electromagnetic code written in C++ that can be incorporated into an optimization system directly or by linking to a library. For small structures, we find that Nec2++ will evaluate an individual approximately three orders of magnitude faster than calling NEC-2 as a separate executable. For large structures with more than 300 segments, we find that Nec2++ is approximately ten times faster than the original NEC-2 code compiled with gfortran 4.1.

3. THE CROOKED-WIRE PROBLEM

To compare the performance of the GP approach against the chromosome-based GA approach, we consider the "crooked-wire problem" of Linden and Altshuler [7]. Linden's goal was to design an antenna that had a uniform gain pattern across the hemisphere for a right hand circularly polarized (RHCP) signal at 1600 MHz. The antenna would consist of straight pieces of wire connected in series.

The antenna geometry was only loosely constrained by the antenna size, excitation source, number of wires, and presence of a ground plane. The antenna was confined to a cube, half a wavelength on each side. Binary strings were used to encode the antenna design, five bits were allowed for each component of the coordinates of the endpoints of each wire. There were therefore 32 possible positions in each dimension for each endpoint. The number of wires was also a constrained—to seven. As there are five bits per axis-coordinate, three coordinates per endpoint, and seven unique endpoints to be designated (each wire endpoint forms the start of the next wire), the chromosome consists of $5 \times 3 \times 7$ genes (or bits).

3.1. Fitness

In the original crooked-wire GA optimization (see Section 3), the fitness of each individual was a measure of how isotropic the RHCP radiation is from the antenna. For each individual, the radiation pattern $G(\theta, \phi)$ was computed at increments of 5 degrees in elevation (θ) and azimuth (ϕ). The range of elevation was $-80 \leq \theta \leq 80$, and the range of azimuth was $0 \leq \phi \leq 175$. The fitness cost function, C, of an individual is

$$C = \sum_{\theta \in [-80,80]} \sum_{\phi \in [0,175]} \left(G(\theta, \phi) - \overline{G}\right)^2, \tag{2}$$

where $G(\theta, \phi)$ is the gain (in dB) at angles θ and ϕ, and $\overline{G}$ is the average gain over all the angles. Low-cost functions correspond to highly isotropic radiation and high fitness. If the average gain was less than -10 dB, then it was set to -10 dB. This heavily penalized designs that did not radiate at all.

3.2. Results

Using a genetic algorithm, Linden produced a 7-wire antenna with unusual shape that had a radiation pattern that varied by less than 4 dB over the desired angles. This antenna, with a cost $C = 234$, is shown in Figure 2. He also found many

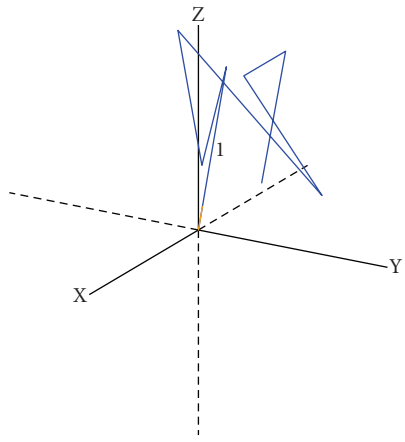

Figure 2: Linden's GA optimized crooked-wire antenna with $C = 234$.

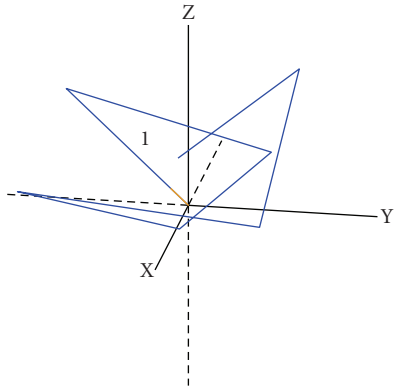

Figure 3: A GP-optimized seven-wire antenna with $C = 177$.

antenna designs with similar fitness but of very different geometry.

4. GP RESULTS

The performance of the genetic programming approach was evaluated by applying it to the crooked-wire problem. Linden's original simulations used a population of 500 and evolved for 50 generations, or 25000 fitness evaluations. For our genetic programming system using the same population and running for the same number of generations, complete optimization time was around 5 hours on a single computer. We performed each optimization 30 times because, due to the randomness inherent in the genetic algorithm, each run produces a different design from the last but with similar performance.

4.1. Modified fitness

Our GP algorithm attempts to maximize fitness, so a modified fitness function, F_m, was chosen that is proportional to the reciprocal of Linden's original fitness cost C. In addition, in order to limit our search to geometries with only seven

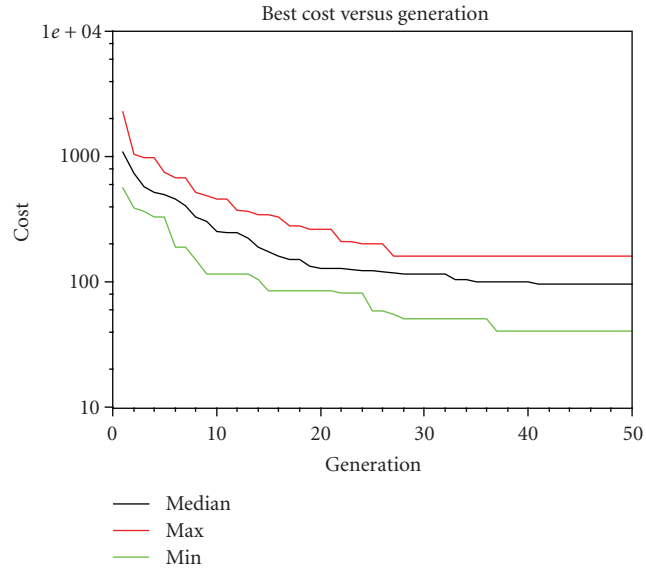

Figure 4: Best individual cost versus generation for single-branch GP optimization.

wires, we severely penalized the fitness of individuals that did not have seven wires. The modified fitness then became

$$F_m = \frac{1}{F + P(n)}, \tag{3}$$

where $P(n)$ is the penalty function, and n is the number of wires in the geometry,

$$P(n) = \begin{cases} 0 & n = 7, \\ 100 & n \neq 7, \end{cases} \tag{4}$$

However for the purposes of comparison with Linden's original results, we still quote the fitness cost C which is minimized during the optimization process.

4.2. Crooked-wire revisited

Using the modified fitness F_m, 30 optimization runs were performed each with a population of 500 for 50 generations. During these runs, the cost of the best individual after each generation was recorded. The results are shown in Figure 4. After 50 generations, the cost, C, for the best-individual ranged from 40.6 to 161.4 with a median value of 96.

Figure 3 shows a sample best-cost individual. It has a cost C of 177—a better result than Linden's best individual which had $C = 234$ [8]. Because of the randomness inherent in the GA direct comparisons are difficult, however it appears that the GP approach yields better results for the same computational effort (number of individual fitness evaluations) when compared with the GA approach used by Linden.

4.3. Multibranch seven-wire geometries

Our structure generation language (see Section 2) allows individuals with more complex geometry than Linden's

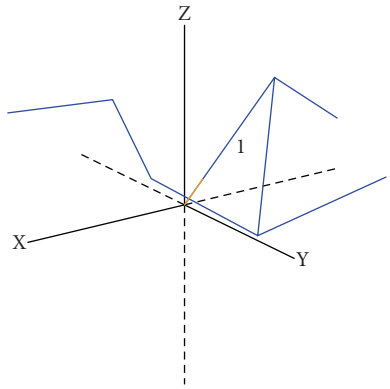

Figure 5: A GP-optimized multibranch antenna with $C = 57$.

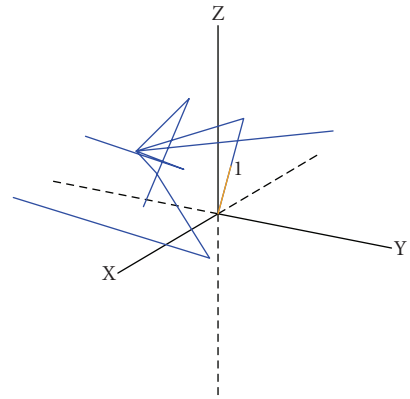

Figure 6: A GP-optimized complex-geometry antenna with $C = 20$.

original work. We extend the language to include the "join" operation, in other words we allow a single wire to branch into two wires.

The optimization was then run with a population of 500 for 50 generations, while still restricting the design to a total of seven wires. For thirty optimization runs, after 50 generations the best individual from each run ranged from $C = 25$ to $C = 185$ with a median cost of $C = 85$. The more flexible geometry resulted in a wider spread of designs (an example is shown in Figure 5). In the best case, the design had a cost two times lower than the best case individual from Section 4.2. In the worst-case, the cost was slightly higher.

For a given computational effort, the grammar of the structure generation language can clearly influence the performance of the GP optimization.

4.4. Complex geometries

We finally allowed an arbitrary number of wires in the simulation. This was achieved by removing the penalty function $P(n)$ from the fitness function. Individuals are no longer penalized, if they have a larger number of wires. Using this modified fitness, we ran the optimization with a population of 5000 for 50 generations. This resulted in a best individual with ten elements and a cost $C = 20$, shown in Figure 6. The coordinates of this individual are shown in Table 1.

Table 1: Coordinates (in mm) for the $C = 20$ GP-optimized antenna. All wires have a radius of 1 mm.

Wire	Start (mm)			End (mm)		
(tag)	x	y	z	x	y	z
1	0	0	0	−2.1	3.8	17.2
2	−2.1	3.8	17.2	8.6	−11	12
3	8.6	−11	12	−9.3	17.1	15.1
4	8.6	−11	12	2.4	−5.3	7.8
5	2.4	−5.3	7.8	27	−10.3	20.6
6	8.6	−11	12	−3.4	−7.6	18.3
7	−3.4	−7.6	18.3	26.8	0.9	9.9
8	8.6	−11	12	3.6	−1	6.7
9	3.6	−1	6.7	29	15.1	3.9
10	29	15.1	3.9	30.8	−22.2	8.6

5. CONCLUSIONS

We have carried out a direct comparison of GP against a fixed-length chromosome GA approach. The results of this comparison, described in Section 4.2, show that genetic programming produces equivalent or better results than GA for a similar computational effort. We also showed (in Section 4.3) how the addition of a "join" operation to the structure generation language dramatically improves the performance of GP optimization without increasing computational effort. This richer language allows the individuals to have multiple branch geometries. Further exploration of structure generation grammars is warranted, as they clearly have a dramatic effect on the GP optimization results, without increasing the computational effort required.

In addition, the flexibility of the GP approach allows for a richer variety of antenna geometries and we find, for the crooked-wire problem, that this flexibility enables the evolution of much better designs. When we relaxed the seven-wire requirement of Linden's original crooked wire geometry, our best design, described in Section 4.4, has a cost ($C = 20$) more than ten times lower than that obtained by a fixed-length chromosome GA approach ($C = 234$). It should be noted that this complex geometry GP optimization took significantly more computational effort than the original GA optimization, however with the same computational effort (using a population of 500 for 50 generations) and allowing multiple branch geometries, we achieved a cost $C = 57$—still significantly better than the fixed-length genome GA approach.

A natural extension of this work is to use automatically defined functions (ADF-s). This enriches the grammar of the GP language to include the definition and calling of functions. For many problems, the use of ADF's reduces the computational effort required to achieve a solution of a given fitness [15].

REFERENCES

[1] J. R. Koza, *Genetic Programming*, MIT Press, Cambridge, Mass, USA, 1992.

[2] J. R. Koza, F. H. Bennett III, D. Andre, M. A. Keane, and F. Dunlap, "Automated synthesis of analog electrical circuits by means of genetic programming," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 2, pp. 109–128, 1997.

[3] J. D. Lohn, G. S. Hornby, and D. S. Linden, "Evolutionary antenna design for a NASA spacecraft," in *Genetic Programming Theory and Practice II*, U.-M. O'Reilly, R. L. Riolo, T. Yu, and B. Worzel, Eds., chapter 18, pp. 301–315, Springer, Ann Arbor, Mich, USA, 2004.

[4] J. D. Lohn, G. S. Hornby, and D. S. Linden, "Rapid re-evolution of an X-band antenna for NASA's space technology 5 mission," in *Genetic Programming Theory and Practice III*, U.-M. O'Reilly, R. L. Riolo, T. Yu, and B. Worzel, Eds., chapter 5, pp. 65–78, Springer, Ann Arbor, Mich, USA, 2005.

[5] R. L. Haupt, "Introduction to genetic algorithms for electromagnetics," *IEEE Antennas and Propagation Magazine*, vol. 37, no. 2, pp. 7–15, 1995.

[6] J. M. Johnson and Y. Rahmat-Samii, "Genetic algorithms in engineering electromagnetics," *IEEE Antennas and Propagation Magazine*, vol. 39, no. 4, pp. 7–21, 1997.

[7] D. S. Linden and E. E. Altshuler, "Automating wire antenna design using genetic algorithms," *Microwave Journal*, vol. 39, no. 3, pp. 74–86, 1996.

[8] E. E. Altshuler and D. S. Linden, "Wire-antenna designs using genetic algorithms," *IEEE Antennas and Propagation Magazine*, vol. 39, no. 2, pp. 33–43, 1997.

[9] E. E. Altshuler and D. S. Linden, "Design of a loaded monopole having hemispherical coverage using a genetic algorithm," *IEEE Transactions on Antennas and Propagation*, vol. 45, no. 1, pp. 1–4, 1997.

[10] S. Cui, A. Mohan, and D. S. Weile, "Pareto optimal design of absorbers using a parallel elitist nondominated sorting genetic algorithm and the finite element-boundary integral method," *IEEE Transactions on Antennas and Propagation*, vol. 53, no. 6, pp. 2099–2107, 2005.

[11] E. A. Jones and W. T. Joines, "Design of Yagi-Uda antennas using genetic algorithms," *IEEE Transactions on Antennas and Propagation*, vol. 45, no. 9, pp. 1386–1392, 1997.

[12] N. Wirth, "What can we do about the unnecessary diversity of notation for syntactic definitions?" *Communications of the ACM*, vol. 20, no. 11, pp. 822–823, 1977.

[13] W. Gropp, E. Lusk, and A. Skjellum, *Using MPI: Portable Parallel Programming with the Message-Passing Interface*, MIT Press, Cambridge, Mass, USA, 1994.

[14] T. C. A. Molteno and P. J. Williams, "NEC2++: high-performance numerical electromagnetics code," in *Proceedings of the 13th Electronics New Zealand Conference (ENZCON '06)*, Christchurch, New Zealand, November 2006.

[15] J. R. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs*, MIT Press, Cambridge, Mass, USA, 1994.