OPTIMISING ALGORITHMS FOR ANTENNA DESIGN

B. Orchard* and A.R. Clark*

*School of Electrical and Information Engineering, University of the Witwatersrand, Johannesburg, 2050, South Africa. B. Orchard has since joined Improvico Engineering Solutions.

Abstract: In this paper two global optimisation techniques were investigated, namely, the Genetic Algorithm and Simulated Annealing. These techniques were programmed in Matlab® and implemented in SuperNEC which is an Electromagnetics Method-of-Moments software simulation program. The optimisation techniques were then applied to the 5, 12 and 17 element Yagi antennas by firstly varying only element lengths and then by varying element lengths and the spacing between them. The optimisation techniques were also applied to the 5, 15 and 35 turn axial mode helix antennas by varying the spacing between turns and the radius of the turns. The results showed improvements to the classical designs in both cases for both optimisation techniques. Comparisons between the two optimisation techniques showed that the Genetic Algorithm worked more effectively in terms of results obtained and efficiently in terms of run time.

Key words: Genetic Algorithms, Simulated Annealing, Optimising, Antenna Design

1. INTRODUCTION

Due to the vast number of variables and large number of output characteristics associated with many antenna structures, antennas can not be optimised by traditional techniques which use gradient methods. This paper will therefore examine the theory and implementation of two global optimisation techniques namely: the Genetic Algorithm and Simulated Annealing. These techniques are able to handle continuous or discrete variables that may often contain constraints in allowable values and find a solution which represents a ‘global’ optimum of the variables involved.

In this paper a short overview of the optimisation techniques will be presented, the various forms of the algorithms investigated, and comparisons made between them and classical designs. The implementation of the algorithms in SuperNEC, which is a Mom/UTD (Method-of-Moments/Universal Theory of Diffraction) antenna simulation program, will also be discussed. Finally comparisons will be drawn between the two techniques and the final conclusions will be made. The antennas that will be investigated include the 5, 12 and 17 element Yagi’s and the 5, 15 and 35 turn axial mode helices. [16]

2. THE GENETIC ALGORITHM

2.1 Overview of the Genetic Algorithm

Genetic Algorithms (GA) are stochastic-based search methods modelled on the principles and concepts of natural selection and evolution [1–4]. They can handle the common characteristics of electromagnetic optimisation problems that are not readily handled by other traditional optimisation techniques as they operate on a group, or population, of trial solutions in parallel, and they operate on a coding of the function parameters rather than the parameters directly. In the GA each variable is represented as a binary code called a gene. These genes are then arranged and combined to form a chromosome. Each chromosome has an associated fitness value, assigning a value of merit to the chromosome. A high fitness value being the characteristic of a good chromosome.

The algorithm begins by generating a number of random chromosomes. A selection strategy is then implemented which decides which chromosomes will take part in the evolution process. The unacceptable chromosomes are simply discarded. The chromosomes that survive become parents, by swapping some of their genetic material to produce two new offspring. The mating process is repeated between the remaining chromosomes until enough new chromosomes have been produced, leaving a constant number of chromosomes after each iteration. This algorithm is repeated until a satisfactory fitness is reached or a predefined number of iterations/generations have been run. The algorithm also makes use of mutations, which cause small random changes in chromosomes to explore other solutions which may not have been in the optimiser’s search space.

2.2 Coding of parameters

As mentioned above a binary string known as a chromosome is used to represent the system variables.
These chromosomes are encoded/decoded using a pre-defined method. In order to reduce redundancy and improve the performance of the algorithm, the sizes of genes representing different variables may differ in size. The binary size of a gene for variable $v$ can be calculated in the following manner:

$$n_{\text{bits}} = \log\left(\frac{\text{range}}{\text{resolution}}\right) \log 2 \quad (1)$$

where $\text{range} = v_{\text{max}} - v_{\text{min}}$ and resolution is the smallest value of change in variable $v$ that can be distinguished. In the case that $n_{\text{bits}}$ is calculated not to be an integer value, it is always rounded up and a scaling factor used when encoding/decoding the chromosome.

A test case antenna will be used throughout this paper (except where indicated), allowing comparisons to be made between various forms of the optimisation techniques. The test case antenna consists of a 5 element Yagi operating at 300MHz which uses a folded dipole as the feed element. The spacing between the folded dipole is 0.05m, the radii of all the elements is 0.01m, and a constant spacing of 0.2m is used between elements. When used with an optimisation technique the required Gain and VSWR is 10dBi and 75 ohms respectively and the optimisation variables are the element lengths which are allowed to vary between 0.350–0.650m, giving a problem with 300$^5$ possible solutions.

2.3 Various selection strategies

The various selection strategies are presented in [3],[5]:

Population Decimation: In this strategy the chromosomes are ranked according to their fitness values (fitness is defined in section 2.8) from highest to lowest. Only the top predefined number of chromosomes or chromosomes with a fitness higher than a predefined minimum limit are selected to take part in the evolution process. Once a chromosome with a unique characteristic has been removed, the only way that it can be reintroduced is by mutation [3].

Proportionate Selection: For this selection strategy the probability that a chromosome is selected, $p_{\text{selection}}$, is proportionate to the fitness of the chromosome:

$$p_{\text{selection}} = \frac{\text{fitness of chromosome}_i}{\sum_i \text{fitness of chromosome}_i} \quad (2)$$

The higher the fitness the greater the probability a chromosome will be selected and vice versa. One of the implications of using this strategy is that it is possible that a highly unfit chromosome can be selected to take part in the mating procedure.

Tournament Selection: In tournament selection a number of individuals $N$, are randomly selected from the population. Those individuals then ‘compete’ on a basis of their fitness, with the individual with the highest fitness ‘winning’. This process is repeated a number of times until the new population is filled. This strategy also allows an individual to be selected more than once as is the case with the proportionate selection method mentioned above. In this research $N = 2$, which is the most commonly used form of tournament selection, and is known as Binary tournament selection [3]. Figure 1 shows a comparison between the various selection strategies for the test case antenna. We see that population decimation appears to work the best.

![Figure 1. Various selection techniques for 30 optimisations of the test case antenna using 70 chromosomes, a 0.15 mutation probability, an AFP mating scheme and random crossover.](image)

2.4 Mating Schemes

While the selection strategies are involved with selecting which individuals will take part in the evolution process (be parents), the mating strategies select which two parent chromosomes will mate with one another. Before the mating schemes are implemented all the selected individuals are ranked from best to worst in terms of fitness values. The following schemes are described in [5]:

Best-Mates-Worst(BMW): As the name suggests, the chromosomes with the highest fitness mates with the chromosome with the lowest fitness, and the chromosome with the second highest fitness mates with the chromosome with the second lowest fitness value etc.
Adjacent-Fitness-Pairing (AFP): In this scheme the two individuals ranked the highest mate together and then the next two highest ranked individuals mate together, etc.

Emperor-Selective (EMS): In EMS the highest ranked individual mates with the second highest, fourth highest and sixth highest ranked individuals etc. This leaves the third, fifth etc individuals unchanged for the next generation. From figure 2 we see that AFP and BMW are the most effective mating schemes.

2.5 Crossover

Crossover rearranges genes between chromosomes in an effort to try and find the best combination of genes. In this paper only Normal Single Point Crossover will be dealt with. In this method the crossover point $p_{\text{cross}}$ determines at what points the two parent chromosomes will be cut. When a crossover occurs, the first new child chromosome is generated by taking the first slice of parent chromosome 1 and the second slice of parent chromosome 2. The inverse is true for generating the second chromosome. Figure 3 shows that for $p_{\text{cross}} = \text{random value}$, is the most effective.

2.6 Mutation

A mutation occurs in a chromosome with the probability $p_{\text{mutation}}$. When a mutation is to occur (if $p$ falls within $p_{\text{mutation}}$), a random bit in the chromosome is simply inverted, that is a ‘1’ is changed into a ‘0’ and vice versa. Figure 4 shows the use of mutation probabilities for the test case antenna. From the figure we see that the optimum mutation probability lies between 5–25% which is in general agreement with [6] which found the optimum mutation rate to be 5–20%.

2.7 Other Forms/Extensions of the Genetic Algorithm

In this section we will look at two other forms/extensions of the GA.
Elitist Strategy: This is simply the saving and inserting of the best individual from the previous generation into the new generation. This ensures that the best individual is not lost due to crossover or mutation.

Steady-State Genetic Algorithms: In the Steady-State Genetic Algorithm a portion of the best individuals are inserted as part of the next generation. Steady-state GA’s have been shown to converge faster in many applications [3].

Figure 5. Various steady state GA’s for 30 optimisations of the test case antenna using using 80 chromosomes, 0.15 mutation probability, population decimation, an AFP mating scheme and random crossover.

Figure 5 shows the optimum number of individuals to save and insert in the new generation to be 10.

2.8 Fitness function for the Genetic Algorithm

The following function is used to calculate the fitness of individuals:

\[
\text{Fitness} = \left( \frac{\text{Actual Gain}}{\text{Req. Gain}} \right)^{g\text{Factor}} + \left( \frac{\text{Req. VSWR}}{\text{Actual VSWR}} \right)^{v\text{Factor}}
\]

(3)

where gFactor and vFactor are user-defined values, and will be set=2 (the norm value).

If requirements dictate that the antenna structure be simulated for more than one frequency, there are two options for calculating the fitness: by calculating the fitness in equation 3 above for each frequency and averaging them. A second approach is to simply use the worst ratio of actual gain to required gain and worst ratio of required VSWR to actual VSWR over the frequency range.

2.9 Number of chromosomes and number of generations

Various numbers of chromosomes and number of generations were investigated in the optimisation of the test case antenna. Figure 6 summarises the results obtained. We observe that the number of chromosomes has the effect of determining the best overall solution that can be obtained—in general the more chromosomes the better the solution. It is also clear that the optimum number of chromosomes to use for this problem is between 50–80 chromosomes. We also notice that the number of generations only affects the best solutions obtained for each number of chromosomes used until convergence is reached. In this problem we see that convergence is reached after approximately 60 generations, no matter how many chromosomes used.

Figure 6. The effect of using different numbers of chromosomes for 30 optimisations of the test case antenna using a 0.15 mutation probability, population decimation, AFP mating scheme and random crossover.

2.10 Genetic Algorithm results

In this section we will look at results obtained by optimising the 5, 12 and 17 element Yagi’s, and the 5, 15 and 35 turn helices, using the best techniques described above. In addition the corresponding classical/textbook designs will also be presented.

The Yagi optimisations make use of 80 chromosomes and 80 generations, while the smaller helix problems only use 30 generations and 15 chromosomes and saving and inserting only the best 5 from each iteration. Each optimisation problem was run 10 times of which one was selected for the the purpose of making comparisons.

The Yagi’s all have elements with a 0.008m radius.
and a folded dipole as the feed element. Another important point to note is that all the Yagi designs use constant length directors and the Yagi’s have a spacing which is constant between all the elements except in some cases between the feed and reflector.

Table 1. Genetic Algorithm Yagi design results

<table>
<thead>
<tr>
<th></th>
<th>Gain @ 300MHz (dB)</th>
<th>VSWR @ 300MHz</th>
<th>%gain BW</th>
<th>%vswr BW</th>
</tr>
</thead>
<tbody>
<tr>
<td>The 5 element Yagi textbook*</td>
<td>10.48</td>
<td>1.0247</td>
<td>14.0</td>
<td>1.3</td>
</tr>
<tr>
<td>same spacing</td>
<td>11.04</td>
<td>1.0046</td>
<td>17.0</td>
<td>3.0</td>
</tr>
<tr>
<td>GA spacing</td>
<td>10.51</td>
<td>1.0016</td>
<td>18.7</td>
<td>5.3</td>
</tr>
<tr>
<td>The 12 element Yagi textbook*</td>
<td>12.36</td>
<td>1.2643</td>
<td>14.3</td>
<td>2.3</td>
</tr>
<tr>
<td>same spacing</td>
<td>12.11</td>
<td>1.0059</td>
<td>29.3</td>
<td>20.0</td>
</tr>
<tr>
<td>GA spacing</td>
<td>12.59</td>
<td>1.0020</td>
<td>25.0</td>
<td>13.0</td>
</tr>
<tr>
<td>The 17 element Yagi textbook*</td>
<td>12.72</td>
<td>1.0143</td>
<td>14.3</td>
<td>3.3</td>
</tr>
<tr>
<td>same spacing</td>
<td>14.59</td>
<td>1.5319</td>
<td>15.7</td>
<td>11.0</td>
</tr>
<tr>
<td>GA spacing</td>
<td>13.97</td>
<td>1.0079</td>
<td>17.3</td>
<td>12.7</td>
</tr>
</tbody>
</table>

*These Yagis were designed using techniques described by Lawson in [7] and then finding the resonant feed element length

The definition of gain bandwidth used in this paper is:

\[
\text{% gain BW} = \left( \frac{f_h - f_l}{f_c} \right) \times 100 \quad (4)
\]

where \(f_c\) = design frequency and \(f_l\) and \(f_h\) are the lowest and highest frequencies at which the gain is 3dB lower than the gain at \(f_c\).

The definition of VSWR bandwidth is that bandwidth where the VSWR is below 2:1.

Table 1 summarises the Yagi textbook designs and results obtained by just optimising element lengths (same spacing results) and then by optimising for both element lengths and spacing between elements (GA spacing results). The tests show that optimising for element spacing has little effect on the outcome of the results obtained in terms of fitness since the spacing used by the same spacing Yagi’s is relatively near optimum.

The helices were optimised by varying spacing between turns and the radius of turns. Table 2 summarises the measured results and textbook designs.

All the optimised antennas performed better than the textbook designs.

Table 2. Genetic Algorithm helix design results

<table>
<thead>
<tr>
<th></th>
<th>Gain @ 1000MHz (dB)</th>
<th>VSWR @ 1000MHz</th>
<th>%gain BW</th>
<th>%vswr BW</th>
</tr>
</thead>
<tbody>
<tr>
<td>The 5 turn helix textbook*</td>
<td>10.25</td>
<td>1.2454</td>
<td>68.5</td>
<td>70.0</td>
</tr>
<tr>
<td>GA design</td>
<td>10.86</td>
<td>1.1902</td>
<td>65.0</td>
<td>68.0</td>
</tr>
<tr>
<td>The 15 turn helix textbook*</td>
<td>14.04</td>
<td>1.3078</td>
<td>44.5</td>
<td>54.0</td>
</tr>
<tr>
<td>GA design</td>
<td>14.05</td>
<td>1.3032</td>
<td>47.5</td>
<td>55.0</td>
</tr>
<tr>
<td>The 35 turn helix textbook*</td>
<td>15.52</td>
<td>1.3100</td>
<td>36.5</td>
<td>62.0</td>
</tr>
<tr>
<td>GA design</td>
<td>15.81</td>
<td>1.2689</td>
<td>42.1</td>
<td>218.9</td>
</tr>
</tbody>
</table>

*Helices designed using techniques described by King et al in [8]

3. SIMULATED ANNEALING

3.1 Overview of Simulated Annealing

Simulated Annealing is derived from the physical process of annealing metals, in which a metal is heated and then slowly cooled in order to form strong crystalline bonds within the structure [8–10].

In [11] the authors describe the Simulated Annealing optimisation algorithm, which in short randomly selects evaluation points and accepts all points that yield an increase in the objective function, but also accept some points that lead to a decrease in the objective (this allows the algorithm to escape from local minima). Unlike the fitness function in the Genetic Algorithm, a decrease in cost in Simulated Annealing represents a increase in the objective function. As more and more points are tried periodically, the algorithm focuses on the best point so far, and reduces the range over which random points are generated. This technique does not require any derivatives to be calculated.

The algorithm begins with an initialisation process. In this process a number of random configurations, \(N_{\text{init}}\), are generated and the number of steps increasing cost \(N_+\), number of steps decreasing cost \(N_-\) and the corresponding average increase in cost \(\Delta C_+\) is calculated. Using these quantities and one of the methods described below, the initial temperature \(T_0\) is calculated. These methods ensure the initial variance and initial temperature relate to one another, which is necessary for proper convergence [12].

Once the initialisation process has been completed, the best configuration obtained in the process, with cost \(C_j\), is used as the starting point. This point is then varied randomly within a step size (initially set as the range), for all the variables \(v_i\) where \(i\) = number of variables and the new cost \(C_{j+1}\) evaluated. If the cost \(C_{j+1} < C_j\) then the new configuration
is automatically accepted. However if \( C_{j+1} > C_j \), the configuration is accepted with a small probability 
\[
P_{\text{accept}} = e^{-\frac{(C_{j+1} - C_j)}{T}} \quad [13][14][15].\]
We observe that the probability is proportional to temperature and inversely proportional to the change in cost. This process is continued until thermal equilibrium is reached which is determined by the techniques presented below.

Once equilibrium is reached, the temperature is lowered using a cooling algorithm, and the step size in which the parameters may vary is reduced. This process is repeated a number of times, using the new temperature and last accepted configuration obtained as the new starting point. Finally the optimisation process terminates once a specific temperature is reached or a set number of optimisation cycles or number of temperature points are exceeded.

### 3.2 Calculation of Initial Temperature

Various techniques exist for the calculation of the initial temperature. These techniques include:

- Method proposed by Johnson et. al. [16]
\[
T_0 = \frac{\Delta C_+}{\ln(\chi_0)} \quad (5)
\]
A useful value for \( \chi \) was found by Kirkpatrick to be 0.8 [12],[17].
- Method proposed by Aarts et. al. [18]
\[
T_0 = \frac{\Delta C_+}{\ln \left( \frac{N_{\text{acc}}}{\chi_0 N_{\text{acc}} - (1-\chi_0) N_{\text{rej}}} \right)} \quad (6)
\]
- The initial temperature can be calculated using the average cost
\[
T_0 = \bar{C} \quad (7)
\]
- the median cost
\[
T_0 = \tilde{C} \quad (8)
\]
- or using the initial variance
\[
T_0 = \frac{\sum_{i=0}^{N_{\text{init}}} \bar{C} - C_i}{N_{\text{init}}} \quad (9)
\]

### 3.3 Thermal Equilibrium

Thermal equilibrium is calculated using inhomogeneous algorithms, where the temperature is decreased after each fulfilled Markov chain. A Markov chain is a sequence of trials where each trial depends only on the previous trial [12]. The value \( N \) represents the total number of configurations run since the previous

<table>
<thead>
<tr>
<th>Equilibrium Methods</th>
<th>Sechen</th>
<th>Johnson</th>
<th>Johnson variation</th>
<th>Nahar</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ave. Cost (norm.)</td>
<td>1.077</td>
<td>2.059</td>
<td>2.114</td>
<td>1.000</td>
</tr>
<tr>
<td>Median Cost (norm.)</td>
<td>1.147</td>
<td>2.363</td>
<td>2.365</td>
<td>1.000</td>
</tr>
</tbody>
</table>

Refer to table 3 for a comparison of the equilibrium techniques for the test case antenna, showing that Nahar’s method operates the most effectively.

### 3.4 Changing the step size

The step size for variable \( i \) is determined by the following equation:
\[
\text{Stepsize}(v_i) = \text{range of } v_i \times \text{Step Value} \quad (14)
\]
Two methods of reducing the step size (by decreasing the Step Value) for decreasing temperature was investigated. The first employs a technique which relates the current temperature to the initial temperature:
\[
\text{Step Value} = \frac{T_0}{T} \quad (15)
\]
The second method decreases the temperature relative to the number of temperature points:
\[
\text{Step Value} = 1 - \left( \frac{1}{\text{Number of temp points} + 1} \right) \times k \quad (16)
\]
where \( k \) = Current Temperature Point Number. The second method was found to work slightly better (see table 4)
Table 4. Comparison of methods to calculate changes in step size

<table>
<thead>
<tr>
<th>Method</th>
<th>Average Cost (normalised)</th>
<th>Median Cost (normalised)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method 1</td>
<td>2.666</td>
<td>3.383</td>
</tr>
<tr>
<td>Method 2</td>
<td>1.000</td>
<td>1.000</td>
</tr>
</tbody>
</table>

3.5 Various forms of Simulated Annealing

Original Simulated Annealing: This method uses a cooling schedule given by

$$T_k = \alpha^k T_0$$

(17)

where $\alpha$ is a scaling constant between 0 and 1. Useful values for $\alpha$ have been claimed to be $0.8 < \alpha < 0.99$ [12]. The original Simulated Annealing uses a random number generator with a uniform distribution to create the new configurations (within the step size).

Boltzmann Annealing: The cooling schedule used by this method is:

$$T_k = \frac{T_0}{\ln(k+1)}$$

(18)

Boltzmann Annealing uses a random number generator with a gaussian distribution:

$$g(x, s) = \frac{1}{\sqrt{2\pi s}} e^{-\frac{(\Delta x)^2}{2s}}$$

(19)

where $s$ is the current step size and $\Delta x$ is the change in variable $x$.

Fast Annealing: In Fast Annealing the temperature decreases at a faster rate than the other annealing methods described above:

$$T_k = \frac{T_0}{k+1}$$

(20)

This method makes use of the following random number distribution:

$$g(x, s) = \frac{s}{\pi[(\Delta x)^2 + s^2]}$$

(21)

Very-Fast Re-Annealing: — For this technique, the following cooling schedule is used:

$$T_k = T_0 e^{-\sqrt{x}}$$

(22)

Sharma et al. [21] found the optimum value of $c = 0.5$. The number distribution for Very Fast re-annealing is given by:

$$g(x, s) = \frac{1}{2\pi[(\Delta x) + s] \ln(1 + 1/s)}$$

(23)

Table 5 compares the Simulated Annealing various methods showing Original Simulated Annealing to operate the most effectively.

Table 5. Comparison of the various Simulated Annealing techniques

<table>
<thead>
<tr>
<th>Method</th>
<th>Original Simulated Annealing</th>
<th>Boltzmann annealing</th>
<th>Fast annealing</th>
<th>Very fast annealing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ave. Cost (norm.)</td>
<td>1.000</td>
<td>2.047</td>
<td>1.468</td>
<td>2.003</td>
</tr>
<tr>
<td>Median Cost (norm.)</td>
<td>1.000</td>
<td>2.084</td>
<td>1.670</td>
<td>2.004</td>
</tr>
</tbody>
</table>

Table 6. The effect of different number of initial points

<table>
<thead>
<tr>
<th>No. of initial pts</th>
<th>10</th>
<th>20</th>
<th>50</th>
<th>70</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ave. Cost (norm.)</td>
<td>1.070</td>
<td>1.062</td>
<td>1.000</td>
<td>1.154</td>
</tr>
<tr>
<td>Median Cost (norm.)</td>
<td>1.000</td>
<td>1.016</td>
<td>1.041</td>
<td>1.178</td>
</tr>
</tbody>
</table>

3.6 Number of initial configurations and optimisation cycles

The effect of using various number of initial configurations and number of optimisation cycles was investigated for the test case antenna, tables 6 and 7 summarise the results obtained. From these results we see that the optimum number of initial configurations lies between 20–50 and the optimum number of optimisation cycles for this problem is approximately 4000.

3.7 Cost function

The cost function in the Simulated Annealing optimisation algorithm is analogous to the lowering of the energy levels in the bonds of the physical annealing process. As mentioned above a decrease in Cost indicates an increase in the objective function.

$$\text{Cost} = \left(\frac{\text{Req. Gain}}{\text{Actual Gain}}\right)^{g\text{Factor}} + \left(\frac{\text{Actual VSWR}}{\text{Req. VSWR}}\right)^{v\text{Factor}}$$

(24)

where $g\text{Factor}$ and $v\text{Factor}$ are user defined parameters.

Table 7. The effect of different number of optimisation cycles

<table>
<thead>
<tr>
<th>No. of Opt. Cycles</th>
<th>1000</th>
<th>2000</th>
<th>3000</th>
<th>3500</th>
<th>4000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ave. Cost (norm.)</td>
<td>1.633</td>
<td>1.219</td>
<td>1.045</td>
<td>1.008</td>
<td>1.000</td>
</tr>
<tr>
<td>Median Cost (norm.)</td>
<td>1.375</td>
<td>1.121</td>
<td>1.024</td>
<td>1.072</td>
<td>1.000</td>
</tr>
</tbody>
</table>
The best forms of Simulated Annealing found above were used to optimise Yagi and helix design problems. The problems used are identical to those used in section 2.10.

Table 8 shows a summary of the Yagi design results obtained using 20 initial points and 4000 optimisation cycles.

As was the case with the Genetic Algorithm, we see that additionally optimising for element spacing, has little effect on the cost of the results obtained.

Table 9 shows the helix design results using 20 initial points and 450 optimisation cycles. Once again for both types of antennas the optimised results are better than those of the textbook ones.

4. IMPLEMENTATION OF THE OPTIMISATION STRATEGIES

The Genetic Algorithm and Simulated Annealing were implemented in Matlab® along with the various variances mentioned above. In order to effectively use the optimisation strategies, a language was developed for specifying the required optimisation parameters. The language, written in Matlab®, allows any variable for any SuperNEC assembly, or combination of assemblies, to be optimised. This is of great significance as it makes the optimisation techniques more accessible to a user, and would otherwise require the optimisation techniques to be reprogrammed every time the optimisation problem was re-defined. This would require the user to be familiar with the code, and although small changes (eg. changing the min and max values of a variable), would only require a small amount of recoding, large changes (eg. changing the assembly to be optimised), would require a significant amount of reprogramming. In addition the Language also allows for selected variables to vary by a user-defined ratio from another variable [22].

As the simulations of the actual antenna designs in SuperNEC is the part that takes the most time in the optimisation process, the GA implementation also included the creation and saving of lookup tables. Once an antenna design, represented by a chromosome, has been simulated, it is entered into a lookup table along with the chromosome value, the fitness of the chromosome and the output information pertaining to the chromosome such as the gain values and VSWR. This allows a considerable amount of time to be saved as a chromosome that has already been simulated need not be re-simulated again, but rather the values just need be read from the table. This is especially useful once the Genetic Algorithm starts to converge and even more so when a steady-state GA is used.

The effectiveness of using a lookup table was tested using the test case antenna with a steady state GA saving and inserting the best 10 individuals, 80 chromosomes, 100 generations, 0.15 mutation probability, AFP mating scheme and random crossover. The results showed that the optimisation runs that did not use a lookup table took over 5.6 times longer than the ones that did use lookup tables.

The implementation of the Simulated Annealing optimisation algorithm also included, ‘The Method of Cumulative Inversion’, which allows the various random number distributions to be produced using the uniform number generator available in Matlab® [23].
Table 10. Comparison between the Genetic Algorithm and Simulated Annealing for Yagi designs using the same spacing as textbook designs

<table>
<thead>
<tr>
<th></th>
<th>Genetic Algorithm</th>
<th>Simulated Annealing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>gain</td>
<td>VSWR</td>
</tr>
<tr>
<td>5 el. Yagi</td>
<td>11.04</td>
<td>1.0046</td>
</tr>
<tr>
<td>12 el. Yagi</td>
<td>12.11</td>
<td>1.0059</td>
</tr>
<tr>
<td>17 el. Yagi</td>
<td>14.63</td>
<td>1.5710</td>
</tr>
</tbody>
</table>

Table 11. Comparison between the Genetic Algorithm and Simulated Annealing for Yagi designs using different spacing as textbook designs

<table>
<thead>
<tr>
<th></th>
<th>Genetic Algorithm</th>
<th>Simulated Annealing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>gain</td>
<td>VSWR</td>
</tr>
<tr>
<td>5 el. Yagi</td>
<td>10.51</td>
<td>1.0016</td>
</tr>
<tr>
<td>12 el. Yagi</td>
<td>12.86</td>
<td>1.0031</td>
</tr>
<tr>
<td>17 el. Yagi</td>
<td>13.97</td>
<td>1.0079</td>
</tr>
</tbody>
</table>

Table 12. Comparison between the Genetic Algorithm and Simulated Annealing for helix designs

<table>
<thead>
<tr>
<th></th>
<th>Genetic Algorithm</th>
<th>Simulated Annealing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>gain</td>
<td>VSWR</td>
</tr>
<tr>
<td>5 turn</td>
<td>10.88</td>
<td>1.1918</td>
</tr>
<tr>
<td>15 turn</td>
<td>13.93</td>
<td>1.2622</td>
</tr>
<tr>
<td>35 turn</td>
<td>16.22</td>
<td>1.3311</td>
</tr>
</tbody>
</table>

Table 13. Comparison between the Genetic Algorithm and Simulated Annealing for the test case antenna

<table>
<thead>
<tr>
<th></th>
<th>Genetic Algorithm</th>
<th>Simulated Annealing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Best config</td>
<td>gain=11.2dBi</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VSWR=1.0084</td>
</tr>
<tr>
<td></td>
<td>No. within 5% of best config</td>
<td>28 out of 30</td>
</tr>
<tr>
<td></td>
<td>Worst config</td>
<td>gain=10.77dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VSWR=1.0028</td>
</tr>
<tr>
<td>Ave. Run time</td>
<td>716.1 seconds</td>
<td>2274.1 seconds</td>
</tr>
</tbody>
</table>

5. COMPARISON BETWEEN THE GENETIC ALGORITHM AND SIMULATED ANNEALING RESULTS

In this section we will make two sets of comparisons between the optimisation algorithms. In the first comparison we will look at the Yagi and helix antenna optimisation problems already presented. As these antenna designs were only designed for maximum gain and minimum VSWR without any bandwidth considerations, this will serve as the platform of comparison between the two techniques.

The results are summarised by tables 10, 11 showing the comparisons between the Yagi antenna designs using the same spacing and optimised spacings respectively, and table 12 the comparison between the helix antenna designs. At this point it is important to note that although the requirements were set the same for both optimisation techniques, the cost/fitness functions do vary slightly from one another (refer to equations 3 and 24 above). Taking this into account we see that all the Genetic Algorithm designs are better in terms of both cost/fitness functions except for the 12 and 17 element Yagi’s in table 10, in which each design is the best within it’s own costing/fitness function. The only other exception is the 15 turn helix in which the Simulated Annealing design is fractionally better for both cost functions.

In the second set of comparison we will now look at a higher variable problem, namely a 5 element Yagi with all the element lengths allowed to vary independently in length with $300^5$ possible solutions (the test case). Table 13 summarises the results obtained. We see that Simulated Annealing performed poorly with respect to consistency when compared to the Genetic Algorithm.

6. CONCLUSIONS

In this paper various Genetic Algorithms and Simulated Annealing Algorithms were implemented and tested on Yagi and helix antennas. These designs were then compared to their corresponding textbook designs.

From the Genetic Algorithm results presented above we conclude that the best GA employs a steady state GA which saves and inserts the best 10 individuals from the previous generation into the new generation; uses population decimation as the selection strategy; a mutation probability of approximately 15%; and Adjacent-Fitness-Pairing or Best-Mates-Worst as a mating scheme.

The results also show that the algorithm works effectively and efficiently up to a tested problem with $2.45 \times 10^{12}$ possible solutions using only 80 chromosomes and 80 generations.

For Simulated Annealing we can conclude that the Original Simulated Annealing works the best utilising the method proposed by Sechen to determine when equilibrium is reached and approximately 20 initial configurations and a total of 4000 optimisation cycles for the test case antenna.

The results also showed that SA worked well up to a problem with $1.19 \times 10^{12}$ possible solutions but struggled to work consistently when this problem size was
doubled.

From comparisons between the two algorithms it is clear that the Genetic Algorithm worked the most effectively in terms of solutions obtained, efficiently in terms of run time and consistent in terms of sizes of optimisation problems that the algorithm can handle.

REFERENCES


