Genetic Algorithms in Engineering Electromagnetics

J. Michael Johnson and Yahya Rahmat-Samii
Department of Electrical Engineering
University of California, Los Angeles
405 Hilgard Ave.
Los Angeles, CA 90095-1594
Tel: (310) 206-4801 (JMJ); (310) 206-3847 (YRS)
Fax: (310) 206-8495
E-mail: johnson@ee.ucla.edu ; rahmat@ee.ucla.edu
http://www.antlab.ee.ucla.edu

Keywords: Genetic algorithms, absorbing media, antenna arrays, array synthesis, microstrip antennas, natural modes, radar target identification

1. Abstract

This paper presents a tutorial and overview of genetic algorithms for electromagnetic optimization. Genetic-algorithm (GA) optimizers are robust, stochastic search methods modeled on the concepts of natural selection and evolution. The relationship between traditional optimization techniques and the GA is discussed. Step-by-step implementation aspects of the GA are detailed, through an example with the objective of providing useful guidelines for the potential user. Extensive use is made of sidebars and graphical presentation to facilitate understanding. The tutorial is followed by a discussion of several electromagnetic applications in which the GA has proven useful. The applications discussed include the design of lightweight, broadband microwave absorbers, the reduction of array sidelobes in thinned arrays, the design of shaped-beam antenna arrays, the extraction of natural resonance modes of radar targets from backscattered response data, and the design of broadband patch antennas. Genetic-algorithm optimization is shown to be suitable for optimizing a broad class of problems of interest to the electromagnetic community. A comprehensive list of key references, organized by application category, is also provided.

2. Introduction

The application of modern electromagnetic theory to radiation and scattering problems often either requires, or at least benefits from, the use of optimization. Among the typical problems requiring optimization are shaped-reflector antenna design [1], target image reconstruction [2], and layered material, anti-reflective-coating design for low radar cross section (RCS) [3]. Other problems, such as antenna-array beam-pattern shaping [4, 5], while solvable without optimization, are often more readily handled using optimization. This is particularly true when one is faced with realization constraints imposed by manufacturing considerations or environmental factors.

Electromagnetic optimization problems generally involve a large number of parameters. The parameters can be either continuous, discrete, or both, and often include constraints in allowable values. The goal of the optimization is to find a solution that represents a *global* maximum or minimum. In addition, the solution domain of electromagnetic optimization problems often has non-differentiable and/or discontinuous regions, and often utilizes approximations or models of the true electromagnetic phenomena to conserve computational resources. These characteristics sorely test the capabilities of many of the traditional optimization techniques, and often require *hybridization* of traditional optimization methods, if these methods are to be applied at all.

This paper focuses on a relatively new approach to optimization, called the genetic algorithm (GA). Genetic algorithms are robust, stochastic-based search methods, which can handle the common characteristics of electromagnetic optimization problems that are not readily handled by other traditional optimization methods. The goal of this paper is to explore genetic algorithms and their application to a variety of electromagnetic and electromagnetic-related problems. Section 3 gives an overview of the genetic algorithm and its relationship to other traditional optimization methods. Section 4 describes a simple genetic algorithm, and provides some of the details of its implementation and use. Section 4 presents a case study of the use of GAs to find the maxima of several two-dimensional surfaces. Section 5 is intended to acquaint the reader with the GA concepts as used in practice, as well as to attempt to substantiate the claim that GA optimization is applicable to electromagnetic problems. Finally, Section 6 discusses several applications of GAs to electromagnetic optimization to illustrate the wide applicability of GAs, and to emphasize some decisions that must be addressed by the user in applying GA optimization.

Before beginning the presentation of the GA, it may be helpful to consider what kind of problems might benefit from GA optimization. A good candidate problem for GA optimization is illustrated in Figure 1, where the problem is to design a broadband patch antenna. Parameters that are usually included in this type of

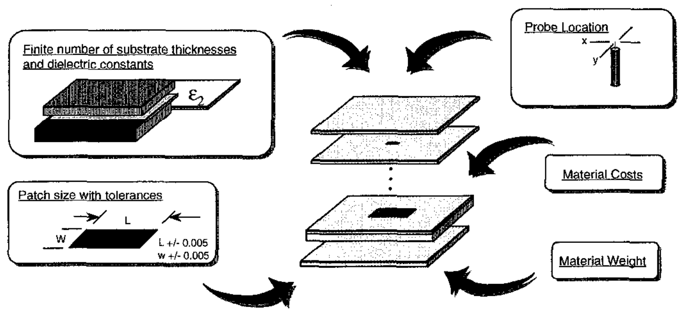

Finite number of substrate thicknesses and dielectric constants
ε_r
Probe Location
x
y
Material Costs
Patch size with tolerances
L
W
L +/- 0.005
w +/- 0.005
Material Weight

Figure 1. Real-life optimization of a broadband patch-antenna design might include finite numbers of available dielectric substrates, material costs, and material weights. GA optimization is robust enough to handle such a diverse set of parameters.

optimization problem include the location of the feed probe, the width and length of the patch(es), and the height of the patch(es) above the ground plane(s). In addition, it may be desirable to include constraints on the available dielectric materials, both in terms of thickness and dielectric constants; tolerance limits on the patch size and probe location; constraints on the weight of the final design; and possibly even cost constraints for the final production model. Given the large number of parameters, and the unavoidable mixture of discrete and continuous parameters involved in this problem, it is virtually impossible to use traditional optimization methods. GA optimizers, on the other hand, can readily handle such a disparate set of optimization parameters. The rest of this paper will attempt to show why GAs would work well for this and related problems in electromagnetics.

3. Genetic algorithm overview

Genetic-algorithm optimizers are robust, stochastic search methods, modeled on the principles and concepts of natural selection and evolution. As an optimizer, the powerful heuristic of the GA is effective at solving complex, combinatorial and related problems. GA optimizers are particularly effective when the goal is to find an approximate global maximum in a high-dimension, multi-modal function domain, in a near-optimal manner. This is particularly true when the problem can be cast in a combinatorial form, a form at which GAs excel. Some important terminology and concepts of GA optimizers are presented in a sidebar.

3.1 Global versus local optimization

Before dealing with the specific details of genetic algorithms and their implementation, it is useful to consider the relationship between GA optimizers and the more traditional, and possibly more familiar, optimization methods. Genetic algorithms are classified as global optimizers, while more familiar, traditional techniques–such as conjugate-gradient and the quasi-Newton methods–are classified as local techniques. Figure 2 illustrates this relationship between the most commonly used optimization methods.

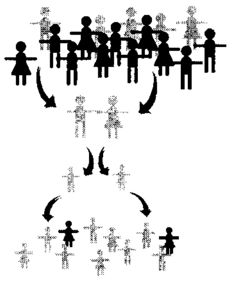
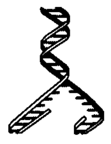
The distinction between local and global search or optimization techniques is that the local techniques produce results that are highly dependent on the starting point or initial guess, while global methods are largely independent of the initial conditions. In addition, local techniques tend to be tightly coupled to the solution domain. This tight coupling enables the local methods to take advantage of the solution-space characteristics, resulting in relatively fast convergence to a local maximum. However, the tight solution-space coupling also places constraints on the solution

The GA in a Nutshell - Concepts and Terminology

Genetic-algorithm (GA) optimizers are robust, stochastic search methods modeled on the principles and concepts of natural selection and evolution. As an optimizer, the powerful heuristic of the GA is effective at solving complex, combinatorial and related problems. GA optimizers are particularly effective when the goal is to find an approximate global maxima in a high-dimension, multi-modal function domain in a near-optimal manner. GAs differ from more conventional techniques in that:

1) they operate on a group (or *population*) of trial solutions in parallel,
2) they normally operate on a coding of the function parameters (a *chromosome*) rather than on the parameters themselves, and
3) they use simple, stochastic operators (*selection*, *crossover*, and *mutation*) to explore the solution domain in search of an optimal solution.

In keeping with the natural-selection analogy, successive *populations* of trial solutions are called *generations*. Subsequent *generations* are made up of *children*, produced through the selective reproduction of pairs of *parents* taken from the current *generation*. A list of some of the commonly encountered GA terms relating to the optimization problem is presented below.

◆ Population	set of trial solutions	
◆ Parent	member of the current generation	
◆ Child	member of the next generation	
◆ Generation	successively created populations (GA iterations)	
◆ Chromosome	coded form of a trial solution vector (string) consisting of genes made of alleles	
◆ Fitness	positive number assigned to an individual representing a measure of goodness	

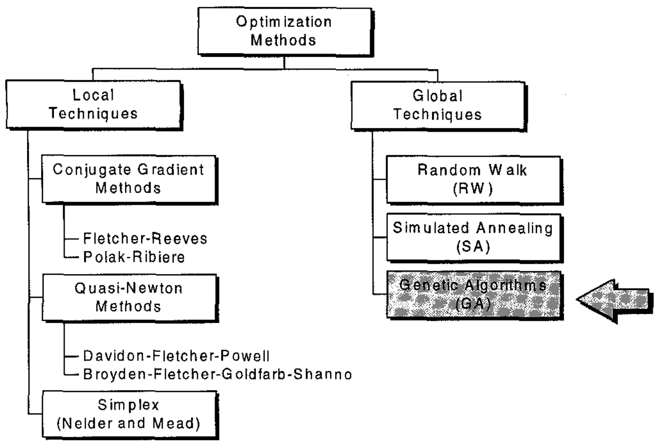


Figure 2. The major optimization methods can be classified as either global or local techniques.

domain, such as differentiability and/or continuity: constraints that can be hard or even impossible to deal with in practice.

The global techniques, on the other hand, are largely independent of and place few constraints on the solution domain. This absence of constraints means that the global methods are much more robust when faced with ill-behaved solution spaces. In particular, global techniques are much better at dealing with solution spaces having discontinuities, constrained parameters, and/or a large number of dimensions with many potential local maxima. The downside to the global method is that it cannot, or at least does not, take advantage of local solution-space characteristics, such as gradients, during the search process, resulting in generally slower convergence than the local techniques.

In electromagnetic-design problems, convergence rate is often not nearly as important as getting a solution. Having found a solution, the ultimate goal is to find the *best* solution, or global maximum. In these applications, global methods are favored over local methods. Global techniques either yield a global or near-global maximum, instead of a local maximum, and often find useful solutions where local techniques cannot. Global methods are particularly useful when dealing with new problems, in which the nature of the solution space is relatively unknown.

Of the global techniques, genetic algorithms are particularly well suited for a broad range of problems encountered in electromagnetics. Genetic algorithms are considerably more efficient, and provide much faster convergence, than random-walk searches. In addition, they are easily programmed and readily implemented. Unlike gradient searches, GA optimizers can readily handle discontinuous and non-differentiable functions. GA optimizers are also well suited for constrained-optimization problems. A comparison among the major features of conjugate-gradient (CG), random-walk (Random), and GA optimization is presented qualitatively in Figure 3.

4. A simple genetic algorithm

This section presents the basic elements of a genetic-algorithm optimizer. It is suggested that the reader read this and the following section, presenting a case study of the use of this GA optimizer, and then re-read this section, to fully appreciate the GA optimizer presented here. A block diagram of a simple genetic-algorithm optimizer is presented in Figure 4. This GA optimizer and the description that follows is modeled after that presented by Goldberg [6]. Extensions to the simple GA optimizer are presented at the end of this section.

The concept of the genetic algorithm, first formalized by Holland [7] and extended to functional optimization by De Jong [8], involves the use of optimization search strategies patterned after the Darwinian notion of natural selection and evolution. During a GA optimization, a set of trial solutions, or individuals, is chosen, and then *evolved* toward an optimal solution, under the *selective pressure* of the fitness function.

In general, a GA optimizer must be able to perform six basic tasks:

1. Encode the solution parameters as genes,
2. Create a string of the genes to form a chromosome,
3. Initialize a starting population,
4. Evaluate and assign fitness values to individuals in the population,
5. Perform reproduction through the fitness-weighted selection of individuals from the population, and
6. Perform recombination and mutation to produce members of the next generation.

	CG	Random	GA
Global Optimization	🙁	😐	🙂
Discontinuous Object Functions	🙁	🙂	🙂
Non-differentiable Object Functions	🙁	🙂	🙂
Convergence Rate	🙂	🙁	😐

Figure 3. Genetic-algorithm (GA) optimization compared qualitatively to conjugate-gradient (CG) and random search (Random).

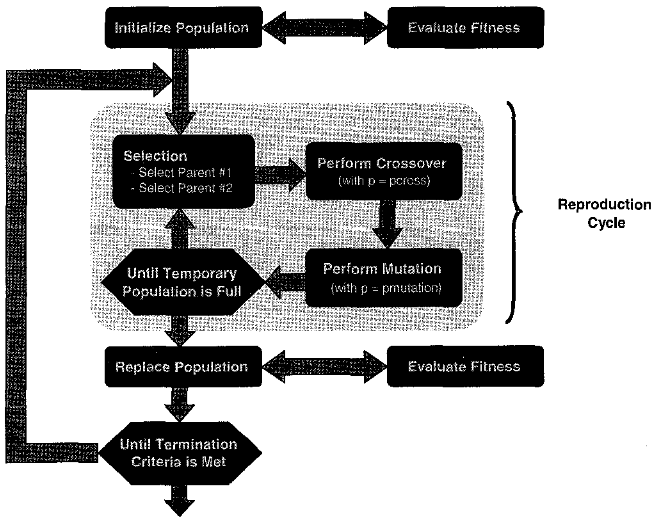


Figure 4. A block diagram of a simple genetic-algorithm optimizer.

These performance requirements lead to the existence of three phases in a typical genetic-algorithm optimization. These phases are (1) initiation, (2) reproduction, and (3) generation replacement. In the simple, typical genetic-algorithm optimizer of Figure 4, initiation consists of filling an initial population with a predetermined number of encoded, usually randomly created, parameter strings, or chromosomes. Each of these chromosomes represents an individual prototype solution or, simply, an *individual*. The set of individuals is called the current generation. Each individual in the set is assigned a fitness value by evaluating the fitness function for each individual.

The reproduction phase produces a new generation from the current generation. In reproduction, a pair of individuals is selected from the population to act as parents. The parents undergo crossover and mutation, thereby producing a pair of children. These children are then placed in the new generation. The selection, crossover, and mutation operations are repeated until enough children have been generated to fill the new generation. In some GA implementations, this scheme is altered slightly. Selection is used to fill the new generation, and then crossover and mutation are applied to the individuals in the new generation, through random pairings. In either case, the new generation replaces the old generation.

In the simple genetic algorithm presented here, the new generation is the same size as, and completely replaces, the current generation. This is known as a *generational* genetic algorithm. Alternatively, in slightly more complicated GA implementations, the new generation can be of a different size than its predecessor, and/or there can be overlap between the new generation and the old generation. GA methods having overlapping populations, called *steady-state* genetic algorithms, will be discussed in more detail, below.

In the generation-replacement phase, the new generation replaces the current generation, and fitness values are evaluated for and assigned to each of the new individuals. The termination criterion is then evaluated and, if it has not been met, the reproduction process is repeated.

4.1 Chromosomes and parameter coding

Genetic algorithms operate on a coding of the parameters, instead of the parameters themselves. The coding is a mapping from the parameter space to the chromosome space that transforms the set of parameters, usually consisting of real numbers, to a finite-length string. The coded parameters, represented by genes in the chromosome, enable the genetic algorithm to proceed in a manner that is independent of the parameters themselves and, therefore, independent of the solution space. Typically, a binary coding is utilized, but any encoding from binary to continuous, floating-point number representations of the parameters can be used.

Generally, it can be shown that using a coding that has some underlying relevance to the problem at hand produces better results. In addition, it is generally best to use the shortest possible alphabet. Binary coding has the shortest possible useful alphabet and, while it may have little relevance to a given problem, it does yield very simple GA operators. In a binary coding, the parameters are each represented by a finite-length binary string. The combination of all of the encoded parameters is a string of ones and zeros. The coded parameters, represented by a set of 1s and 0s for binary coding, are analogous to, and often referred to, as genes.
The genetic algorithm acts on the chromosome to cause an evolution towards an optimal solution. Fitness values provide a measure of the goodness of a given chromosome and, by direct association, the goodness of an individual within the population. Fitness evaluation involves decoding of the chromosome to produce the parameters that are associated with the individual, followed by the evaluation of the fitness function for the decoded parameters.

4.2 Selection strategies

Selection introduces the influence of the fitness function to the genetic-algorithm optimization process. Selection must utilize the fitness of a given individual, since fitness is the measure of the "goodness" of an individual. However, selection cannot be based solely on choosing the best individual, because the best individual may not be very close to the optimal solution. Instead, some chance that relatively unfit individuals are selected must be preserved, to ensure that genes carried by these unfit individuals are not "lost" prematurely from the population. In general, selection involves a mechanism relating an individual's fitness to the average fitness of the population.

A number of selection strategies have been developed and utilized for genetic-algorithm optimization. These strategies are generally classified as either stochastic or deterministic. Usually, selection results in the choice of parents for participation in the reproduction process. Several of the more important and most widely used of these selection strategies are discussed below.

4.2.1 Population decimation

The simplest of the deterministic strategies is population decimation. In population decimation, individuals are ranked from largest to smallest, according to their fitness values. An arbitrary minimum fitness is chosen as a cutoff point, and any individual with a lower fitness than the minimum is removed from the population. The remaining individuals are then used to generate the new generation through random pairing. The pairing and application of GA operators are repeated until the new generation is filled.

The advantage of population-decimation selection lies in its simplicity. All that is required is to determine which individuals are fit enough to remain in the population, and to then provide a means for randomly pairing the individuals that survive the decimation process.

The disadvantage of population decimation is that once an individual has been removed from the population, any unique characteristic of the population possessed by that individual is lost. This loss of diversity is a natural consequence of all successful evolutionary strategies. However, in population decimation, the loss can, and often does, occur long before the beneficial effects of a unique characteristic are recognized by the evolutionary process. The normal action of the genetic algorithm is to combine good individuals with certain characteristics to produce better. Unfortunately, good traits may not be directly associated with the best fitness in the early stages of evolution toward an optimal solution.

When a characteristic is removed from a population by decimation selection, the only way that the characteristic may be reintroduced is through mutation. Mutation is used in GAs as a means for exploring portions of the solution domain. In genetic terms, mutation is a way of adding new genetic material, or characteristics, but it is a very poor mechanism for adding *specific* genetic material. It is best to keep good genes or good portions of genes whenever possible.

It is due to the serious detrimental effects of this premature loss of beneficial characteristics that more sophisticated, stochastic-selection techniques were developed. It is a testament to the GA's robustness as an optimization technique that population decimation works at all.

4.2.2 Proportionate selection

The most popular of the stochastic-selection strategies is proportionate selection, sometimes called roulette-wheel selection [8]. In proportionate selection, individuals are selected based on a probability of selection given in Equation (1), where $f(parent_i)$ is the fitness of the ith parent:

$$p_{selection} = \frac{f(parent_i)}{\sum_i (parent_i)}. \tag{1}$$

The probability of selecting an individual from the population is purely a function of the relative fitness of the individual. Individuals with high fitness will participate in the creation of the next generation more often than less-fit individuals. This has the same effect as the removal of the least fit in population decimation, in that characteristics associated with higher fitness are represented more in subsequent generations. The distinction between population decimation and proportionate selection is that in proportionate selection, there is still a finite probability that highly unfit individuals will participate in at least some of the matings, thereby preserving their genetic information.

Figure 5 depicts the proportionate-selection process as a roulette wheel, where individuals are assigned a space on the wheel that is proportional to their relative fitness. The wheel is "spun," and the individual pointed to at the end of the spin is the individual selected. Proportionate selection can have stochastic errors when population sizes are small. Modifications, such as fitness scaling, have been developed to attempt to deal with these errors.

4.2.3 Tournament selection

A second popular strategy (and perhaps among the most effective for many applications) is tournament selection [9]. Tour-

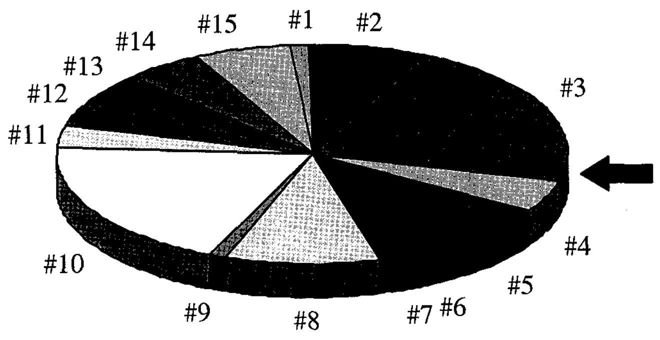


Figure 5. Proportionate selection represented as a roulette wheel, with spaces on the wheel proportional to an individual's relative fitness.

GA Optimization Implementation Guideline

Genetic algorithm (GA) optimizers generally require the user to select among a number of options before beginning the optimization. The options allow the user to tailor the optimization to better fit the optimization job at hand. Aside from the choices of things such as selection type, encoding/decoding scheme, and details of the fitness function, the GA optimizer user must choose the population size, the probability of crossover, and the probability of mutation. If a binary encoding scheme is used, the user must decide how many bits a given parameter is allotted in the chromosome. These choices are summarized below, with some guidelines and comments.

Population Size: Typically 30-100

Comments: Larger populations provide more genetic diversity, enabling faster convergence. Smaller populations yield faster execution, especially when dealing with complicated fitness functions.

Probability of Crossover: Typically 0.6-0.9

Comments: A probability of crossover around 0.7 has been found to be optimal for a wide variety of problems. Generally speaking, crossover is the primary way a GA optimizer searches for new, better solutions. High crossover probability ensures rapid searching.

Probability of Mutation: Typically 0.01-0.1

Comments: The probability of mutation should generally be low. Mutation introduces new genetic material into the search process, but also tends to push the population's average fitness away from the optimal value. Some workers report the probability of mutation at the bit or element level, while others report it at the chromosome level, as is usually done for the probability of crossover.

Replacement Strategy: Generational versus Steady State

Comments: Steady-state generally converges faster than generational replacement in many applications. Lower values of replacement percentage usually converge faster than higher values.

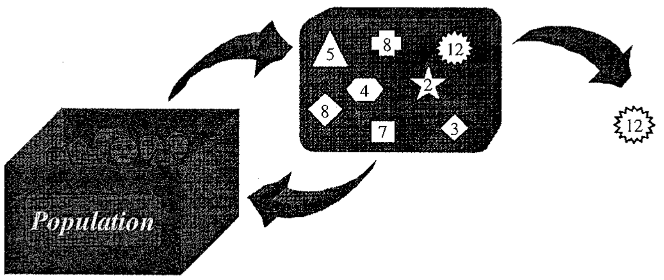


Figure 6. Tournament selection, where N individuals are selected at random from the population, and the individual with the highest fitness in the selected sub-population becomes the selected individual.

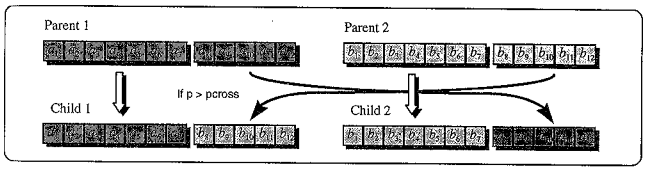


Figure 7. The single-point crossover operator redistributes the characteristics of a pair of parents, and creates a pair of children.

nament selection is depicted in Figure 6. In tournament selection, a sub-population of N individuals is chosen at random from the population. The individuals of this sub-population *compete* on the basis of their fitness. The individual in the sub-population with the highest fitness *wins the tournament*, and becomes the selected individual. All of the sub-population members are then placed back into the general population, and the process is repeated. The most commonly used form of tournament selection is binary tournament selection, in which N equals two.

Both tournament selection and proportionate selection use selection with replacement, so that individuals may, and usually do, participate in multiple pairings. Tournament selection has been shown [9] to provide slightly better convergence toward a solution in the initial stages of the optimization process. Tournament selection also has a somewhat faster execution time. The time complexity of proportionate selection is $O(n^2)$, while tournament selection has $O(n)$ time complexity.

4.3 GA operators

Once a pair of individuals has been selected as parents, a pair of children is created by recombining and mutating the chromosomes of the parents, utilizing the basic genetic-algorithm operators, crossover and mutation. Crossover and mutation are applied with probability p_{cross} and $p_{mutation}$, respectively.

4.3.1 Crossover

The crossover operator accepts the parents and generates two children. Many variations of crossover have been developed. The simplest of these is single-point crossover. In single-point crossover, shown in Figure 7, if $p > p_{cross}$, a random location in the parent's chromosomes is selected. The portion of the chromosome preceding the selected point is copied from parent number 1 to child number 1, and from parent number 2 to child number 2. The portion of the chromosome of parent number 1 following the randomly selected point is placed in the corresponding positions in child number 2, and vice versa for the remaining portion of parent number 2's chromosome. If $p < p_{cross}$, the entire chromosome of parent number 1 is copied into child number 1, and similarly for parent number 2 and child number 2.

The effect of crossover is to rearrange the genes, with the objective of producing better combinations of genes, thereby resulting in more fit individuals. It has been shown [6] that the recombination process is the more important of the two GA operators. Typically, it has been found that probability p_{cross} values of 0.6-0.8 are optimal.

4.3.2 Mutation

The mutation operator provides a means for exploring portions of the solution surface that are not represented in the genetic makeup of the current population. In mutation, if $p > p_{mutation}$, an element in the string making up the chromosome is randomly selected and changed. In the case of binary coding, this amounts to selecting a bit from the chromosome string and inverting it. In other words, a "1" becomes a "0" and a "0" becomes a "1." If higher-order alphabets are used, slightly more complicated forms of mutation are required.

Generally, it has been shown that mutation should occur with a low probability, usually on the order of $p_{mutation} = 0.01 - 0.1$. The action of the mutation operator is illustrated in Figure 8. Figure 8 shows a randomly selected element in a chromosome (shaded element 9) being changed to a new form ($0 \rightarrow 1$ and $1 \rightarrow 0$ in the binary case).

4.4 Fitness functions

The fitness function, or object function, is used to assign a fitness value to each of the individuals in the GA population. The fitness function is the ***only connection*** between the physical problem being optimized and the genetic algorithm. The only constraints on the form and content of the fitness function, imposed by the simple GA, are that (1) the fitness value returned by the fitness function is in some manner proportional to the *goodness* of a given trial solution, and (2) that the fitness be a positive value. In some implementations of the GA optimizer, the constraint that the fitness value be a positive value is not even required.

Genetic algorithms are maximizers by nature. To find a minimum, a slight modification to the usual functional form is required. One simple method is to define the fitness function for minimizing $f(x[n])$ as in Equation (2):

$$fitness = Max - f(x[n]), \tag{2}$$

where Max is a positive number larger than the largest expected value of $f(x[n])$. Alternatively, minimization in the GA can be

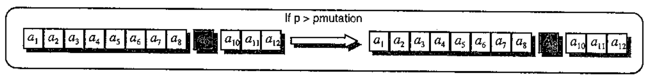


Figure 8. The mutation operator randomly modifies elements within the chromosome.

accomplished through appropriate fitness scaling, or through the use of modified selection operators that seek out lower fitness values.

4.5 Extensions and improvements to the simple GA optimizer

Many extensions and improvements to the simple GA optimizer have been developed and utilized. Among these are the concepts of elitism [6], the use of steady state algorithms [10], the use of real coded parameters [11], the concept of community based genetic algorithms [12], and approaches applicable to the traveling-salesman problem. Several of these extensions are discussed in more detail, below.

4.5.1 Elitist strategy

One of the most important practical extensions is the elitist strategy. In the simple genetic algorithm, it is possible for the next generation to have a *best individual* with a lower fitness than a preceding generation. This loss of the best individual occurs due to the probabilistic nature of the GA selection, crossover, and mutation. A simple test can be added to verify that the best individual in the new generation is at least as good as the one from the preceding generation. If a decrease in the fitness of the best individual is observed, the best individual from the preceding generation can be inserted into the new generation [6]. Saving and inserting the best individual from the last generation is known as the elitist strategy, or simply elitism. Elitism can be used to insure that there is a monotonic increase in the best fitness in the population as a function of time.

4.5.2 Steady-state genetic algorithms

A second important extension to the simple GA, presented above, is the use of overlapping generations. The simple GA, presented above and shown in the block diagram of Figure 4, is more properly known as a *generational* replacement GA, or simply a generational GA. Each reproduction cycle in a generational GA produces an entirely new generation of children, which then replaces the parent generation. An alternative to this approach is the *steady-state* replacement GA, or simply, the steady-state GA [10]. Steady-state GAs have been shown to exhibit advantages, such as faster convergence, in many applications. In the steady-state GA, only a portion of the current generation is replaced by children generated in the reproductive cycle, resulting in overlapping generations. The percentage or portion of the current generation that is to be replaced is a parameter set by the user, and can range from a single individual to 100% of the population.

Steady-state GAs can be implemented as having a block diagram that is essentially identical to that shown in Figure 4. The difference between the generational approach of Figure 4 and the steady-state approach is in the size of the temporary population. The actions of selection, crossover, and mutation create a temporary population of children. The temporary population of children is then inserted into the parent population, by replacing selected individuals in the parent population. A variant to steady-state GAs, described above, is to insert the temporary population into the parent population, producing a temporarily expanded parent population. Individuals are then selected and deleted from the expanded population until the original population size is reached again. In addition to timing of replacement and on what criteria population replacement takes place, consideration must be given to how population replacement is accomplished.

In generational replacement, only the method of selection has to be determined. In steady-state replacement, however, both the method of selecting parents for participation in reproduction and the method of selecting individuals that are to be replaced must be determined. Many methods of selecting the individuals to be replaced have been reported, including replacing the least fit, replacing selected parents, replacing random individuals, and using a second stochastic selection to select individuals to be replaced. If replacement of the least-fit individuals is used, the preservation of the fittest individuals is guaranteed, and the use of elitism is not required.

Other extensions, such as dominant and recessive genes, and the concept of the niche and speciation, have been and are continuing to be explored. However, the complexity and costs of these techniques may outweigh the marginal benefits gained from their use in many practical electromagnetic applications.

4.5.3 The traveling-salesman problem (TSP)

Some problems are best handled by identifying particular features of the problem, and then adapting or extending the genetic algorithm to better accommodate those features. This is particularly true of problems such as the traveling-salesman problem (TSP). The classical TSP involves a salesman who is going to visit several cities. Each city is to be visited by the salesman only once, and the salesman is to return home at the end of the trip. The problem is to find an ordering of the cities that minimizes the total trip length. The traveling-salesman type of problem arises in electromagnetics in several instances, including optimal circuit layout, wireless network layout, and related problems.

The traditional, simple GA operators need only a minor modification to greatly improve their performance in the TSP. A modification to the crossover operator, discussed above in Section 4.3, called a partially matched crossover, is designed to allow a GA to be used for the TSP [6]. This method is illustrated in Figure 9. Partially matched crossover eliminates the generation of invalid trips by the crossover operation, thereby reducing the total space that must be searched to find an optimal solution. A useful modification of the mutation operator, discussed in Section 4.3 for the TSP, is depicted in Figure 10. In this version of mutation, instead of randomly changing an element in the chromosome, a pair of elements is interchanged.

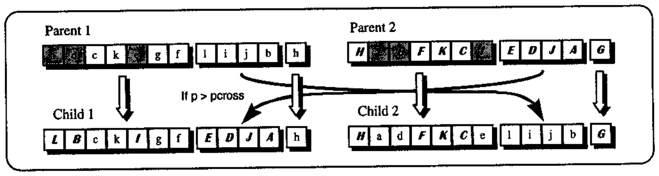


Figure 9. The partially matched crossover operator redistributes the characteristics of a pair of parents and creates a pair of children, while preserving a valid tour.

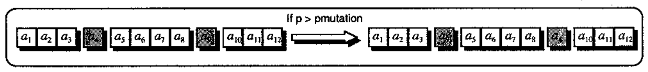


Figure 10. The TSP mutation operator randomly selects and interchanges a pair of elements within the chromosome.

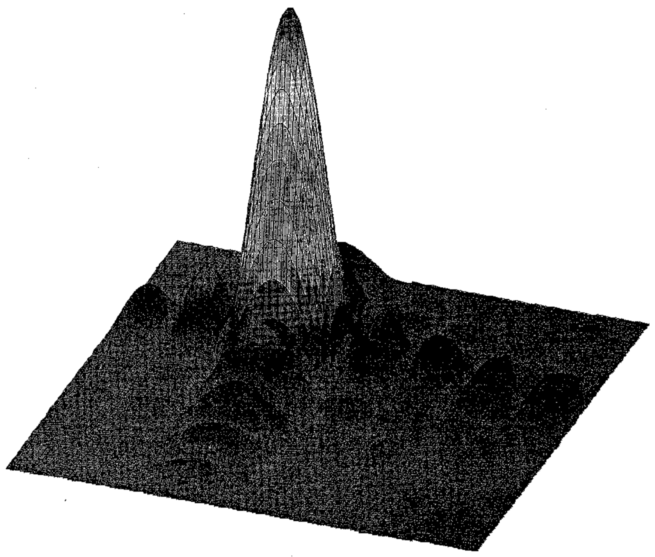


Figure 11. A plot of the solution surface for the two-dimensional magnitude sinc function example problem of Equation (3), which has a global maximum at ($x = 3.0$, $y = 3.0$).

5. A step-by-step implementation of a case study

In order to better understand the operation of the genetic-algorithm optimizer, described above, it is helpful to consider a specific, albeit arbitrary, example. Consider, for example, the use of a GA optimizer to find the maximum value of function within a constrained range of input parameters. Specifically, consider the function $f(x,y)$, given by Equation (3) and plotted in Figure 11, with the parameters x and y constrained to the range from 0.0 to 8.0. Equation (3) is a two-dimensional function, having parameters x and y, and consists of the product of a pair of the magnitude sinc functions:

$$f(x,y) = \left|\frac{\sin[\pi(x-3)]}{\pi(x-3)}\right| \left|\frac{\sin[\pi(y-3)]}{\pi(y-3)}\right|. \tag{3}$$

As a candidate for an optimization-based determination of a maximum value, Equation (3) presents a number of problems. While Equation (3) has a well-defined, global maximum within the constrained parameter range, it also has a number of local maxima, along with non-differentiable regions resulting from the magnitude operator. A GA optimizer can be used to find the maximum of the function within the range $x, y \in \{0,8\}$ by simply defining the $f(x,y)$ of Equation (3) to be the fitness function.

The first step in using a genetic-algorithm optimizer on this problem is to choose a coding, or mapping of the parameters into genes. In this case, a binary-string coding will be used. Since the problem represented by Equation (3) is a two-dimensional problem, the most natural chromosome structure is a two-part string. The first part of the string, or first gene, corresponds to the first parameter, x, and the second gene, representing the parameter y, is contained in the second portion of the string.

GA optimization actually operates on the coded form of the parameters, so what is actually needed in practice is a way to decode the chromosomes into the parameters for use in the fitness function given by Equation (3), as opposed to a way of encoding the parameters. A binary-decoding method, from an N-bit gene to a pair of parameters x and y, which incorporates the range constraints of the problem, is given by Equation (4):

$$p = \left(\frac{p_{max} - p_{min}}{2^N - 1}\right) \sum_{n=0}^{N-1} 2^n b_n + p_{min}, \tag{4}$$

where p is either x or y, p_{min} and p_{max} are the minimum and maximum range bounds corresponding to the parameter p, and b_n is the binary bit in the nth place along the gene corresponding to parameter p. Examination of Equation (4) reveals that it is a scaled binary-to-decimal conversion, where the scaling adjusts the range to lie between p_{min} and p_{max}. In this example, $p_{min} = 0.0$ and $p_{max} = 2.0$, for both $p = x$ and $p = y$. Alternatively, a gray code could be used. With Equation (4), any chromosome created during the GA optimization can be decoded into the parameters x and y, and a fitness value can be assigned by using Equation (3).

Having chosen a coding, the GA optimization can begin by creating a population of chromosomes, and calculating fitness values for each of these individuals. The chromosomes of the starting population are typically created by filling binary arrays of 32 elements with randomly generated 1s and 0s. A random-number generator that has a uniform distribution, a long (preferably infinite) repetition period, and a range between 0 and 1, is called successively for each bit in each chromosome in the population. If the random-number generator returns a value greater than 0.5, a "1" is inserted in the chromosome. For anything less than 0.5, a zero is inserted into the chromosome. Decoding of the chromosomes into parameter values is then performed, using Equation (4), followed by the calculation of fitness values for each individual, using the decoded parameters and the fitness function of Equation (3). The use of a randomized initial population attempts to insure that the solution domain is sampled in an unbiased, uniform way, thereby providing a reasonable starting point for the optimization.

Table 1 tabulates a typical initial or starting population, resulting when 16 bits were used for each gene in the chromosomes, and 20 individuals were included in the population. The size of the population is up to the user, and is based on the expected complexity of the solution space and the number of parameters that are involved. Generally, however, a population of between 20 and 100 individuals seems to work well for most practical problems.

In the case of Table 1, the best individual in the starting population happens to be individual 6, with a fitness of 0.489281. The decoded parameters of individual 6 are {x = 3.548760, y = 2.694621}. Since the maximum of the function is known to occur at {x = 3.000, y = 3.000}, it can be seen that individual 6 is not very close to the maximum.

Table 1: Genetic algorithm population details after initialization of the population for the 2D magnitude SINC example.

Indiv	Chromosome	Parameters {x,y}	Fitness
0	0100101011000000 0111110010000010	2.335973, 3.890929	0.050072
1	0011010111101011 0111111100001100	1.684962, 3.970275	0.006189
2	0011010111001111 0101001000111010	1.681544, 2.569619	0.146716
3	0001000010101011 0110111100001001	0.520882, 3.469902	0.086401
⋮	⋮	⋮	⋮
6	0111000110001111 0101011000111010	3.548760, 2.694621	0.489281
⋮	⋮	⋮	⋮
18	0100011110110000 1000100010111100	2.240268, 4.273015	0.054290
19	0110101001001110 0111010000001000	3.322072, 3.626032	0.393072

After choosing a coding and filling the initial population, the genetic algorithm can begin optimizing. Successive generations are produced by the application of selection, crossover, and mutation. In the example illustrated here, tournament selection was used. The probability of crossover was set at 0.7, and the probability of mutation was set at 0.05 for each chromosome. The crossover and mutation probabilities used here were in the middle of the normally acceptable ranges of 0.6-0.8 for p_{cross} and 0.01-0.1 for $p_{mutation}$, respectively. Elitism was also employed.

For a particular trial of this example, a fitness value of $f(x,y) = 0.998757$ was achieved after only eight generations. An optimal fitness value of $f(x,y) = 1.00000$ was achieved after 41 generations, with the parameters $x = 3.000290$ and $y = 2.999924$. So, within about 160 evaluations of the fitness function, a near-optimal value was located on the solution surface.

Since GA optimization involves various randomized operations, the progress towards the optimum is not always the same, and convergence characteristics are often best studied in terms of an average over many trials. Figure 12 shows the average progress of the GA optimization as a function of the number of generations. Since the population included 20 individuals, each generation required up to 20 evaluations of the fitness function, Equation (3). Two of the curves presented in Figure 12 show the convergence of the GA optimizer, the first representing an average of 20 independent trials, and the second showing the average of 50 independent trials. As can be seen, the GA optimizer, on average, makes rapid progress towards the optimum solution. After only seven to nine generations (140 to 180 evaluations of the fitness function), a near optimum had been located, on average.

These results can be compared to those produced by a completely random search method. Random-search results, averaged over 20, 50, and 1000 independent trials, are also plotted in Figure 12. To facilitate comparison with the GA results, the random search in this case was constrained to the $x, y \in \{0.0, 8.0\}$ range. In addition, groups of 20 randomly selected pairs of x and y values were evaluated using Equation (3), and the best result from each

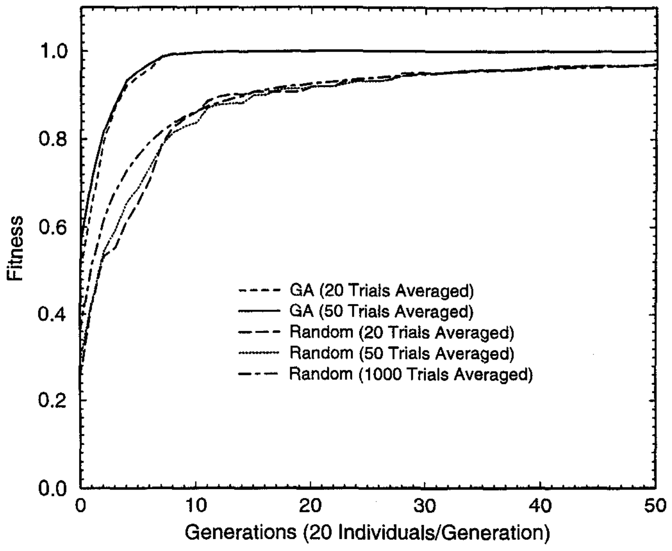


Figure 12. The genetic-algorithm optimization converges to the optimal solution (fitness = 1.0) much faster, on average, and with smaller variance over a number of successive, independent trials than the random-search method.

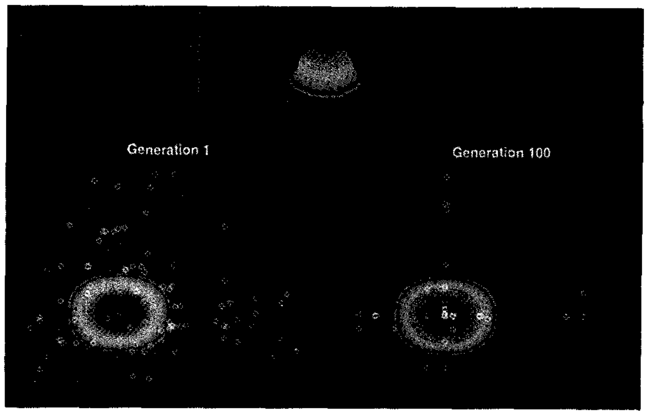


Figure 13. Distribution of a GA population at the first generation and after 100 generations, showing the convergence of the population toward the peak, and the general alignment of the population along planes that intersect the peak.

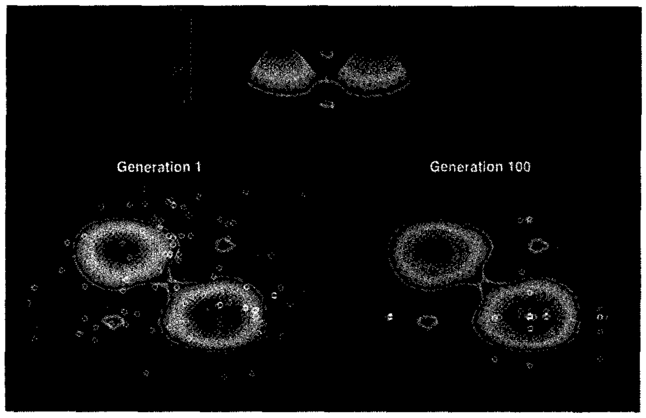


Figure 14. Distribution of a GA population on a bi-modal surface at the initialization point and after 100 generations, once again showing alignment of the population on planes intersecting the peak.

group was recorded. This is analogous to the concept of generations in the GA optimization. In addition, if a group or generation failed to produce a better result than the last group, the best result from the last group was carried over as the current best value. As can be seen, the GA optimizer out-performed the random-search method in terms of an average convergence rate. The random-search method did not converge to the optimal value until well beyond 100 generations.

The variance between GA trials, when compared to the random-search method, was also relatively small. The agreement between 20 and 50 averaged trials for the GA optimization was very good, while there was more difference exhibited in the case of the random search. GA has been called a "directed" random search, a notion supported by the results presented in Figure 12.

In the case presented above, the termination criterion was simply to test whether a specified number of generations had been reached. Other termination conditions could have been used, but would have added little to such a simple problem.

Another way to view the operation of the genetic algorithm is to consider its action at the population level. Again, consider a two-dimensional magnitude sinc example, similar to that above. In this example, the population size was set at 100, to provide enough individuals to visually illustrate the population dynamics to the GA optimizer.

Figure 13 is a plot of the distribution of a GA population at the beginning of the optimization and after 100 generations. At the beginning of the optimization, the population was randomly distributed on the fitness-function domain. After 100 generations, the population was no longer randomly distributed, but instead was largely grouped on the main peak. Those individuals in the population that were not on the peak were generally lined up with the peak. This shows that the GA optimizer was working with the population as a whole. Ultimately, the GA optimizer tried to put all individuals to the peak; the action of the mutation operator tended to work against this goal, especially toward the end of the optimization process.

5.1 A more difficult problem

Now consider a more difficult two-dimensional problem, with two relatively equal-height maximal regions. In this case, let the solution surface be represented by a sum of two offset magnitude sinc functions, similar to those used above, where the smaller peak was chosen to be 90% as high as the higher peak. This example is depicted in Figure 14, with 100 individuals in the population.

Again, the initial population is chosen to sample the potential solution domain as evenly as possible by filling the initial chromosomes with a random selection of 1s and 0s. After 100 generations, it was found again that most of the population was grouped on or around the highest peak. In fact, no individuals ended up on the lower peak after 100 generations. This result can be explained by noting that a GA works by selecting individuals with the best characteristics. The individuals that occupy one of the two coordinate axes that intersect the larger peak are favored by the GA optimization and, therefore, tend to predominate in the population. Of course, it is expected that the action of the mutation operator would occasionally introduce individuals that could end up on the lower peak. In all of the examples discussed above, the goal was to find the peak value on the surface. The presence of a few less-fit individuals is of little consequence.

5.2 S comparison of steady-state and generational GA performance

Consider a final example, comparing the performance of a generational and a steady-state genetic algorithm. The problem is to find the maximum in a multi-dimensional hypersurface described by

$$f(x_1 \ldots x_n) = \prod_{i=0}^{n} \left| \frac{\sin(x_i - 2)}{x_i - 2} \right|, \qquad (5)$$

where the variables are constrained to lie in the range $0 \le x_i \le 8.0$, and $n = 8$. The maximum value of the fitness functions is located at $x_i = 2.0$ for all i. As before, the choice of the function of Equation (5) is arbitrary, and is simply intended to provide a sufficiently complex, multi-dimensional example to illustrate the points below.

The optimization problem was conducted using a generational GA and a steady-state GA, and roulette-wheel selection. The steady-state GA that was used created children, inserted them into the population, and then deleted the worst individuals from the temporarily expanded parent/child population. Three different replacement percentages were used in the steady-state GA runs. A total of 50 separate optimization runs were conducted, and the average convergence properties were collected for each of the generational and steady-state configurations. The population size was 100 individuals, the probability of crossover was 0.7, and the probability of mutation was 0.01. The average progress of the best individual toward the goal is plotted against the number of times Equation (5) was evaluated in Figure 15, for the configurations under consideration. As can be seen in Figure 15, decreasing the percentage replacement generally increased the convergence rate. The generational GA approach exhibited slower convergence than the steady-state GA approach for all tested percent replacements, including 100% replacement. In addition, the generational GA had significantly slower convergence in the later stages of the optimization process.

The differences in the observed convergence results can be attributed to several factors. First, the lower the percentage replacement in the steady-state GA, the sooner the newly created genetic material is able to participate in the reproductive process. For instance, when replacement is 100% in the above example, 100 evaluations of the fitness function take place prior to the insertion of the individuals into the "breeding" population. Secondly, the generational GA loses out to the steady-state GA because of the total replacement of the parent population by the child population. This total replacement takes place without regard for the fitness of the parents or the newly created children, and has the potential for eliminating large numbers of highly fit individuals in the later stages of the optimization process. Since it is likely that many newly created children will be less fit than many of the parents, the steady-state GA that was used has the advantage of preserving large numbers of the most fit of both the parent and child populations. Finally, progress in GA optimization occurs primarily as a result of the recombination process embodied in the crossover operator. The steady-state GA with low replacement percentages provides a large pool of genetic variability, while limiting the number of fitness evaluations required. In a population of 100

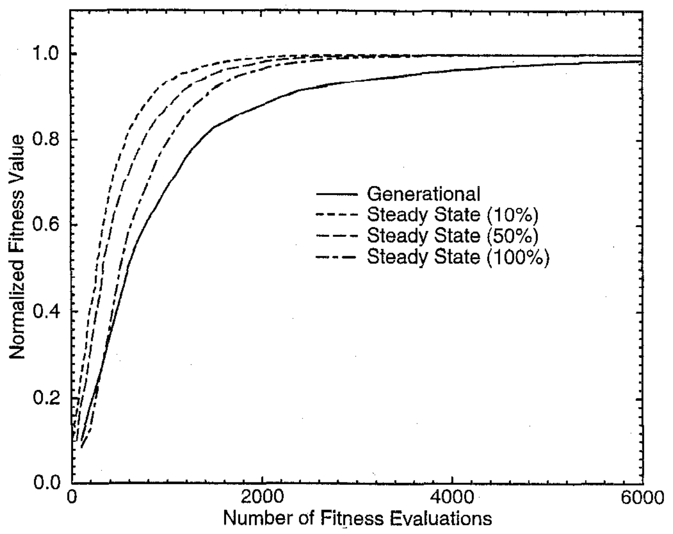


Figure 15. Comparison between average progress towards convergence for generational and steady-state replacement schemes, showing better convergence for the steady-state schemes.

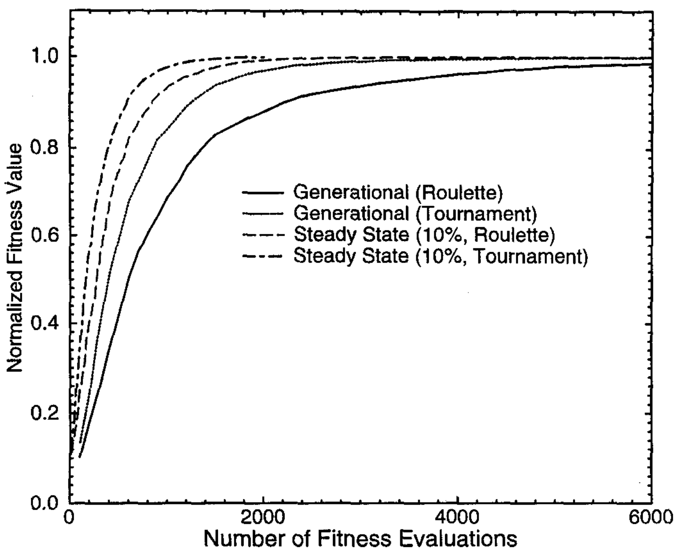


Figure 16. Comparison between average progress towards convergence for roulette wheel versus tournament selection, in both generational and steady-state replacement schemes.

individuals, a 10% replacement in the steady-state GA requires 10 fitness-function evaluations per generation. There are, however, 100 individuals that can participate in the reproductive selection process. Contrasting this to the generational case, to achieve only 10 fitness-function evaluations per generation, the population size would be 10. A population size of only 10 individuals would likely be far too few individuals to ensure proper GA optimization, in most cases.

A second convergence comparison, using the function of Equation (5), is provided in Figure 16. In this figure, the average progress of the generational GA, with roulette-wheel and binary-tournament selection, is plotted along with roulette-wheel and binary-tournament selection cases for a 10% replacement steady-state GA. All other of the parameters are as they where for the case presented above, in Figure 15. Binary-tournament selection improved the convergence of both the generational and the steady-state GAs. However, the steady-state GAs still provide an advantage over the generational GAs in these examples.

The generational GA is slightly more computationally efficient if the cost of evaluating the fitness function is ignored. The lack of efficiency in the steady-state GA results from the need to perform selection of the individuals to be deleted. Generally, however, in electromagnetic problems, the cost of the fitness-function evaluation so completely outweighs the cost of the GA operations that the computational inefficiency of the steady-state GA is of no consequence.

6. Example applications of genetic algorithms in electromagnetics

Genetic algorithms have found, and are continuing to find, numerous applications to real-world problems in engineering and computer science. In particular, GA optimization has been applied successfully to a wide variety of electromagnetic problems. Among these successful applications are the design of broadband microwave absorbers [3], the synthesis of antenna arrays [4, 13, 14, 15] and wire antennas of various forms [15, 17], the design of frequency-selective surfaces [18], radar target recognition and backscattering problems [2], and wireless-network layout [19]. Five of these applications of GAs to electromagnetic problems are described in more detail below. This section provides a representative picture of the range of genetic-algorithm optimization applications possible in electromagnetics. The selection of these particular examples is intended to demonstrate the breadth of the applicability of GA optimization to electromagnetics.

6.1 Broadband multi-layer microwave absorbers

Genetic algorithms have been used to synthesize light-weight, broadband, multi-layer, microwave absorbers [3]. The goal was to produce light-weight, relatively thin multi-layer microwave absorbers, backed by a perfect conductor, that could be used for radar-absorbing applications on airborne platforms, where weight and thickness are critical. Additionally, the multi-layer materials were to be constructed from a discrete set of N_m available material types, with frequency-dependent permittivities and permeabilities. Optimization of reflectivity over a large number of incident angles was also included in the optimization goal. The idea behind this optimization and its application is shown in Figure 17.

A database, listing the available materials, was developed. A binary encoding was used, wherein each layer was allocated several bits within the chromosome. Some of these bits encoded the thickness of the layer, using a standard finite-length binary encoding, while the remaining bits identified the material by pointing to an entry in the material database. The genetic algorithm utilized a fitness function that minimized reflectivity over a range of angles and frequencies, and included penalties for total thickness. Proportionate selection was used as the selection method.

Successful designs of absorbing coatings, with minimized thicknesses and minimized reflection coefficients in the 0.2-2 GHz, 0.5-8 GHz, and 2-8 GHz ranges were reported in [3].

6.2 Low sidelobe levels in thinned and non-uniform arrays

In this application of genetic algorithms, the goal was to minimize the sidelobes produced by an antenna array by thinning or removing elements making up the array [13]. Thinning was

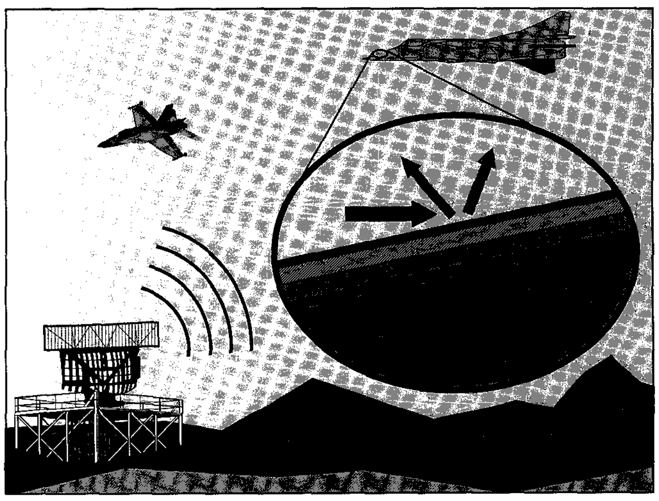


Figure 17. Broadband multi-layer absorbers for low radar cross section.

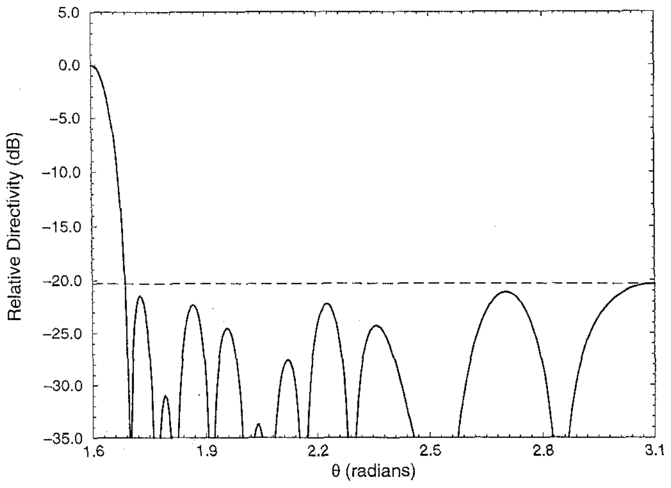


Figure 18. Results of thinning an array to achieve an optimal low sidelobe pattern using the GA.

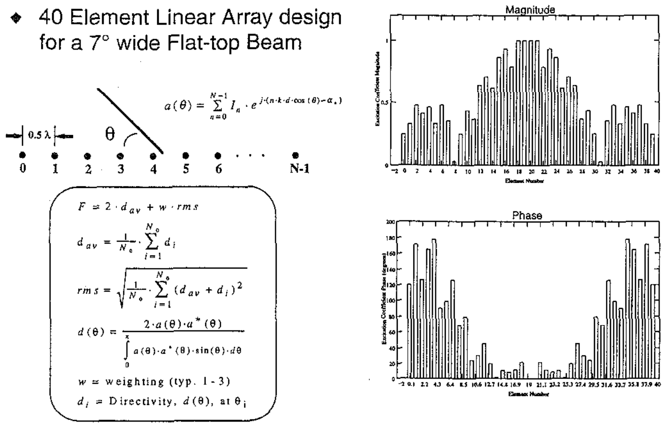

40 Element Linear Array design for a 7° wide Flat-top Beam
$a(\theta) = \sum_{n=0}^{N-1} I_n \cdot e^{j\cdot(n\cdot k\cdot d\cdot\cos(\theta)-\alpha_n)}$
0.5λ
θ
0 1 2 3 4 5 6 . . . N-1
$F = 2 \cdot d_{av} + w \cdot rms$
$d_{av} = \frac{1}{N_o} \cdot \sum_{i=1}^{N_o} d_i$
$rms = \sqrt{\frac{1}{N_o} \cdot \sum_{i=1}^{N_o} (d_{av} + d_i)^2}$
$d(\theta) = \frac{2 \cdot a(\theta) \cdot a^*(\theta)}{\int_0^\pi a(\theta) \cdot a^*(\theta) \cdot \sin(\theta) \cdot d\theta}$
w = weighting (typ. 1 - 3)
d_i = Directivity, $d(\theta)$, at θ_i
Magnitude
Phase

Figure 19. The geometry and fitness function, as well as the amplitude and phase of the optimized excitation coefficients, for the shaped-beam genetic algorithm array-design example.

accomplished in the fitness function by switching on and off elements within the array, and testing for and locating the maximum sidelobe levels.

The chromosomes in this case consisted of binary strings, with one bit per antenna array element. Elements that were "on" were represented by a "1," and elements that were "off" were represented by a "0." Population decimation was used as the selection criterion. Here, again, the parameters are naturally highly discretized (i.e., 1 = on and 0 = off). In fact, this is a case where the natural coding, a binary, is also the optimal coding.

An example result, similar to those reported by [13], is provided in Figure 18. The case in Figure 18 involved an array with 0.3λ element spacing and a 40-element array. The goal, as in [13], was to minimize the maximum sidelobe level by selectively removing elements from the array. Even symmetry about the center of the array was assumed. A maximum sidelobe level of better than -20 dB was achieved. It should be noted that this result was achieved with less than 5000 evaluations (100 individuals in the population and 50 generations) of the fitness function, and represents a stable, repeatable result. To exhaustively survey all possible combinations of present and absent array elements in this case would have required $2^{40} = 1.0995 \times 10^{12}$ fitness evaluations. For large arrays, the GA is obviously a better approach than the prohibitively expensive approach of exhaustive examination of possible solutions.

Sidelobe levels of -22.09 dB are reported by [13] for linear arrays with 200 elements: 77% of the elements were "on," and had 0.5λ nominal spacing. Two-dimensional arrays were also optimized successfully using the genetic-algorithm approach.

6.3 Shaped-beam arrays

Genetic algorithms have been used to synthesize shaped-beam antenna patterns for linear arrays [5]. The goal in this application was to select a set of amplitude and phase weights to achieve a narrow, flat-topped beam design, given a finite set of available amplitude and phase states. The geometry, fitness function, and resultant excitation coefficients for the linear array are presented in Figure 19.

Two different approaches were studied. In the first approach, linearly variable weights were assumed. The second approach assumed that the amplitude and phase weights were being generated by digital attenuators and digital phase shifters, with relatively few available states. In the first case, the linear amplitude and phase ranges were coded as 16-bit natural binary numbers. In the second case, a mapping from the available digital amplitude/phase states to a binary representation was made. Both proportionate and tournament-selection strategies were investigated.

Good results were achieved with both continuously variable (albeit discretized due to parameter encoding) and highly discretized parameter sets. Tournament selection was found to be more effective at reaching an optimum result, both in terms of maximizing the fitness values and in terms of achieving faster convergence.

Of particular interest was the result that, given a limited number of phase and amplitude states, the GA optimization produced better results operating on the states directly, than were achieved by optimizing the linearly variable weights and then rounding off to the nearest available digital state. These results are presented in Figure 20, where the use of GA optimization to directly choose the discrete values of phase and amplitude can be seen to have produced a flatter beam than the method of rounding-off the continuous results.

6.4 Natural resonance extraction

Genetic algorithms have been applied to extracting the natural resonance frequencies from a radar target [2] using multiple data sets. The chief application of resonance extraction from radar returns is for automatic target recognition. In the genetic algorithm, performance and results are compared to results generated by the constrained E-pulse technique.

Population decimation was used as the selection technique in this work. The goal was to minimize the residual $R(\sigma_n, \omega_n)$ with respect to $\{\sigma_n, \omega_n\}$, where

$$R(\{\sigma_n, \omega_n\}) = \sum_k \frac{1}{\varepsilon_k} \left\{ \sum_i \left[m_{k,i} - g_k(t_i) \right]^2 \right\}. \qquad (6)$$

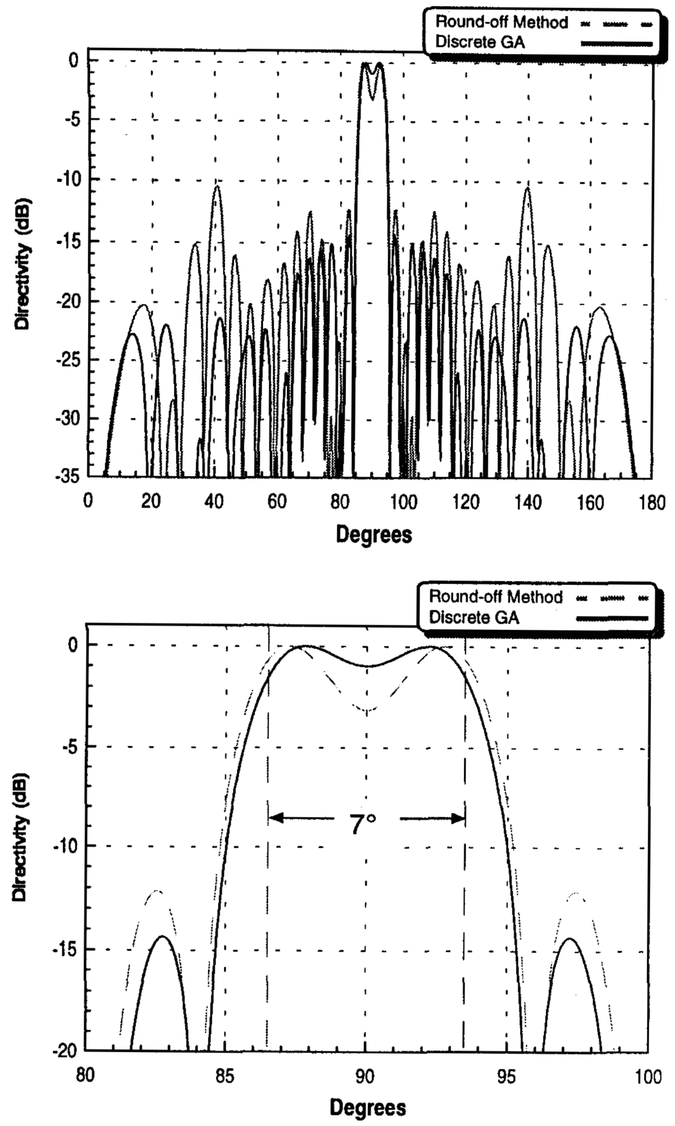


Figure 20. Discretized versus rounded-off flat-top beam patterns produced using genetic-algorithm optimization.

$\{m_{k,j}\} = \{m_k(t_i)\}$ is the sampled late-time response of the target, measured at several different aspect angles, $k = 1, \ldots, K$; ε_k is the energy in the kth response in Equation (6). The fitting function, $g_k(t)$, for the kth aspect angle is given by Equation (7):

$$g_k(t) = \sum_{n=1}^{N} a_{k,n} e^{\sigma_n t} \cos(\omega_n t + \phi_{k,n}). \qquad (7)$$

The minimization problem was converted to a maximization problem, using the approach of Equation (2), with $f(x[n]) = R(\{\sigma_n, \omega_n\})$ of Equation (6). Resonance parameters were coded using a natural binary coding. The results reported indicate that the genetic-algorithm method produced better results than previously explored methods, including the constrained E-pulse technique (CET). The GA approach required less computational effort than the CET, did not require an initial guess, and it simplified the inclusion of constraints on the damping coefficients. The GA approach worked well with large numbers of modes and large data sets, producing rapid and efficient resonance extraction.

6.5 Broadband patch-antenna design

Genetic algorithms have recently been used successfully in the design of broadband patch antennas [20]. The design goal in this example was to produce sub-wavelength patch antennas with significantly wider operational bandwidths than are typical of classical designs. A novel direct coupling of the GA with the Method of Moments was developed, in which the binary string of the chromosome was used to represent the presence or absence of a subsection of metal in the patch. To illustrate the use of this technique, a $\lambda/2$ square patch, fed by a simple wire feed, was optimized for improved operational bandwidth. The bandwidth of the patch antenna was increased from 6% to over a 20% frequency bandwidth. A graph showing the return-loss performance before and after optimization is provided in Figure 21. The shape of the patch before and after optimization is shown in Figure 22.

As can be seen in Figure 21, the original square-patch antenna, as indicated by a dotted line, had a 2:1 VSWR bandwidth of approximately 6%. The original patch design consisted of a 0.48 × 0.48 cm patch, suspended 0.048 cm above an infinite ground

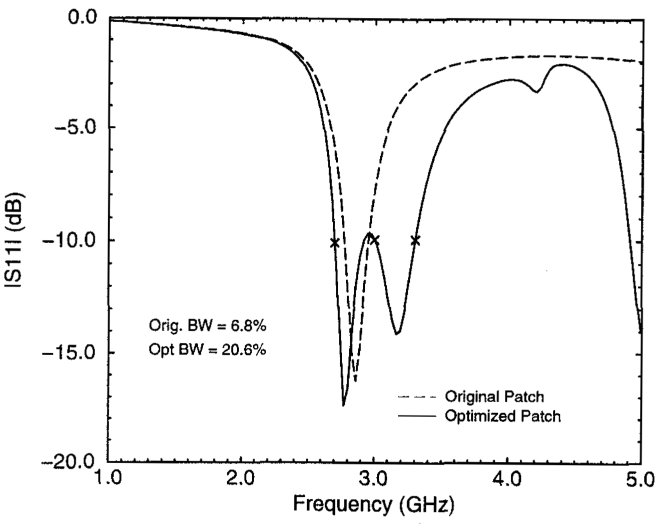


Figure 21. Magnitude S11 results for the rectangular patch and the GA optimized patch.

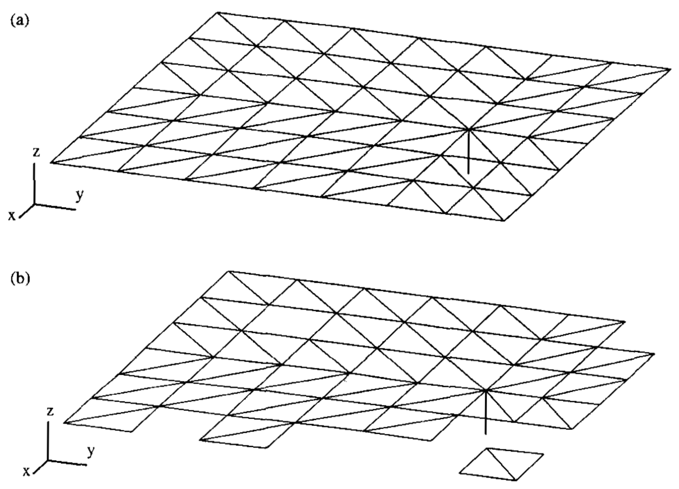


Figure 22. A patch antenna with a simple wire feed above an infinite ground plane before (a) and after (b) GA optimization.

plane. The feed consisted of a wire with a voltage source at the ground plane/wire interface, and located 0.24 × 0.12 cm from the corner of the patch. GA optimization was performed by removing square metal from the patch region. The population size was 100 individuals, and 100 generations were produced. The probability of crossover was set at 0.7, while the probability of mutation was equal to 0.02. Elitism and roulette-wheel selection [6] were used. The goal was to minimize the maximum S11 magnitude at three frequencies: 2.7 GHz, 3 GHz, and 3.3 GHz. The fitness function in this case was given by Equation (8):

$$fitness = \max_{\forall n}\left(-\mathrm{S11}_n \,\mathrm{db}\right), \qquad (8)$$

where the subscript n refers to sample points in the S11 (dB) versus frequency function. The GA optimizer here operated as a maximizer, with the goal of producing a large negative S11 magnitude at each of the sampled frequency points. By defining the fitness function as the return loss of the least-well-matched frequency point, the GA optimization had the affect of producing return loss values at all sampled points that were about the same magnitude.

The results of the best individual from the GA/MoM optimization, in terms of the magnitude of S11, are shown in Figure 21 as a solid line. As can be seen, an acceptable 2:1 match has been achieved over the entire 2.7-3.3 GHz band. The resulting patch is not symmetrical. This is not a surprise, however, since no constraints were placed on the GA optimizer to produce a symmetrical structure. Additionally, there are several entirely equivalent structures with identical S11 performance. While not an intuitive result, the design result of Figure22 is readily realizable by standard printed-circuit-board techniques.

7. Conclusion

Genetic algorithms have been introduced as another tool in the arsenal of the electromagnetic designer. A simple genetic algorithm was presented, and the operation of the principal GA operators, selection, crossover, and mutation, was discussed. A step-by-step case study was then presented, in an attempt to put the discussion of the genetic algorithm into a specific context. It was shown that, at least for a simple multi-dimensional problem, the steady-state GA has significantly better convergence performance than the generational GA.

Five representative examples of the use of the genetic algorithm, applied to electromagnetics, were then presented. Though only five specific applications were presented, the GA is applicable to a broad class of optimization problems. In fact, the five examples presented were chosen with the intent of demonstrating this idea of broad applicability.

While perhaps not always the best method for attacking an optimization, genetic algorithms have the distinct advantage that they tend to produce globally optimal results without requiring a great deal of information about the solution domain. Genetic algorithms are particularly useful when applied to problems that are highly constrained, or have discretized parameter sets that cause problems for the local techniques. A general-purpose genetic-algorithm engine can be developed that will apply to almost all problems, requiring only modifications to the fitness function. Genetic algorithms should be among the first methods considered when faced with a new electromagnetic optimization problem.

8. Acknowledgments

This work was supported in part by DARPA Contract JFBI94-222/J4C942220.

9. References

1. D-W Duan and Y. Rahmat-Samii, "A Generalized Diffraction Synthesis Technique for High Performance Reflector Antennas," *IEEE Transactions on Antennas and Propagation*, **AP-43**, 1, January 1995, pp. 27-40.

2. P. Ilavarasan, E. J. Rothwell, K. Chen, and D. P. Nyquist, "Natural Resonance Extraction from Multiple Data Sets Using Genetic Algorithm," *IEEE Transactions on Antennas and Propagation*, **AP-43**, 8, August 1995, pp. 900-904.

3. E. Michielssen, J. Sajer, S. Ranjithan, and R. Mittra, "Design of Lightweight, Broad-Band Microwave Absorbers Using Genetic Algorithms," *IEEE Transactions on Microwave Theory and Techniques*, **MTT-41**, 6/7, June/July 1993, pp. 1024-1031.

4. R. L. Haupt, "An Introduction to Genetic Algorithms for Electromagnetics," *IEEE Antennas and Propagation Magazine*, **37**, 2, April 1995, pp. 7-15.

5. J. M. Johnson and Y. Rahmat-Samii, "Genetic Algorithm Optimization and its Application to Antenna Design," IEEE Antennas and Propagation Society International Symposium *Digest*, June 19-24, 1994, pp. 326-329.

6. D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, New York, Addison-Wesley, 1989.

7. J. H. Holland, *Adaptation in Natural and Artificial Systems*, Ann Arbor, The University of Michigan Press, 1975.

8. K. A. De Jong, "An Analysis of the Behavior of a Class of Genetic Adaptive Systems," Doctoral Dissertation, University of Michigan (*Dissertation Abstracts International*, **36**, 10, 5140B, University Microfilms No. 76-9381, 1975.

9. D. E. Goldberg, and K. Deb, "A Comparative Analysis of Selection Schemes Used in Genetic Algorithms," *Foundations of Genetic Algorithms*, Morgan Kaufmann, 1991, pp. 69-93.

10. G. Sywerda, "Uniform Crossover in Genetic Algorithms," in J. D. Schaffer (ed.), *Proceedings of the Third International Conference on Genetic Algorithms and their Applications*, Morgan Kaufmann, June 1989, paper 2-9.

11. R. L. Haupt and S. E. Haupt, "Continuous Parameter vs. Binary Genetic Algorithms," *1997 Applied Computational Electromagnetics Society Symposium Proceedings, Volume II*, Monterey, CA, March 17-21, 1997, pp. 1387-1392.

12. D. Wiele, E. Michielssen, and A. Boag, "Community-Based Evolutionary Optimization of Frequency Selective Surfaces," IEEE Antennas and Propagation Society International Symposium and USNC/URSI Radio Science Meeting *URSI Abstracts*, Baltimore, MD, July 1996, p. 345.

13. R. L Haupt, "Thinned Arrays using Genetic Algorithms," *IEEE Transactions on Antennas and Propagation*, **AP-42**, 7, July 1994, pp. 993-999.

14. J. M. Johnson and Y. Rahmat-Samii, "Genetic Algorithm Optimization for Aerospace Electromagnetics," *IEEE Aerospace Applications Conference Proceedings Volume 1* (Snowmass at Aspen, CO), 1996, pp. 87-102.

15. J. M. Johnson and Y. Rahmat-Samii, "Genetic Algorithms in Electromagnetics," IEEE Antennas and Propagation Society International Symposium *Digest, Volume 2*, Baltimore, MD, July 21-26, 1996, pp. 1480-1483.

16. A. Boag, A. Boag, E. Michielssen, and R. Mittra, "Design of Electrically Loaded Wire Antennas using Genetic Algorithms," *IEEE Transactions on Antennas and Propagation*, **AP-44**, 5, May 1996, pp. 687-695.

17. D. S. Linden and E. E. Altshuler, "Automating Wire Antenna Design Using Genetic Algorithms," *Microwave Journal*, **39**, March 1996, pp. 74-86.

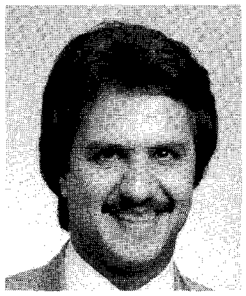
18. E. Michielssen, J. M. Sajer, and R. Mittra, "Design of Multi-layered FSS and Waveguide Filters using Genetic Algorithms," IEEE Antennas and Propagation Society International Symposium *Digest*, Ann Arbor, MI, 28 June-2 July 1993, pp. 1936-1939.

19. J. M. Johnson and Y. Rahmat-Samii, "Genetic Algorithms Optimization of Wireless Communication Networks," IEEE Antennas and Propagation Society International Symposium *Digest*, June 18-23, 1995, pp. 1964-1967.

20. J. M. Johnson and Y. Rahmat-Samii, "A Novel Integration of Genetic Algorithms and Method of Moments (GA/MoM) for Antenna Design," *1997 Applied Computational Electromagnetics Society Symposium Proceedings, Volume II*, Monterey, CA, March 17-21, 1997, pp. 1374-1381.

Introducing Feature Article Authors

J. Michael Johnson


Yahya Rahmat-Samii

J. Michael Johnson received the BS and MSE degrees from the University of California, Irvine, in 1980 and 1983, respectively.

He has worked for Hughes Aircraft Company in advanced hybrid microelectronics, Lockheed Missiles and Space as a satellite receiver designer, and for Deskin Research Group. Most recently, he was employed by Condor Systems as a microwave design engineer and systems engineer. In 1991, he returned to academia, where he is currently pursuing his PhD degree at the University of California, Los Angeles. At UCLA, he is specializing in computational electromagnetics, with an emphasis in optimization techniques (GA) and finite-difference time-domain methods (FDTD).

Mr. Johnson is a registered US patent agent. He is the founder and president of North Shore Associates, founded in 1994. North Shore Associates provides electrical engineering consulting and patent preparation/prosecution services.

Yahya Rahmat-Samii is a Professor of Electrical Engineering at the University of California, Los Angeles (UCLA). He was a Senior Research Scientist at NASA's Jet Propulsion Laboratory/California Institute of Technology, before joining UCLA. He was a Guest Professor at the Technical University of Denmark (TUD) in the summer of 1986. He has also been a consultant to many aerospace companies. He received the MS and PhD degrees in Electrical Engineering from the University of Illinois, Urbana-Champaign.

Dr. Rahmat-Samii was the 1995 President of the IEEE Antennas and Propagation Society. He was appointed an IEEE Antennas and Propagation Society Distinguished Lecturer, and presented lectures internationally. Dr. Rahmat-Samii was elected a Fellow of the IEEE in 1985, and a Fellow of the IAE in 1986. He has been the guest and plenary session speaker at many national and international symposia. He was one of the Directors and Vice President of the Antenna Measurement Techniques Association (AMTA) for three years. He has been Editor and guest Editor of many technical journals and book publication entities. Dr. Rahmat-Samii was also a member of UCLA's Graduate Council.

Dr. Rahmat-Samii has authored and co-authored over 380 technical-journal articles and conference papers, and has written thirteen book chapters. He is the co-author of the book *Impedance Boundary Conditions in Electromagnetic*, published in 1995. He is also the holder of several patents. His research contributions cover a diverse area of electromagnetics, antennas, measurement and diagnostics techniques, numerical and asymptotic methods, and satellite and personal communications. For his contributions, Dr. Rahmat-Samii has received numerous NASA and JPL Certificates of Recognition. In 1984, he was the recipient of the Henry Booker Fellowship of URSI. In 1992 and 1995, he was the recipient of the Harold A. Wheeler Applications Prize Paper Award for papers published in the 1991 and 1993 IEEE AP *Transactions*. In 1993, 1994, and 1995, three of his PhD students were named the Most Outstanding PhD Students at UCLA's School of Engineering and Applied Science, and another received the first-place Student Paper Award at the 1993 IEEE AP-S Symposium. Dr. Rahmat-Samii is a member of Commissions A, B, and J of USNC/URSI, AMTA, Sigma Xi, Eta Kappa Nu, and the Electromagnetics Academy. He is listed in Who's Who in America, Who's Who in Frontiers of Science and Technology, and Who's Who in Engineering.