Control of a Multi-Beam Antenna

Using a Genetic Algorithm

by

Kevin Everett Gilpin
B.S. Massachusetts Institute of Technology (1994)

Submitted to the Department of Aeronautics
and Astronautics in Partial Fulfillment of the
Requirements for the Degree of

Master of Science

at the

Massachusetts Institute of Technology
May 1995

Signature of Author.....
Department of Aeronautics and Astronautics
May 18, 1995

Certified by.....
Professor Wallace E. Vander Velde
Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by .....
Professor Harold Y. Wachman
Professor of Aeronautics and Astronautics
Chairman, Department Graduate Committee

Control of a Multi-Beam Antenna Using a Genetic Algorithm

by

Kevin Everett Gilpin

Submitted to the Department of Aeronautics & Astronautics
in partial fulfillment of the requirements for the degree of
Master of Science in Aeronautics & Astronautics at the
Massachusetts Institute of Technology

Abstract

The control of an area coverage multi-beam antenna in the presence of interference is a real-time problem which requires an quick solution, with the possibility of further refinement if time permits. The objective is to maximize some objective function related to antenna gain over as much of the service area as possible; this problem exhibits high dimensionalilty, along with a complicated functional relationship. Previously, a simulated annealing algorithm had been used to optimize the performance of one such antenna, the MILSATCOM satellite main antenna. This thesis investigates the use of a genetic algorithm as means of solving this difficult optimization problem, and examines the performance of the algorithm as a function of its control parameters.

Thesis leader: Prof. Wallace E. Vander Velde
Title: Professor of Aeronautics & Astronautics
Massachusetts Institute of Technology

Acknowledgments

I would like to thank Prof. Vander Velde for his guidance and support, and for his confidence in me even when I was an undergraduate.

Thanks to Jay Simon for the use of his work on S-PACE, a paper which proved to be a valuable reference.

To my friends; too many to mention, but most can be summed up as "5-man", and to a broader extent, ΣΦE and friends thereof.

To the fam, who never pushed me any way but up, and incidentally footed most of the bill.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The desirable characteristics of an area coverage satellite are fundamentally different from those of one providing single user to user service. The objective is to maintain communications over as much of a designated service area as possible; this goal is complicated by active jamming from the ground, as well as the fact that the locations and intensities of the users' signals are sporadic and unknown. The problem of providing optimal service to a single user in the presence of interference is traditionally solved using the Howells-Applebaum (A-H) algorithm, which is intended to maximize the signal-to-interference-plus-noise (SINR) ratio of the array. While a similar kind of criterion may be applied to the area coverage problem, the solution must be fundamentally different because the user is, in effect, located everywhere within the coverage area. A-H relies on the specification of a steering vector, which gives the location of the user explicitly; while it is possible to think of the solution to the area coverage problem in terms of some optimal steering vector, it is precisely this vector which it is the objective of the algorithm to find.

The area coverage problem is therefore essentially reduced to one of a multi-dimensional search, following a few constraints, over a complicated functional surface. Given that a problem is in some way well-behaved (a condition which will be examined later in more detail), genetic algorithms have often proven to provide fast, near-optimal solutions. The genetic algorithm is based on an analogy to biological evolution, in which solutions which are successively more and more optimal are derived from a population of other potential solutions; the algorithm begins with a population of random guesses. The genetic algorithm is well suited to optimization problems such as this one examined in this thesis because:

- Genetic algorithms are not a gradient-based search procedure, and thus are far less likely to become trapped in local minima. The functional space of the area-coverage problem is essentially formed from the sums of powers of sinusoids, a functionality which exhibits a large number of such local minima.

- Many forms of constraints are very easy to implement. This property is used extensively in this problem, as the knowledge of the presence of the jammers provides some useful information about the parameter space.

- While being very computationally intensive, genetic algorithms are also highly parallelizable. The evaluation of the performance of each possible solution may be performed independently of all the rest; the only step which requires knowledge of the rest of the population is the process of forming a new generation.

The goal of this thesis is to explore in detail the suitability of a genetic algorithm to the solution of the area coverage problem. Some aspects of this problem, and multi-beam antennas in general, are examined in order to formulate the problem in as efficient a manner as possible. Since the genetic algorithm is essentially stochastic, a statistical investigation of the performance of the algorithm as a function of its control parameters is also performed.

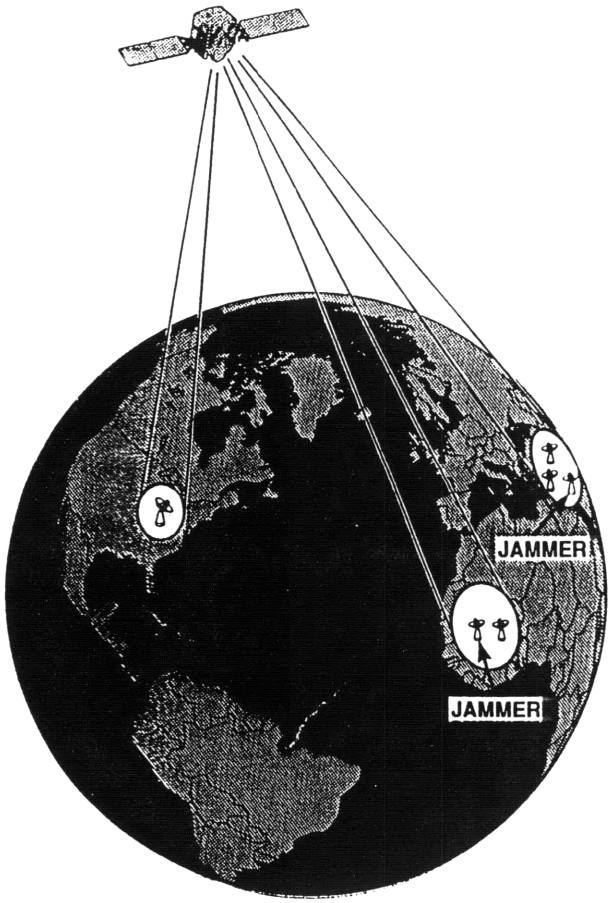
Chapter 2

Problem Formulation

2.1 The MILSATCOM Satellite

The MILSATCOM satellite carries a multi-beam antenna located in geostationary orbit. Some of its characteristics are shown in Table 2.1, and a diagram of its mission is in Figure 2.1.1.

Figure 2.1.1; The MILSATCOM area-coverage satellite



Its mission is to provide coverage to a designated area, by serving as a relay between users on the ground. It does not discriminate between desirable signals and noise, or attempt to track its users. Its antenna consists of 128 isotropic elements, which are electronically grouped into 7 non-isotropic beams. The non-isotropy of the beams may be modeled using an analytical function, typically

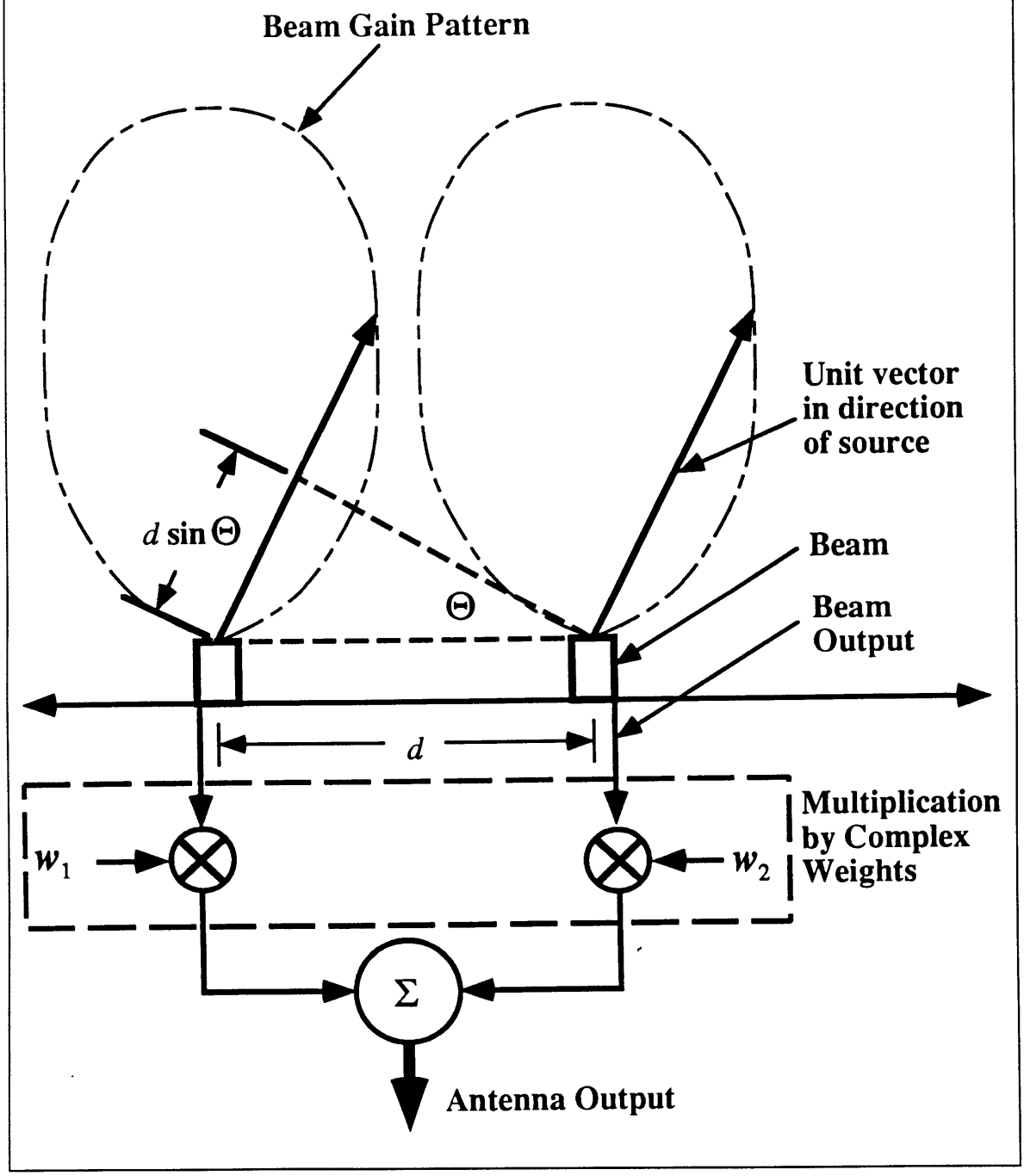
Diameter (in)	40
Operating Frequency (GHz)	8.5
Number of beams	7
Service Area Radius (Degrees from vertical)	2.6

Table 2.1: Antenna characteristics

related to the Bessel family of functions. The sum of the complex outputs of these beams determines the gain pattern of the antenna over the target area, and thus how well the satellite is performing its mission. Ignoring non-idealities, the output of each beam is identical to the output of all the others except for a phase shift which results from the geometry of the antenna, and a modulation which is a result of the non-isotropic gain pattern. In order to adjust the gain pattern of the antenna, the output of each beam is multiplied by a complex weight factor which imposes a phase shift and modulation on the signal which is determined by the real and imaginary parts of this weight factor. After these weights have been applied, the outputs of the beams are summed up in the complex plane. The magnitude of this sum at each point in space is equivalent to the antenna gain at that point. An overview of the operation of the antenna is provided in Figure 2.1.2.

The satellite simply rebroadcasts the signals which it receives. A spread spectrum communications protocol is used to provide some degree of immunity to jamming, but this countermeasure may not be sufficient in all cases. It is to provide an additional means of response to this interference that the complex weight factors are introduced. The response of the antenna to the presence of interference is manipulation of its gain pattern in such a way as to place a null (region of diminished gain) directly on top of the jammer's location. A jamming signal may be identified and located through a series of operations on the matrix formed from the covariances of each of the beam outputs, and in this way the interference may be effectively silenced. However, since the mission of the satellite is to maintain communications as well as possible over the target area, this nulling procedure is only part of the problem. What must also be done is to optimize the service to the ground users, by making the nulled region(s) as small as possible and by restoring as much gain as

Figure 2.1.2: Multi-beam antenna operation



possible to the rest of the target area. This optimization problem may be formulated in many different ways, two of which are

- Maximize the area over which the gain is above some threshold value

- Maximize some measure of signal to interference plus noise ratio

Both of these conditions were used in these experiments.

2.2 The Jamming Environment

While it is possible to run the genetic algorithm without any a priori knowledge about the jammers at all, it is more efficient to try to learn as much about the jamming environment as possible beforehand. One way in which to visualize how this may be done, although it is not possible in practice, is to write down algebraic constraints on the weights for each jammer. The output of each beam is determined by three factors: the value of the gain pattern of the antenna $G(\mathbf{x})$, the real and imaginary weights w_R and iw_I, and the real and imaginary components of the phase shift δ of the signal, relative to some arbitrary reference point (usually taken to be the center of the antenna). The (complex) output of each beam is given by:

$$G(\mathbf{x})\left[\left(w_R \cos\delta - w_I \sin\delta\right) + i\left(w_R \sin\delta + w_I \cos\delta\right)\right] \tag{2.2.1}$$

If a jamming signal is to be completely nulled, then both the real and imaginary terms in the total (summed) gain must be equal to zero in the direction of that jammer. For N jammers and M beams, writing these constraints gives $2N$ constraints in $2M$ variables (2 weight factors for each beam). For each jammer:

$$\sum_{j=1}^{M} G(\mathbf{x}_j)\left[\left(w_{Rj} \cos\delta_j - w_{Ij} \sin\delta_j\right) + i\left(w_{Rj} \sin\delta_j + w_{Ij} \cos\delta_j\right)\right] = 0 \tag{2.2.2}$$

This group of constraints may be more succinctly written as the matrix equation

$$\mathbf{Aw} = \mathbf{0} \tag{2.2.3}$$

where

$$\mathbf{A} = \begin{bmatrix} G(\mathbf{x}_{11})\cos\delta_{11} & -G(\mathbf{x}_{11})\sin\delta_{11} & G(\mathbf{x}_{12})\cos\delta_{12} & \cdots \\ G(\mathbf{x}_{11})\sin\delta_{11} & G(\mathbf{x}_{11})\cos\delta_{11} & G(\mathbf{x}_{12})\sin\delta_{12} & \\ G(\mathbf{x}_{21})\cos\delta_{21} & -G(\mathbf{x}_{21})\sin\delta_{21} & G(\mathbf{x}_{22})\cos\delta_{22} & \\ G(\mathbf{x}_{21})\sin\delta_{21} & G(\mathbf{x}_{21})\cos\delta_{21} & G(\mathbf{x}_{22})\sin\delta_{22} & \\ & \vdots & & \ddots \end{bmatrix}, \mathbf{w} = \begin{bmatrix} w_{R1} \\ w_{I1} \\ w_{R2} \\ w_{I2} \\ \end{bmatrix}$$

For the usual case where $N < M$, the solution is underconstrained; in fact, any set of weights which lie in the nullspace of the matrix **A** will completely null the jamming. The effect of the jammers, then, is to reduce the number of degrees of freedom that the antenna has by $2N$.

In practice, the directions to the jammers are not known, nor is it even theoretically possible to determine them if there is more than one. It is possible, however, to find the nullspace of **A** indirectly. The relevant information is taken from the matrix Φ which is constructed by finding the covariances of all of the unweighted beam output vectors $\mathbf{J}_k$ according to the formula:

$$\Phi = \sum_{k=1}^{N} P_k \mathbf{J}_k \mathbf{J}_k^{T*} + \sigma^2 \mathbf{I} \tag{2.2.4}$$

where σ is the sum of the background and antenna noise, and the P_k are the jammer powers. The superscript ^{T*} denotes the transpose conjugate. It is assumed here that none of the jammer signals are correlated, i.e. there is no persistent interference between them. The justification for this is that such a correlation would require that two or more signals would have to keep a consistent phase relationship over a fairly long time period, at a distance of thousands of kilometers and at a wavelength of less than 5cm. It is also assumed that none of the signals are correlated with the noise, and that the users are sufficiently weak that their contribution to the covariance may be ignored.

An alternative representation of Φ is in terms of its eigenvalue decomposition

$$\Phi = \mathbf{E}_s \Lambda_s \mathbf{E}_s^{T*} + \sigma^2 \mathbf{E}_n \mathbf{E}_n^{T*} \tag{2.2.5}$$

In this case, the vectors $\mathbf{E}_s$ correspond to the "signal" eigenvectors of Φ, and are associated with the eigenvalues Λ_s, while the vectors $\mathbf{E}_n$ correspond to the "noise" eigenvectors and are associated with the noise eigenvalues σ^2. There are several

characteristics of Φ which are useful in it's analysis. It is Hermitian, which implies that all of its eigenvalues are real, and that its eigenvectors are orthonormal. As a consequence of this orthonormality, if we form the product $\Phi\mathbf{E}_n$ the result is $\sigma^2\mathbf{E}_n$; this implies that all of the $\mathbf{E}_n$ are orthogonal to the $\mathbf{J}_k$. Therefore, the space spanned by the $\mathbf{E}_s$ is the same as that spanned by the $\mathbf{J}_k$. The eigenvalue λ_k corresponds to the antenna output if the eigenvector $\mathbf{E}_{sk}$ is used as the weight vector, while the σ^2 eigenvalues correspond to the thermal noise power. Therefore, any weight vector which is purely a linear combination of the $\mathbf{E}_n$ will null the jamming completely, and may be interpreted as lying in the nullspace of the matrix $\mathbf{A}$.

By constructing this equation using a singular value decomposition (SVD), the signal eigenvectors (due to interference) may be picked out as the ones which correspond to the "significant" singular values. The term "significant" may be difficult to interpret if the power of the jammers relative to the noise is not known; Jay Simon, in his study of the S-PACE algorithm, investigated this matter. In this work, the jammer powers were given to the program explicitly, as this matter of picking out significant terms is not directly relevant to the operation of the genetic algorithm. Once the set of $\mathbf{E}_n$ has been found, they may be interpreted as covering some subspace of the $2M$-dimensional space in which the weight vectors may lie; call this subspace Ω_n. Any linear combination of these $2(M\text{-}N)$-dimensional eigenvectors will lie in Ω_n, and thus will null all of the interference. In this manner, the dimensionality of the search is reduced by two each time an additional jammer is encountered.

A weight vector which lies in Ω_n, while it nulls the jamming completely, does not maximize the SINR except for in the case of infinitely powerful jammers. The solution to the problem of finding the maximum SINR given a steady-state (quiescent) weight vector and a set of jammers may be formulated according to a "Weiner solution", for which the optimum weight vector $\mathbf{w}$ is given by

$$\mathbf{w} = \mu\Phi^{-1}\mathbf{v} \tag{2.2.6}$$

where μ is an arbitrary constant (which can be assumed to be 1), and $\mathbf{v}$ is the quiescent weight vector. In situations for which the Weiner solution was originally intended, $\mathbf{v}$ was

determined by the location of the signal which the antenna was attempting to follow. In the area-coverage problem, there is no such reference. For a traditional quiescent weight vector, the effect of the multiplication by Φ^{-1} is to place partial nulls on the jammers which are exactly deep enough to maximize the SINR; for weak jammers they are shallow, while for strong jammers they are deep. In fact, for infinitely strong jammers, the nulls are absolute. This is equivalent to making the weight vectors lie in the jammer nullspace.

The Howells-Applebaum algorithm is an iterative method of converging to the Wiener solution. In the S-PACE paper it is implemented as a follow-on stage which takes the quiescent vector **v**, which lies in $\Omega_{n,,}$ and adds the appropriate signal-space component to it such that the Wiener criterion is satisfied. The approach that was taken in this research was somewhat different. Since all of the jammers which the genetic algorithm encountered in these experiments were at least 2 orders of magnitude greater than the noise, the follow-on stage was omitted. Essentially, this omission is equivalent to the assumption that the signal space component of any reasonable solution vector **w** would be insignificant compared to the noise space component. While this assumption may not be entirely justified, to implement the A-H algorithm would have significantly increased the computation time and complexity, and therefore limited the extent to which the genetic algorithm itself could be investigated.

There is another important constraint which may be applied to the optimization process which involves the phase shifts imposed by the complex weight factors. If every weight factor were rotated in the complex plane through some angle ϕ, then the result would be to rotate the summed gain by the same angle ϕ everywhere in space. Since the gain of the antenna depends only on the magnitude of the sum, and not its phase, the gain pattern of the antenna would be unchanged. Therefore it is possible to constrain the weights so that they have some constant phase relationship. This was implemented in the algorithm by requiring that the last complex weight always have an angle of $\pi/4$. Any other angle would have served equally well.

One final constraint concerns the magnitudes of the weights. If the components of the weights were allowed to take on any real value, then it is possible that the algorithm could become unstable as one or more of the weight factors increased towards + or $-\infty$.

Therefore, an additional constraint was imposed that the absolute value of the maximum weight must always be equal to 1. Changing the values of all the weights by the same factor does not alter the performance of the antenna since the weights scale the received signal and noise equally.

Chapter 3

The Genetic Algorithm

3.1 Introduction

The genetic algorithm is a stochastic search algorithm which is based on the principles of natural selection and evolution. In some older formulations, the idea was to simulate the evolutionary process as closely as possible, even to the point of introducing the concepts of habitat and predators in a mathematical sense [Hillis 5]. In their current form, genetic algorithms are designed to promote survival of the fittest in a stochastic but structured sense, passing information about good solutions from generation to generation through an encoded string. Genetic algorithms differ from most other optimization and search techniques in several ways:

- They do not use any information about the underlying structure of the parameter space in which they operate, including gradients

- They do not optimize on the search parameters directly; rather they use a coded set of parameters which makes the passing of information and the stochastic modification of solutions more efficient

- They are, in essence, completely stochastic. No effort is made to guide the solution in any direction, and the rules which govern the algorithm remain the same throughout the evolutionary process

- They search over a population of points at each iteration. This is fundamentally different from doing restarts of an poorly converged search, as the population is evolving in parallel

The search operation of a genetic algorithm is fundamentally different from almost any other search technique; its closest relative is probably the pure random search. Rather than using gradient information to determine a search direction, genetic algorithms rely on the exchange of encoded parameters to find better and better solutions. This information must be encoded in a sensible manner into finite strings (of binary digits in most cases), whose elements are related in some direct manner to the relevant parameters.

Genetic algorithms derive much of their robustness from this unique fashion of information representation. Each encoded string contains all the information necessary to determine the "fitness", or nearness to an optimal solution, of the population member which possesses it, and by passing portions of this information on to members of the next generation their fitness may be improved. In each successive generation, some of the initial diversity which was present in the initial population may be lost; however, this will usually be a result of discarding poor strings or sections of strings, and has the effect of directing the search in the right direction.

The rules of genetic algorithms are entirely stochastic, but this does not lead to the conclusion that the search is simply random. The search is directed by the fact that more fit members are more likely to be involved in the evolutionary process, while less fit members are more likely to be discarded. In this fashion, the solution does not proceed directly towards the most obvious solution, nor does it wander blindly.

Section 3.2 describes the operation of the genetic algorithm in detail, while their mathematical foundation, the *fundamental theorem of genetic algorithms*, is described in Section 3.3.

3.2 Components of the Genetic Algorithm

In close keeping with their original formulation, the operators and terminology employed in genetic algorithms correspond directly to the processes of natural evolution (see Table 3.1). The operations are performed on strings which merely encode the actual characteristics (or phenotypes) of the population members. The initial characteristics are chosen at random, so as to provide maximum diversity and minimum bias. Successive generations are produced through the action of three operators:

- selection & reproduction
- crossover
- mutation

While there are many implementation-specific details of these operators which may vary from application to application, they are fundamentally very simple.

Biological	
chromosome	**substring**
gene	**feature**
allele	**feature value**
locus	**string position**
genotype	**string**
phenotype	**parameter set**

Table 3.1: Biological and mathematical terms

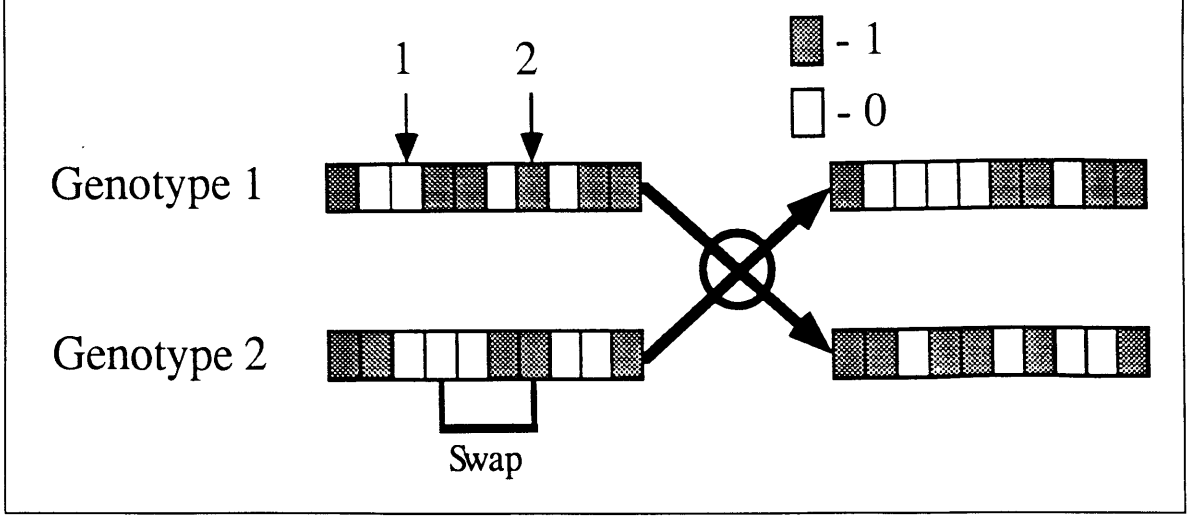
The primary driving force behind evolution is that of natural selection. Members of a population which have a high survivability are able to propagate their traits into future generations, while traits belonging to members of low fitness are generally allowed to die out. The analog of survivability in genetic algorithms is fitness; the fitness of a genotype is

determined by evaluation of the fitness function using the phenotypes of that genotype, which is a direct measurement of its survivability relative to the rest of the population. The larger the fitness, the better chance the genotype has of being selected for the "mating pool," which will serve as the foundation for the next generation. The exact manner in which these fitnesses are evaluated and converted into a mating pool will be examined in detail in Section 4.2.

When two organisms reproduce, the information stored in their chromosomes is combined to produce a new, unique, individual. This process, both in nature and in the genetic algorithm, is accomplished primarily through the crossover operator. This operator essentially exchanges segments of the genotype from a pair of donors to form 2 new, distinct genotypes. One such exchange is pictured in Figure 3.2.1, in which 2 genotypes exchange the genes between 2 random loci. This process may be carried out any number of times, but in this thesis only a single crossover operation was performed, with crossover points chosen randomly from among all the loci.

The mutation operator serves a unique role: to introduce diversity and random exploration in a way that crossover cannot do. There are two main reasons why this is useful. The first is that it is possible that during the operation of the algorithm, a useful piece of genetic code may be lost. The second possibility is that all of the random initial genotypes may have the same allele in one particular locus. In this case, despite the

Figure 3.2.1: The crossover operator



crossover operator, all subsequent genotypes will also have this same allele. The mutation operator essentially picks a random locus, and arbitrarily changes its allele. In this manner, new genetic code may be introduced, and gaps in the diversity of the population may be filled.

Both crossover and mutation are assigned a certain probability of occurrence. These probabilities are 2 of the most important parameters of any genetic algorithm, and it is difficult to see how their optimal values could be determined in advance. While there is some research into this question, most genetic algorithms still rely on some sort of Monte Carlo simulation to determine their optimal values. Typical values of crossover and mutation probabilities are on the order of 80% and 1%, respectively. The mutation probability is especially sensitive; too high a value, and the population will be mutated into a near-random search, too low a value, and it will be difficult to improve on the best solution obtained so far due to lack of diversity. Examples of these cases will be given in Section 4.3.

3.3 The Fundamental Theorem of Genetic Algorithms

3.3.1 Schemata

There is a convincing mathematical demonstration of the power of genetic algorithms, called the *Fundamental Theorem of Genetic Algorithms*. One tool which is used in the proof of this theorem is the concept of *schemata*, which are used to describe classes of genotypes. A genotype consists of a string of symbols, usually bits. To this representation is added an additional character, the *. When comparing two genotypes, the * character serves as a wild card, matching either a 1 or a 0. A genotype in the genetic algorithm may be envisioned as being a sample of a large number of schemata, those schemata which match the genotype in all locations. For binary genotypes which are 4 bits in length, some schemata and their matching bit strings are shown in Table 3.2

	Schemata			
	110*	***11****	***1*00***	****00***
Matching	1100	1100	1000	0000
Bit Strings	1101	1101	1100	0100
		1110		1000
		1111		1100

Table 3.2: Examples of schemata

Notice that 1100 is an example of all 4 of these schemata. Each binary genotype of length n matches

$$\sum_{k=0}^{n}\binom{n}{k} = \sum_{k=0}^{n}\frac{n!}{(n-k)!k!} = 2^n \tag{3.3.1}$$

schemata, ranging from the most specific (that schema which matches the genotype exactly) to the most general (the schema consisting of all *s). There are a total of 3^n binary schemata, of which a population of N genotypes may then match up to almost $N2^n$, neglecting the fact that there must be some duplication among schemata having a large number of * entries, and dependent upon the diversity of the population.

3.3.2 The Fundamental Theorem

The genetic operators may be envisioned as being applied to schemata rather than to genotypes, as they can be treated as variables which represent the actual strings that they match. One can envision that the desired effect of the application of the reproductive, crossover, and mutation operators is to allow strings which have desirable properties to flourish; however, it is difficult to describe exactly what is meant by a "desirable property." The way that schemata help with this problem is that by including the * values, a string which contains some valuable information may be associated with a particular schema. If it can be shown that the prevalence of this schema in the population will grow, then the valuable code in the underlying strings may be seen to be responsible.

The property of a genotype which results in its selection for the mating pool is, of course, its fitness. While a schema does not actually have a fitness per se, its fitness may be interpreted as being the average fitness of those strings which it matches. For an unbiased reproduction operator, the probability that a string with fitness F_i will be placed in the mating pool is determined by the percentage of the total population's fitness which that string is responsible for:

$$p_i = \frac{F_i}{\sum_{j=1}^{N} F_j} \tag{3.3.2}$$

and the expected value of the number of copies of that string is Np_i. This leads to the first step in the fundamental theorem, namely that the expected number of examples of schema H present in the $n+1$st mating pool is proportional to

- The number of examples of that schema in the nth generation: $m(H,n)$
- The population size N
- The probability $P(H)$ that that schema will be selected, as determined by the average of the p_i of the genotypes which match it

In equation form, the expected value of the number of examples of schema H in generation $n+1$ is

$$E[m(H,n+1)] = Nm(H,n)P(H) \tag{3.3.3}$$

Note that if the product $NP(H)$ is greater than one, then the schema may be expected to grow exponentially, while if less than one it will decay exponentially. By noting that the average fitness of the entire population is

$$F_{ave} = \frac{1}{N}\sum_{j=1}^{N} F_j \tag{3.3.4}$$

the growth equation may be written in a form independent of the population size N as

$$E[m(H,n+1)] = m(H,n)\frac{f(H)}{F_{ave}} \tag{3.3.5}$$

where $f(H)$ is the average fitness of the strings representing schema H. This equation is known as the *reproductive schema growth equation*.

Of course, this result shows only the effects of one operator; there are two others as well. Here it is necessary to introduce some terminology about schemata. The first definition is the schema order; the order $o(H)$ of a schema is the number of bits which it contains (as opposed to *s). Secondly, the defining length $\delta(H)$ is the maximum distance between its bits. So for example, the schema **11*10**1* has order 5 and defining length 7. For a schema with $o(H)$=1, $\delta(H) \equiv 0$. Once these terms are defined, the crossover and mutation operators may be taken into account.

The effect of crossover is to transfer information from one schema to another. In the process, it may destroy the schema which it was operating on. For the purposes of establishing a lower bound, it is assumed that any time the (single-point) crossover operator acts within the defining length of a schema, that schema is effectively destroyed. While some researchers have looked into the effects of relaxing this assumption [*], it is not necessary to do so to obtain a lower bound on the schemata growth rate. Consider then that the probability that a string of overall length l will be destroyed by crossover is

$$p_c \frac{\delta(H)}{l-1} \tag{3.3.6}$$

if crossover occurs with probability p_c and is uniformly distributed over the length of the string. Likewise, mutation also has the potential to destroy schemata. In this case, the assumption is that a mutation in any of the bits (0's and 1's) of the schema will destroy it. Therefore, the expected value of the number of harmful mutations per schema is

$$p_m o(H) \tag{3.3.7}$$

Combining these results, a lower bound on the probability of surviving crossover and mutation unscathed is

$$1 - p_c \frac{\delta(H)}{l-1} - p_m o(H) \tag{3.3.8}$$

Taking these operators into account, the probability that a schema will be placed in the mating pool is now the product of the probability that it will be selected and the probability that it will survive these operators. This may be easily incorporated into the reproductive schema growth equation, to give the central result of the fundamental theorem of genetic algorithms:

$$E[m(H,n+1)] \geq m(H,n)\frac{f(H)}{F_{ave}}\left(1 - p_c \frac{\delta(H)}{l-1} - p_m o(H)\right) \tag{3.3.9}$$

This result is fairly easily interpreted. The genetic algorithm will favor schemata which have high average fitness relative to the rest of the population, low order, and short defining length.

The requirements of low order and short defining length have important implications when building a genetic algorithm. In order for the algorithm to function properly, it must be possible to piece together good chromosomes to form a new genotype which combines the good qualities of its "parents". If such a breakdown is not possible, then the genetic algorithm degenerates into a pseudo-random search. An example is provided by Van Deventner[8], in which he attempts to encode the parameters of a Chebyshev polynomial into genotypes in order to optimize the trajectory of an aircraft subject to threats. The problem he encountered was that while Chebyshev polynomials provided a concise way of encoding many different trajectories, the effects of importing a coefficient from another genotype were extremely unpredictable. Changing the formulation of the problem to one which was longer but more simply represented resulted in dramatic improvement in the performance of the algorithm. A similar situation was encountered in this project in its early stages. Before the phase constraint was imposed, the performance of the algorithm had been fairly poor. It was then recognized that by not

constraining the phase, not only did the algorithm have to contend with an unnecessary degree of freedom but, more importantly, the parameters encoded in one genotype did not have any consistent relationship to those in the rest of the population. By imposing the phase constraint, the population members were, in effect, brought into alignment so that they were all working within the same framework. This modification improved the algorithm's performance significantly.

Chapter 4

The Genetic Algorithm Applied to Antenna Control

4.1 Problem Model

The problem naturally breaks itself down into several independent components: the service area, the antenna, and the jammers. Each of these components is modeled as a separate entity, with its characteristics determined by the parameters of the problem. The operation of the genetic algorithm then incorporates information from these three structures to evaluate the fitnesses of the population members.

4.1.1 The Antenna

The antenna is represented as an arbitrary number of beams, each located in 3-dimensional space relative to the origin (0, 0, 0). The beams all have the same gain pattern, which may be modeled by a variety of axisymmetric functions. In addition to physical location, each beam has a principal axis along which its gain pattern is directed. The directions of the principal axes for non-isotropic beams (isotropic beams have no principal axis) are determined by the *crossover gain*, which is the value of the gain function at which the beams overlap. For these experiments, the antenna gain pattern was modeled by the function $2J_1(x)/x$, which has a maximum value of 1, and multiple sidelobes which decrease in amplitude with distance (Figure 4.1.1). The dashed line represents the crossover gain. Certain properties of the antenna which affect the specifics of its operation, such as the operating frequency and dish diameter, are supplied as variable inputs. For a crossover gain of -5.6 and the above gain function, the beams are arrayed as shown in figure 4.1.2.

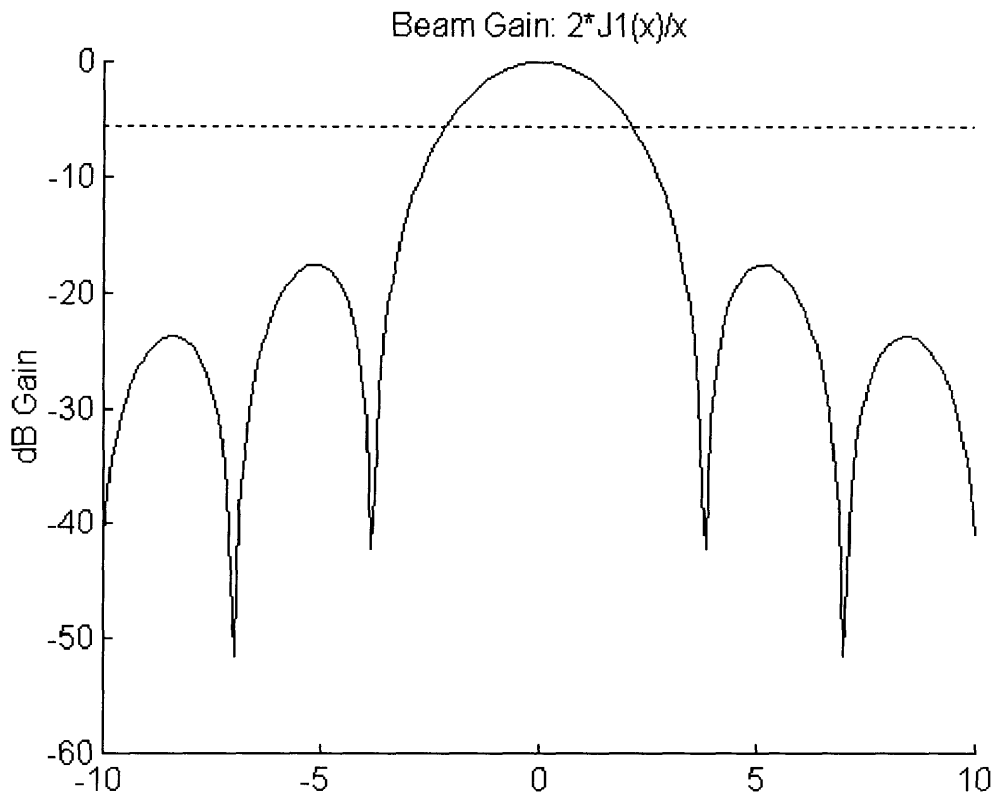


Figure 4.1.1: Beam gain pattern

4.1.2 The Service Area

The antenna is designed to provide coverage over a circular region of the earth, which is modeled in the experiment as a section of a sphere. The size of this target area is determined by the quiescent gain pattern of the antenna; for the trials done in this thesis it included all points whose angular distance from the vertical is less than 2.6 degrees. In order that the gain pattern of the antenna may be reliably estimated, the target area must be discretized into approximately equally shaped elements. This is done by using a variation of a tessellation procedure called the Delauney triangulation [Watson 7]. This algorithm, commonly used in computational fluid dynamics, takes as its input a field of points. It then constructs triangular elements from these points such that the minimum angle of all the elements' corners is maximized. The advantage of doing such a tessellation

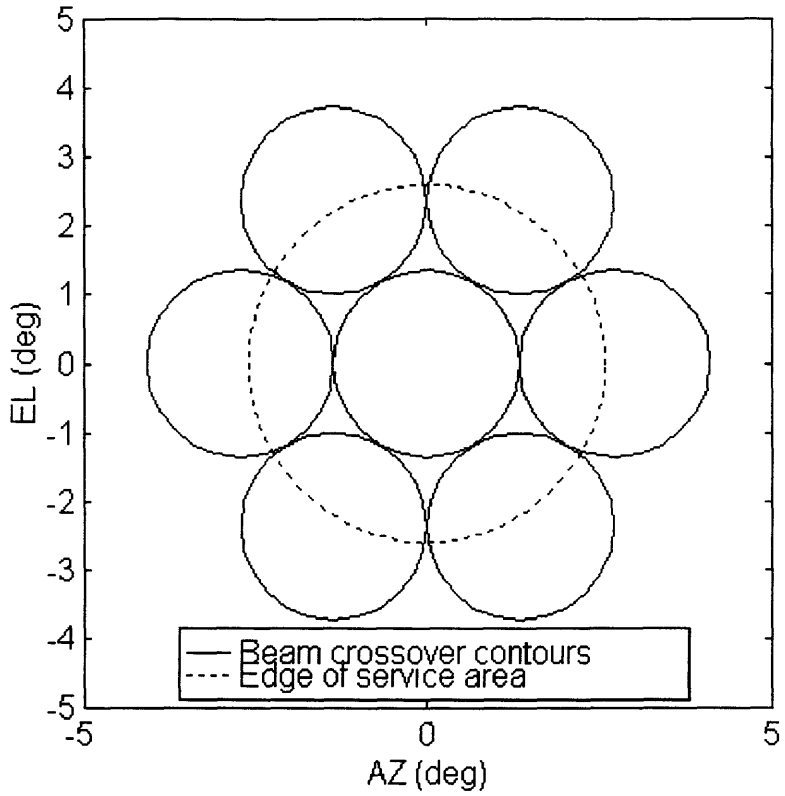


Figure 4.1.2: Beam crossover contours and the boundary of the service area

is that the mesh may be refined by adding additional points anywhere inside the domain, with the guarantee that the resulting elements will remain in some way regular. A common technique for adding additional elements is to add a new point at the centroid of an existing element, with the result that that element, and possibly some of its neighbors, is replaced by a larger number of smaller ones. This procedure is used in these experiments to refine the grid in regions at which high gradients in the gain pattern are expected: namely, the regions around the jammer locations.

Figures 4.1.3 and 4.1.4 illustrate the nature of the grid which was used in the program. The first one shows a 3-D view of a grid which encloses 30 degrees of elevation, in order to exaggerate the 3-dimensional nature of the tessellation. The second figure shows a plane view of the elements, with additional elements located in the vicinity of the jammer locations (indicated by circles). While there is a fully 3-dimensional version of the Delauney triangulation algorithm which constructs tetrahedra rather than triangles, it was

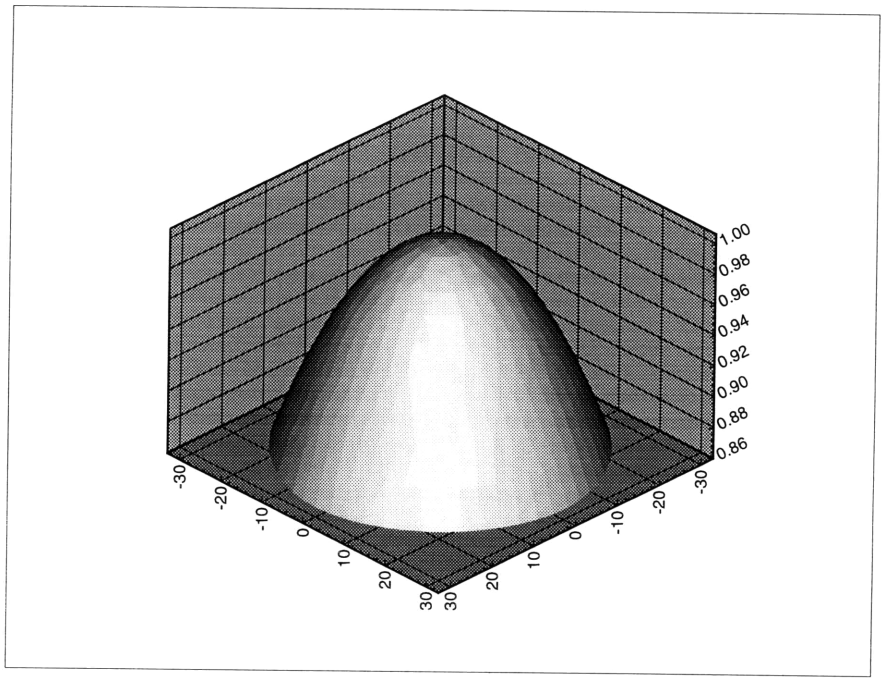


Figure 4.1.3: 3-D view of a large target mesh
Encompasses ±30° from the vertical

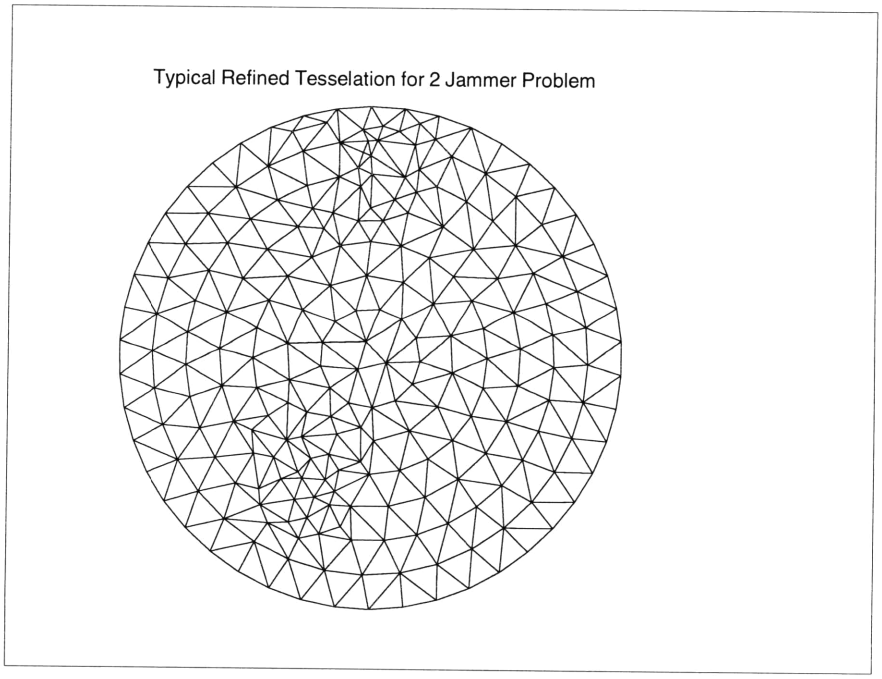


Figure 4.1.4

not necessary to use it in this project. Rather, the grid is constructed as if it were lying in the plane, and then the elements are projected onto the surface of a sphere. For the small azimuth angles used here, the target is essentially planar and the difference is negligible.

While there are several differing methods of evaluating the fitness of the genotypes, the one thing that they have in common is that they all involve the calculation of the antenna gain at each of the grid points. As may be seen from Equation 2.2.2, the gain pattern is a function only of the weights, the value of the beams' gain function at each point, and the phase shifts of the elements relative to that point. For an individual genotype, the weight factors are constant for all points. To speed computation, the values of all the gain functions and phase shifts which will be needed in the program run are precomputed. They are then stored at each appropriate target point; this avoids unnecessary repetitive computations, as these values can then be simply looked up rather than being recomputed. For the SINR fitness criterion, since the gain pattern must be numerically integrated over the target surface, each target point must represent some appropriate percentage of the total target area. To facilitate this, each target point has associated with it an area equal to 1/3 of the sum of the areas of all the elements of which it is a part; a normalizing factor is also applied so all the areas sum to 1.

The placement of the jammers is a simple process. They are specified by their location in the service area and by their power in decibels relative to the background noise. The beam gains and phase shifts are precalculated for each jammer in exactly the same fashion as they are for the target points.

4.2 Operation of the Algorithm

4.2.1 Parameter Representation

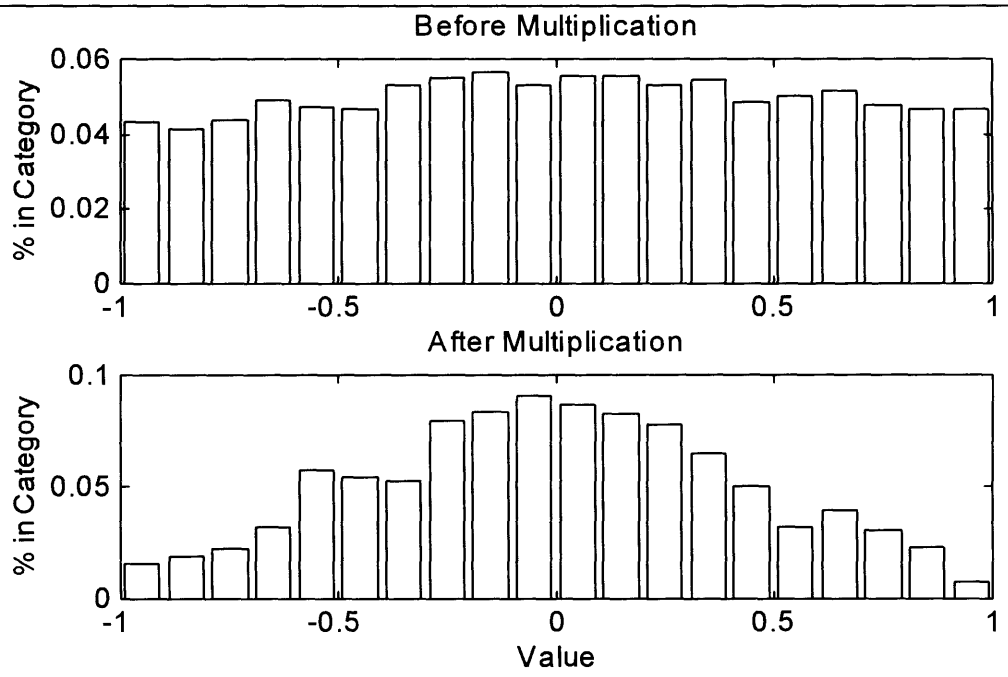
The coding of the parameters into genotypes is fairly simple. Each parameter, constrained to be between -1 and 1, is scaled into an n-bit binary number between 000...0 and 111...1. The transformation is a simple linear one, so that 011...1 corresponds to a weight factor of almost 0. For each antenna beam there are 2 weights, real and imaginary, but the constraints imposed by the S-PACE formulation and the consistent phase requirement reduce this by $2N+1$, where N is the number of jammers. Since the parameters do not correspond in any tangible way to the actual weights which are the result of the matrix transformation (multiplication by the jammer nullspace), the requirement that the solution be well modeled by short, well-defined schemata must be met by the fact that the function space is continuous and is related to the parameter space by simple linear transformations. It is difficult to tell a priori whether, in fact, this condition is met, so the results of these experiments may be taken as empirical evidence of the suitability of the genetic algorithm for this problem.

In practice, the binary representation is actually stored on the computer as an integer value between 0 and 2^n-1; the operators are applied through the use of bitwise arithmetic. In all of the experiments, a two-point crossover operator was used. As its name would suggest, two-point crossover involves the random selection of 2 points along the genotype. Two offspring are created from 2 parents by exchanging the chromosomes which lie between the selected points. Several other types of crossover have been researched in the literature, including multiple-point and adaptive crossover schemes, but there was no strong evidence to suggest that a more complicated operator would be useful for this problem. To carry out the operation, bitwise masks are constructed which, when combined with the parent strings with a logical-and operator, pass only the bit values which are supposed to be copied. The mutation operator is an exclusive-or.

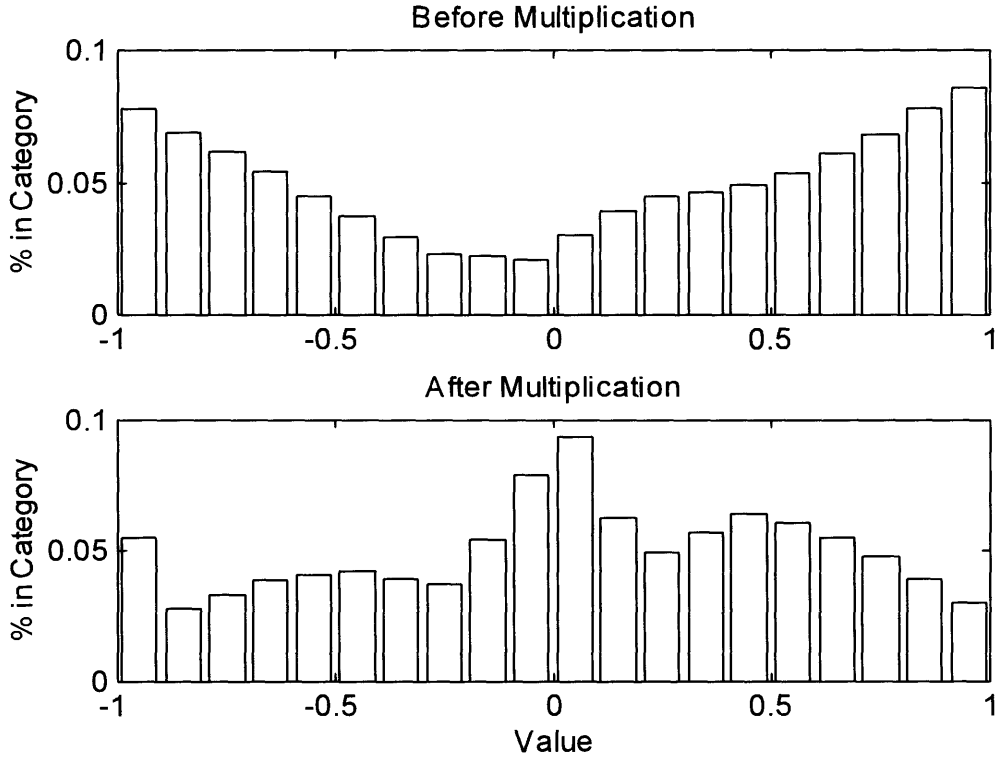
The initialization of the genotypes is a little more subtle than it may first appear; this is due to the fact that there is a matrix transformation which lies between the genotypes and the actual weights. In order to ensure maximum diversity, it is desirable to have a uniform distribution of the initial weights. However, multiplying a set of uniformly distributed weights by a matrix is equivalent to forming a sum of random variables; by the Central Limit Theorem, the distribution of a sum of random variables approaches a Gaussian distribution with probability 1 as the number of summed variables increases to infinity. While the number of variables that are involved in the matrix operation is nowhere near large enough for the Central Limit Theorem to apply, Figure 4.2.1A shows that this tendency is nevertheless observable. In order to obtain a weight distribution which is closer to uniform, the genotypes may be initially created from a non-uniform distribution. As shown in Figure 4.2.1B, an initial distribution with higher variance leads to weight factors which are more likely to take on extreme values closer to -1 and 1. About 1000 initial weights are represented in each of these figures, in order get a reasonably smooth distribution.

4.2.2 Fitness Evaluation

The goal of the reproductive operator is to select the members of the current population which will be acted on by the crossover and mutation operators in order to create the next generation. The criterion for selection is the fitness of each genotype, which is evaluated in several stages. The first stage is the estimation of an unscaled, or *raw* fitness, which is calculated by evaluating some aspect of the gain function of the antenna. These raw fitness values are then scaled in such a manner that the more fit genotypes are clearly separated from the less fit ones, and that their final scaled fitnesses represent in some fashion a probability that they will be selected to be in a mating pair.



A: Uniform initial distribution



B: Nonuniform initial distribution

Figure 4.2.1 A,B: Effect of matrix multiplication on initial weights

Several fitness criteria were applied, with the goal of evaluating different aspects of the algorithm's performance. They are:

- Signal to Interference-plus-Noise Ratio (SINR)
- Percent Coverage Area (PCA)
- Maximized minimum gain (maxmin)

Each of these will be explained in turn.

SINR is interesting from the point of view of the Howells-Applebaum algorithm, whose original form is designed to solve the problem of finding the weights which give the maximum value of signal to noise ratio given a known steering vector **v**. In that case, the SINR is simply

$$SINR = \frac{G(\mathbf{x}_u)^2}{\sum_J G(\mathbf{x}_j)^2 + \sigma^2} \tag{4.2.1}$$

where the user is located at $\mathbf{x}_u$ and the jammers are located at the $\mathbf{x}_j$. For the area-coverage problem, this criterion must undergo some modification. If the SINR is redefined as

$$SINR = \frac{\iint_\Omega G(x,y)^2 dxdy}{\sum_{N_J} G(\mathbf{x}_j)^2 + \sigma^2} \tag{4.2.2}$$

where G is the antenna gain function, Ω is the coverage area, N_J is the number of jammers, and σ^2 is the noise power. In order to estimate this function numerically, the integral of the antenna gain over the coverage area may be approximated as a finite sum over the number of target points P

$$\iint_\Omega G(x,y)^2 dxdy \approx \sum_P G(\mathbf{x}_i)^2 A_i \tag{4.2.3}$$

where A_i is the area associated with point i.

PCA is a measure of how much service area is lost due to the presence of jamming. In an interference-free environment, the antenna operates with a set of optimal "quiescent" weights which result in a gain function which exhibits as high a minimum gain as possible. In the presence of interference, some of this performance must be sacrificed in order to place nulls on the jammers; therefore it is assumed that service is maintained everywhere that the gain remains above some threshold value P_{min}. The PCA is defined as

$$\frac{\iint_{\Omega} 1_{\{G(x,y)^2 \geq P_{\min}\}} dxdy}{\iint_{\Omega} dxdy} \tag{4.2.4}$$

$$1_{\{x \geq N\}} = \begin{cases} 1 & x \geq N \\ 0 & x < N \end{cases} \tag{4.2.5}$$

PCA is therefore the fraction of coverage area which is retained in the presence of interference. While this metric makes more sense from a practical point of view (maintaining coverage is, after all, the real objective), the value P_{min} is rather arbitrary; for the purposes of this investigation it was taken to be 10dB below the minimum quiescent gain in order to allow comparisons with the results in [6]. Not only is P_{min} arbitrary, but the PCA is quite sensitive to its value. Therefore, care must be taken in choosing its value if consistent results are to be obtained.

The third type of fitness function, maxmin, is designed to find the optimal quiescent weights which are to be used when no interference is present. Its value is simply

$$\min\left\{G(\mathbf{x}_i)^2 : i \in [1 \ldots P]\right\} \tag{4.2.6}$$

The optimal quiescent weights were used to determine the corresponding threshold value P_{min} used in the PCA fitness evaluation, based on the minimum value of the quiescent gain function.

4.2.3 The Reproductive Operator

The values of the raw fitness cannot be used directly in the reproductive selection procedure for several reasons.

1. In the initial stages of the genetic algorithm, a genotype with exceptionally high fitness must not be allowed to destroy the diversity of the population by reproducing itself many times.
2. The fitness values should be forced to be between definite bounds so that the operation of the algorithm is consistent.
3. In the later generations when most of the genotypes have a fairly high raw fitness, there must still be a wide range of scaled fitnesses so that the algorithm does not lose direction.

In order to ensure that these conditions are met, the raw fitnesses F_i^0 are subjected to two transformations. The first is an exponential scaling

$$F_i^1 = 1 - \exp(-k * F_i^0) \tag{4.2.7}$$

where k is chosen such that the average value of the F_i^1 is 1/2. This transformation does the bulk of the work in satisfying the 3 conditions. Notice that

- F_i^1 lies between 0 and 1 for all i.
- Since the average scaled fitness is ensured to be 1/2, a single genotype with high raw fitness relative to the rest of the population will not force all the others to have low scaled fitnesses.
- Since the average scaled fitness is 1/2, it is the relative rather that absolute values of the raw fitnesses which determine the scaled fitnesses.

An example of this transformation applied to a generation is shown in Figure 4.2.2.

The second transformation is less important, but still influential. This transformation is a simple linear one, in which the maximum value of F_i^1 is again scaled to 1, and a hypothetical genotype with an exactly average value of F_i^1 is rescaled to 1/2. The formula for this is

$$F_i = \frac{F_i^1 + F_{max}^1 - 2F_{ave}^1}{2\left(F_{max}^1 - F_{ave}^1\right)} \tag{4.2.8}$$

where F_{max}^1 is the largest of the F_i^1, and F_{ave}^1 is their average. Any F_i which comes out to be less than zero is set to zero, as these values are interpreted as probabilities when the reproduction operator is applied. Figure 4.2.3 shows two examples.

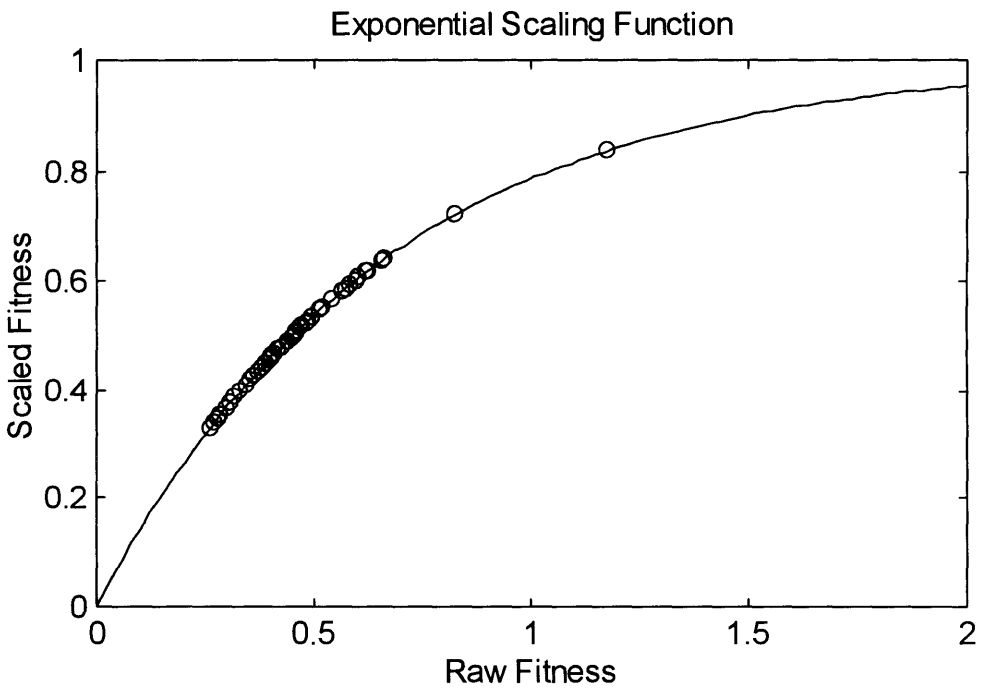


Figure 4.2.2: Exponential scaling function
Notice outlier at raw fitness = 1.177

The fitness values in Figures 4.2.2 and 4.2.3 are from an actual program run, the results of which are shown in Figure 5.2.5. Notice that if exponential scaling is applied, then the rest of the population shows a wider, and hence more accurately differentiated, range of fitness values.

The exact manner in which the reproductive operator is applied has been the subject of much discussion in the literature (for example, [Baker 1]). The objective is to insert a number of copies of each genotype equal to

$$n_i = \frac{F_i}{F_{ave}} \tag{4.2.9}$$

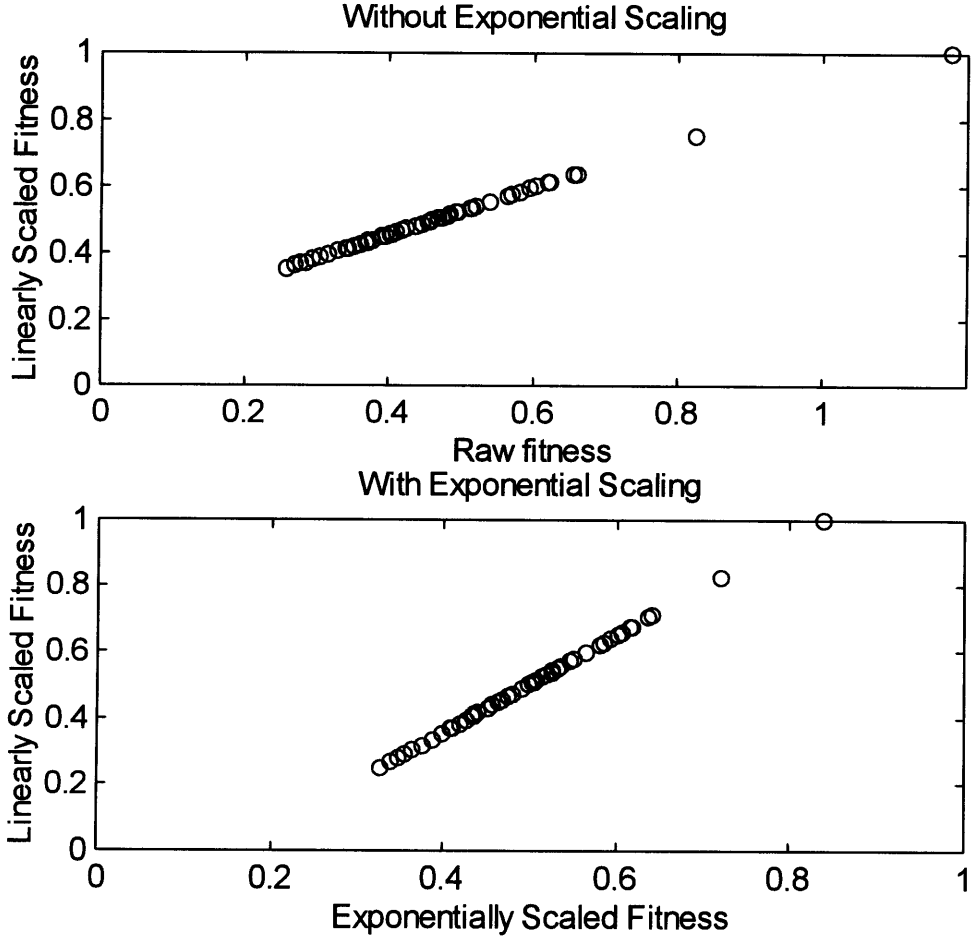


Figure 4.2.3: Effects of initial exponential scaling (See also Figure 5.2.5)

Of course, this fraction is, in general, not an integer, and therefore it is not possible to implement this formula in a direct fashion. The objective for designing a reproductive operator, then, is to try to ensure that over a large number of population members and generations that the average number of copies of each genotype will be close to n_i. The number of copies of the ith genotype which are created as a result of the application of a reproductive operator R is a random variable X_{Ri} with mean close to n_i, and (hopefully) a small variance. The terms used to describe the parameters of this random variable in the literature are *bias* and *spread*. Bias is determined by

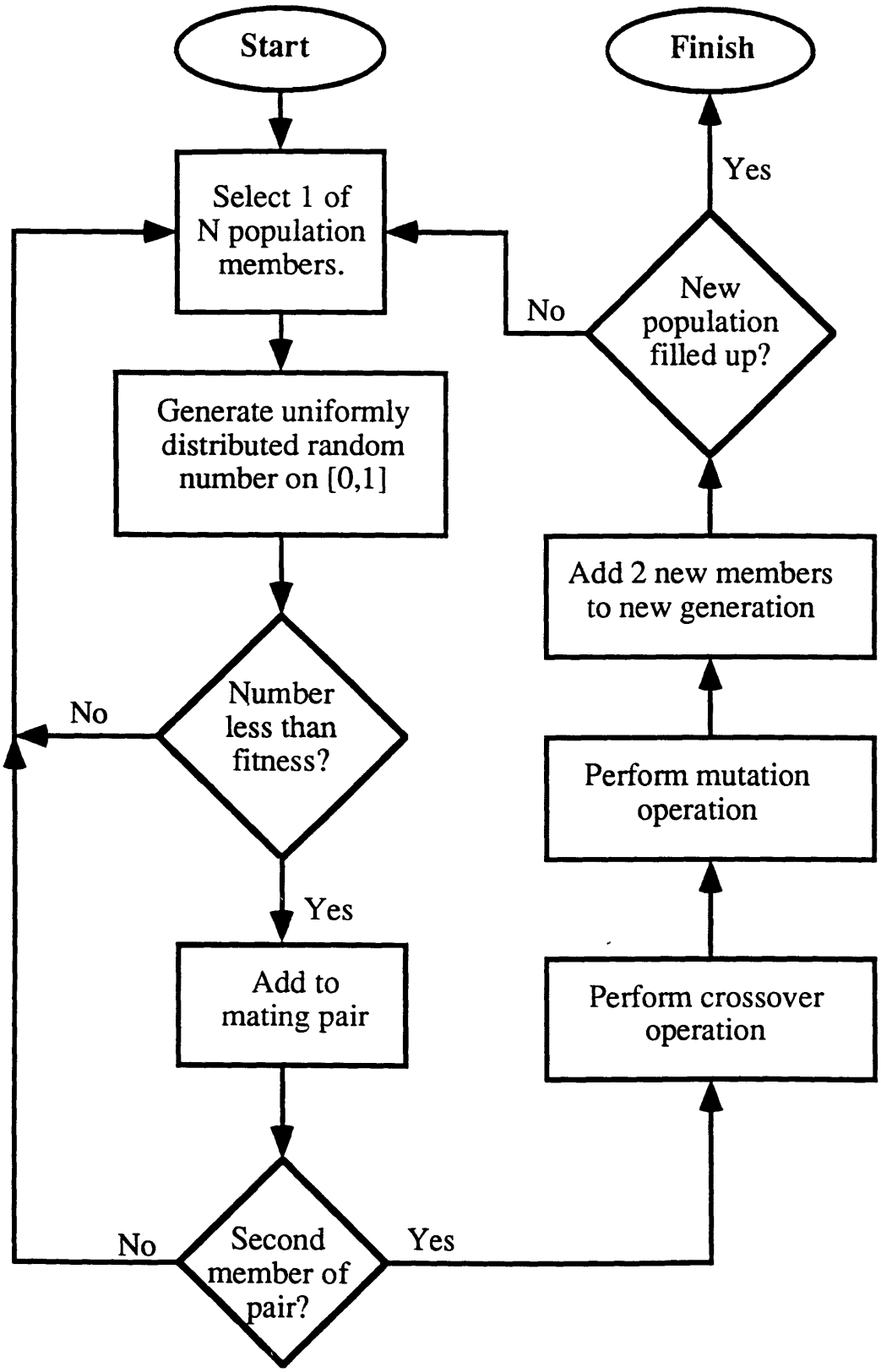
$$bias = \left| E[X_{R_i}] - n_i \right| \tag{4.2.10}$$

or the difference between the expected value of the number of copies of the ith genotype and it's fraction of the fitness n_i. The spread is defined as the range of values which X_R can take, which may be visualized as being loosely related to the variance. In this thesis, a simple interpretation of the reproductive operator was implemented. Reproductive pairs were chosen from the population as a whole by way of the following algorithm:

1. Select a member of the population at random
2. Generate a random number between 0 and 1. If this number is less than the fitness of the chosen member, add it to the pool. Otherwise, return to step 1.
3. If this newly selected member is the second of the pair, apply the crossover and mutation operators to generate 2 new genotypes. If it is only the first of the pair, return to step 1.

This process is diagrammed in Figure 4.2.4. There is also a test to ensure that both of the "parent" genotypes are not the same one. The expected value and variance of this operator may be analyzed in a fairly straightforward manner. Since an "average" member receives a scaled fitness value of 1/2, the expected value of the fitness observed in step 2 is also 1/2.

Figure 4.2.4: The reproduction operator



Therefore, the expected value of the number of times that step 1 must be carried out is twice the population size N. Then, since the average fitness $1/N\sum_J F_J = 1/2$, then

$$n_i = \frac{F_i}{1/N\sum_J F_J} = 2F_i \Rightarrow F_i = \frac{n_i}{2} \tag{4.2.11}$$

and the expected value of the number of copies of genotype i is the product of the expected values of the number of times it is visited and the chance that it passes the test in step 2, which gives simply: $2N * \frac{1}{N} * \frac{n_i}{2} = n_i$. Therefore it would appear that this operator has zero bias. Actually, there is a slight bias towards genotypes with high fitness, as a result of the fact that negative scaled fitnesses are defined to be zero. This effect is very small, however, as these very unfit genotypes are uncommon. The spread, on the other hand, is not quite so well behaved. It is certainly a possibility that the same pair could be picked to produce the entire next generation, giving a spread of $N/2$; however, the probability of this occurring becomes vanishingly small as N increases. The concept of spread, then, does not seem to be a reasonable criterion to apply to this particular reproductive operator.

The distribution of X (the number of copies of a genotype which are selected) as a function of n_i would perhaps give more useful information. First, define a sequence of Bernoulli random variables $B_1, B_2, \ldots B_n$, and their sum $S_n = \sum_{k=1}^{n} B_k$. The probability of failing step 2 of the reproduction operator may be evaluated as

$$\sum_{i=1}^{N} P(Y=i)P(Z<F_i) = \frac{1}{N}\sum_{i=1}^{N}(1-F_i) = 1-\frac{1}{N}\sum_{i=1}^{N}F_i = 1-F_{ave} = \frac{1}{2} \tag{4.2.12}$$

where

$$P(Y=i) = P(\text{picking the } i\text{th member at step 1}) = \frac{1}{N} \tag{4.2.13}$$

and Z is uniformly distributed on (0,1). Therefore, each iteration of step one is equivalent to a Bernoulli trial, where success indicates that a member of the population has been selected to reproduce into the next generation. S_n counts the number of members which have been selected, and the reproductive process continues until S_n=N. Define

$$\tau = \inf\{n: S_n = N\} \tag{4.2.14}$$

Then

$$P(X = m|\tau = n) = \sum_{j=m}^{K-N+m} \binom{K}{j}\left(\frac{1}{N}\right)^{j}\left(1-\frac{1}{N}\right)^{K-j}\binom{j}{m}F_i^m(1-F_i)^{j-m} \tag{4.2.15}$$

The distribution of τ may be determined recursively from S_n by the relation

$$P(\tau = n) = \frac{1}{2}P(S_{n-1} = n-1) \tag{4.2.16}$$

since the chance of success on each trial is 1/2. The distribution of S_n may be obtained in a straightforward manner, as it is simply the n-th convolution of B with itself. Therefore,

$$P(S_n = k) = \binom{n}{k}\left(\frac{1}{2}\right)^{k}\left(\frac{1}{2}\right)^{n-k} = \begin{cases}\binom{n}{k}\left(\frac{1}{2}\right)^{n} & k \le n \\ 0 & k > n\end{cases} \tag{4.2.17}$$

Therefore,

$$P(\tau = n) = \binom{n-1}{N-1}\left(\frac{1}{2}\right)^{n} \tag{4.2.18}$$

Given this fact, the calculation of the distribution of X is straightforward.

$$P(X=m)=\sum_{n=N}^{\infty}P(X=m|\tau=n)P(\tau=n)$$

$$P(X=m)=\sum_{n=N}^{\infty}P(\tau=n)\sum_{j=m}^{n-N+m}\binom{n}{j}\binom{j}{m}F_i^m(1-F_i)^{j-m}\left(\frac{1}{N}\right)^j\left(1-\frac{1}{N}\right)^{n-j}$$

$$P(X=m)=\sum_{n=0}^{\infty}P(\tau=n+N)\sum_{j=0}^{n}\binom{n+N}{j+m}\binom{j+m}{m}F_i^m(1-F_i)^j\left(\frac{1}{N}\right)^{j+m}\left(1-\frac{1}{N}\right)^{n+N-j-m}$$

$$P(X=m)=\sum_{n=0}^{\infty}\binom{n+N-1}{N-1}\left(\frac{1}{2}\right)^n\sum_{j=0}^{n}\binom{n+N}{j+m}\binom{j+m}{m}F_i^m(1-F_k)^j\left(\frac{1}{N}\right)^{j+m}\left(1-\frac{1}{N}\right)^{n-j+N-m}$$

(4.2.19-22)

Plots of this function for various values of n_i are shown in Figure 4.2.5. The expected value and variance of this reproduction operator are both equal to n_i.

One drawback of this specific implementation of the reproductive operator is that

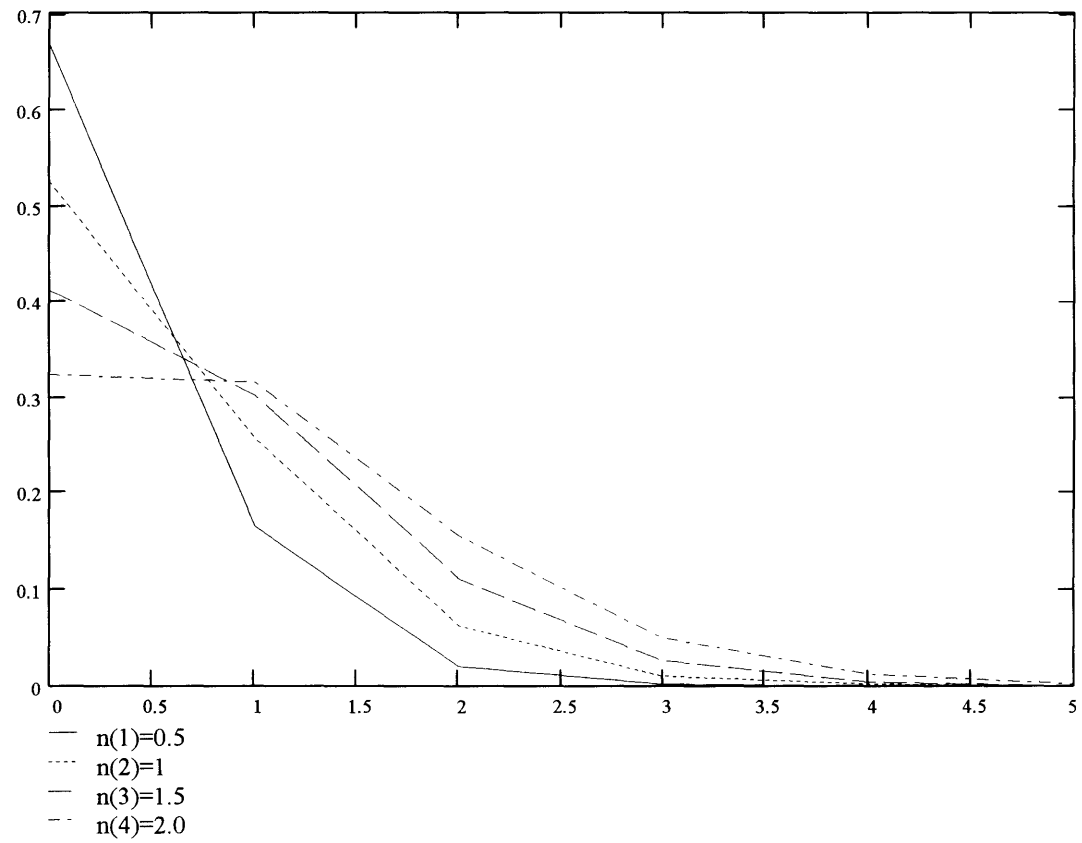


Figure 4.2.5: P(X=m) for various values of n_i

it is possible for the most fit member to be lost, due to the fact that it will not be part of the mating pool with certainty. In order to guard against this possibility, the most fit member is automatically copied directly into the next generation before the probabilistic stage is carried out. This ensures that the maximum fitness will never decrease from one generation to the next.

4.3 Parameter Optimization

A feature of genetic algorithms which limits their applicability to arbitrary problems is their sensitivity to the crossover and mutation probabilities, and to the size of the population. In order to analyze the effects of these parameters on the performance of the algorithm, a variety of simulations were run using a wide range of values. These simulations were designed to determine the optimal values of crossover rate, mutation rate, and population size on two statistics which may be considered to be primary indications of the relative performance of the algorithm, namely the ability to consistently converge to a good solution, and the speed at which this convergence is obtained. Since the genetic algorithm is stochastic in nature, it is necessary to do a fairly large number of trials at each set of parameter values in order to get a clear picture of their effect; however, there is an inherent tradeoff between the accuracy of the results and the range of values which may be explored.

For consistency, all of these trials used the SINR criterion. In order to estimate the consistency of convergence, the algorithm was run through a fixed number of gain evaluations using a variety of parameter values. The tests run to examine the effect of crossover rate are shown in Table 4.1, and those run to examine the effects of mutation rate and population size are shown in Table 4.2. Each case was run through 50 complete runs, with each run consisting of 4000 gain evaluations. In order to make the results comparable, each run was conducted using the same jamming environment. The choice of this environment was completely arbitrary; trying to get enough program runs using a wide variety of jammer configurations would have taken a prohibitively long time.

Population Size	Mutation %	Crossover %
51	0.5	20
51	0.5	40
51	0.5	60
51	0.5	80

Table 4.1: Crossover rate tests

Population Size	Mutation %	Crossover %
21	0.1, 0.5, 1	80
51	0.05, 0.1, 0.5, 1	80
81	0.05, 0.1, 0.5, 1	80
111	0.1, 0.5, 1	80

Table 4.2: Mutation rate and population size tests

Therefore, a "typical" case consisting of 2 jammers located in opposite hemispheres of the target area was chosen, and there was no indication that this specific choice of problems may have led to inaccurate results. Jay Simon's observation that it was in the 2 jammer case that S-PACE seemed to show the most improvement over the Howells-Applebaum algorithm was the basis for this choice of the number of jammers.

The results of these trials are best displayed in graphical form. For each one, the sample mean and standard deviation of the best member in the last generation were calculated to give a feel for the characteristics of the distribution (Figures 4.3.1 A,B,C,D, Figure 4.3.2). The results show some definite trends which are to be expected of reasonable performance of a genetic algorithm. In summary:

- Mean and standard deviation of the converged fitnesses were similar for crossover rates of 60% and above, with steadily decreasing mean and increasing standard deviation for rates of 40% and below.
- Mutation rates of 0.5% and 1% gave similar performance, with lower rates causing a decrease in mean and an increase in variance. This effect was less pronounced for larger population sizes.

- Population sizes of 51 and 81 had similar performance. Both smaller and larger populations led to a mild decrease in mean and small increase in standard deviation. For small populations, this effect may be due to a decrease in the initial diversity of the population which could not be atoned for by a higher mutation rate. For a larger population, there was plenty of diversity but the algorithm was not allowed to run through a sufficient number of generations to allow the genetic operators to propagate the genetic information.

Of these results, perhaps the most interesting is the decrease in performance for a population of more that 81 members. Larger and larger populations may be visualized as approaching a point at which the population consists of 4000 members (the maximum number of gain evaluations), to which none of the genetic operators are applied; this

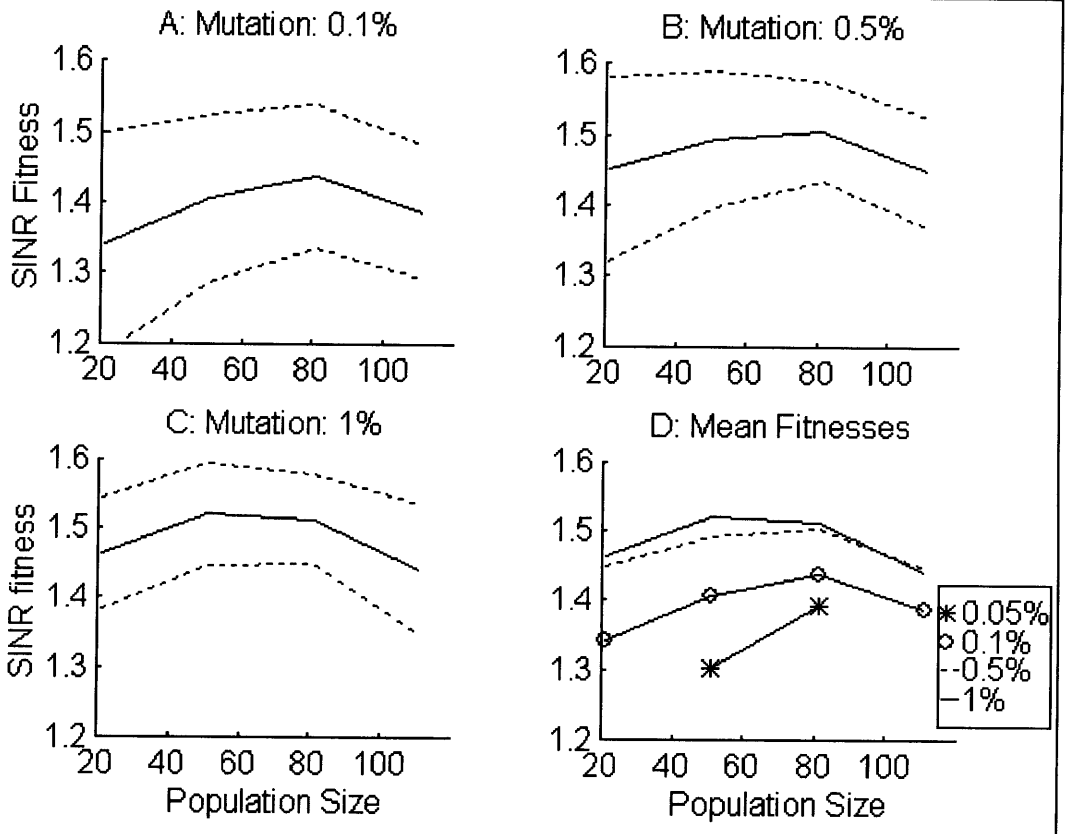


Figure 4.3.1: Effects of mutation rate and population size
A,B,C: Dotted lines show the boundary of one standard deviation
D: Mean fitnesses for all mutation rates

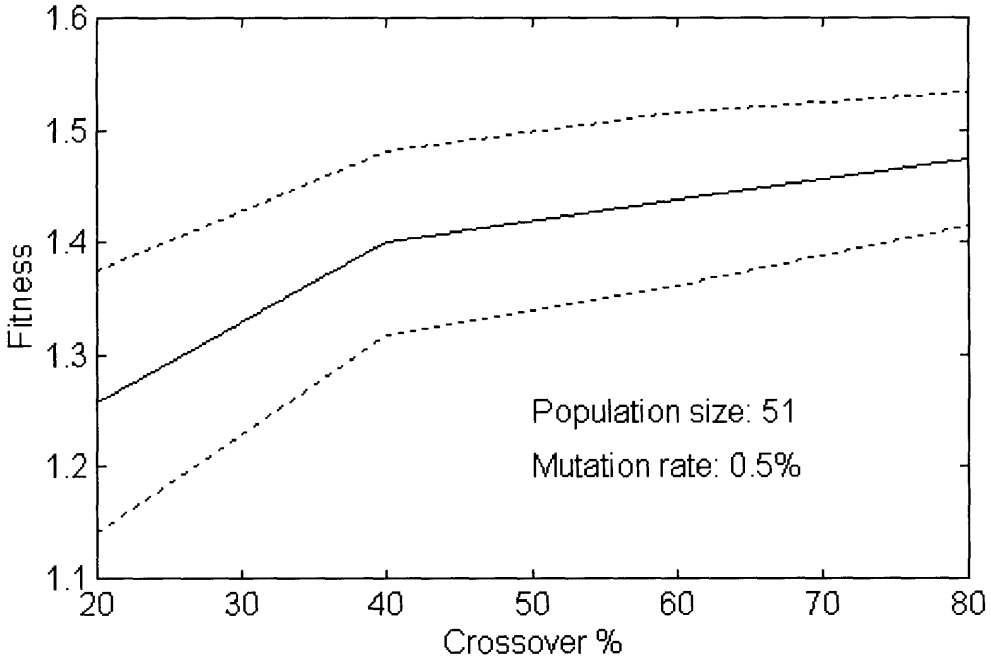


Figure 4.3.2: Effect of crossover rate
Dotted lines show the boundary of one standard deviation

limiting case is equivalent to a random search. In order to examine this case, 8402 random phenotypes were generated and evaluated, and their fitnesses are displayed in Figure 4.3.3 in raw form and as an estimated probability distribution.

The probability distribution was estimated by noting that in a region in which the distribution function $p_X(x)$ is constant, the expected number m out of N total samples which fall between x-$\alpha/2$ and x+$\alpha/2$ is

$$m = Np(x)\alpha \tag{4.3.1}$$

$p_X(x)$ may be estimated either by fixing m and finding $\alpha(x)$, or by fixing α and finding $m(x)$. For the estimates in this project, the former method was used, and the resulting distributions were smoothed using a moving average filter. The results show that even for this large a population, the chance of randomly generating a genotype which has a fitness value approaching that of a typical iteration of the genetic algorithm using near-optimal parameters is very small. This provides strong evidence that the algorithm is performing in an efficient manner.

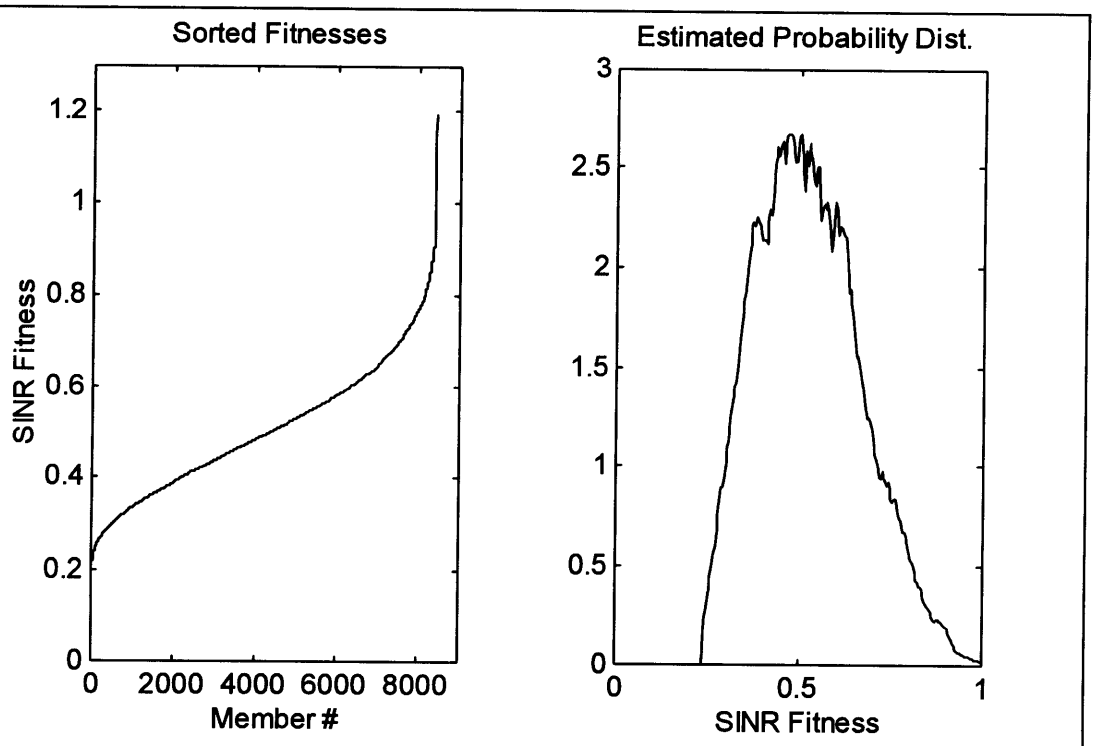


Figure 4.3.3: Fitnesses of random phenotypes

One type of plot which has proven to be invaluable in evaluating in a heuristic manner the performance of this genetic algorithm is shown in Figures 4.3.4 and 4.3.5. This plot shows the fitness of every single member of the population in every generation, grouped by iteration and sorted by fitness. These 2 particular plots were chosen because they illustrate very well why a mutation rate which is too low or too high can seriously impede the progress of the algorithm.

In Figure 4.3.4, the mutation rate corresponds to an expected value of less than 4 mutations per generation. It is clear that once the initial diversity of the population is exhausted (by about iteration 15), the algorithm becomes stuck in a series of local minima, and the crossover operator does little more than make all the genotypes into copies of each other. Figure 4.3.5 shows a case for which the expected number of mutations per iteration is 367, or about 7.2 per genotype (there are a total of 144 genes per genotype in this case). Here the mutation operator is so powerful that it is difficult for the crossover

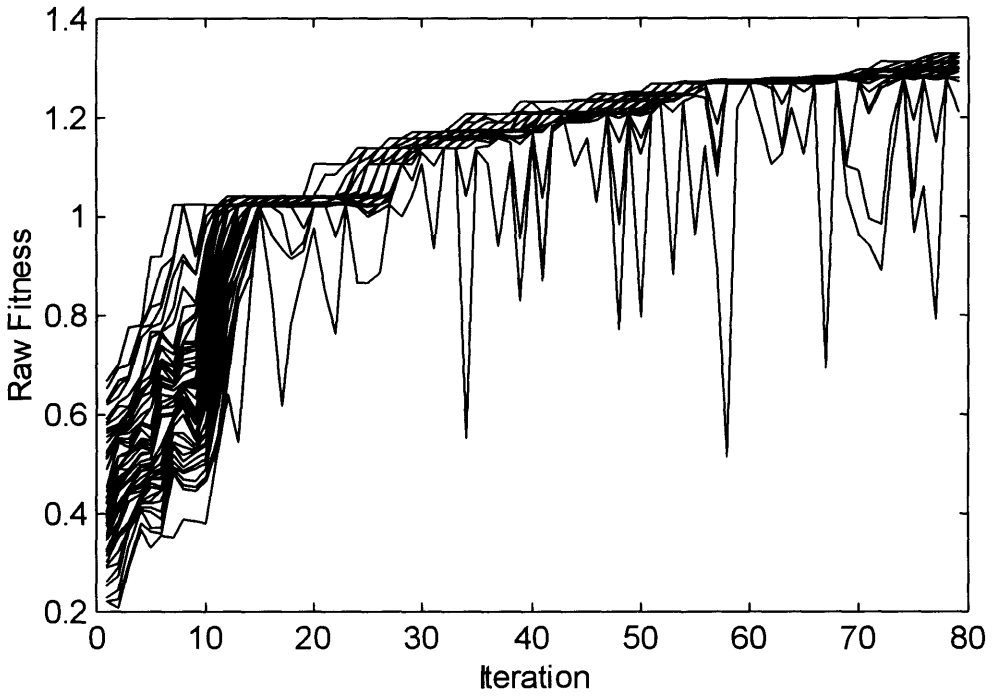


Figure 4.3.4: Convergence history with mutation rate = 0.05%

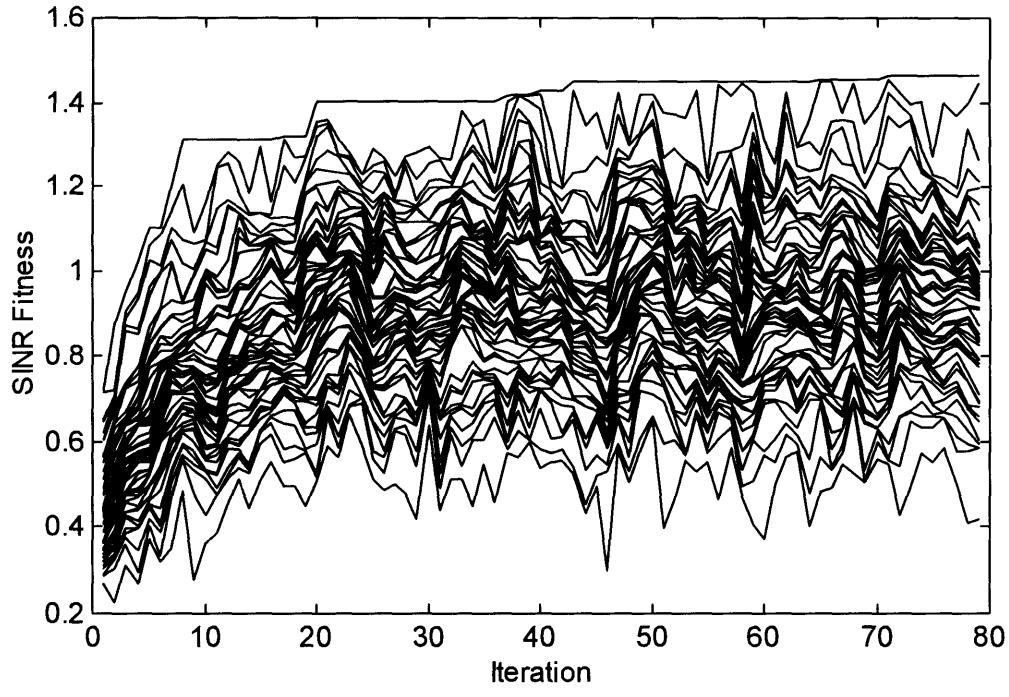


Figure 4.3.5: Convergence history with mutation rate = 5%

operator to have any effect. This is manifested in the fact that the population does not significantly increase in average fitness after about the 20th generation.

The other aspect of algorithm performance for which the effects of these parameters was explored is average time to convergence. The point at which an algorithm such as this one has actually converged is difficult to define, due to the fact that there is no guarantee that a global optimum will be found. One option is to define convergence as that point at which no significant progress in the maximum fitness has been attained. However, for a non-gradient based search such as a genetic algorithm, it is quite possible to make very little progress over many iterations, only to suddenly realize a significant improvement. Since the purpose of these trials was not to determine the amount of time necessary to achieve complete convergence, but rather to compare the speed at which a good solution is obtained, a somewhat different approach was taken. A large number of trials were conducted using "good" values for the algorithm's parameters. Then the mean and standard deviation of this collection of results was determined, and a reasonably good solution was defined as one which comes within one standard deviation of this mean. Then the number of fitness evaluations which it took to achieve this level of fitness was statistically analyzed over a range of parameter values with the goal of finding general trends (Figures 4.3.6). Trials which did not achieve this target fitness value within 4000 fitness evaluations were discarded. Again, very low mutation rates, crossover rates, and population sizes were correlated with poor performance; optimal values were approximately: 51-81 genotypes in each generation, 80% crossover rate, and 0.5-1% mutation rate. It took an average of about 600 fitness evaluations to achieve the target fitness using these parameters, with a standard deviation of about 250, and a failure rate of about 5%.

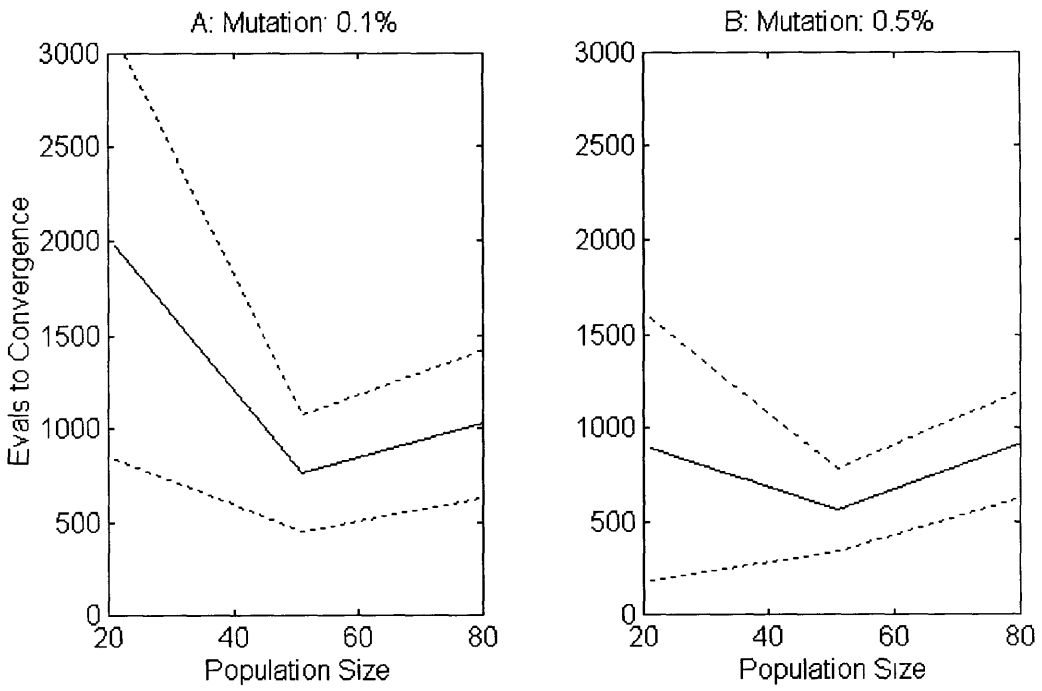


Figure 4.3.6: Times to convergence

A,B: Dotted lines show the boundary of one standard deviation

Crossover rate: 80%

Chapter 5

Results

The results of this project may be broken down into several different parts. The experiments concerning the convergence time and the performance as a function of the input parameters have already been discussed. Secondly, the algorithm was compared with data from Jay Simon's S-PACE paper, both to check the consistency of the results, and also to compare the respective algorithms. In addition, general characteristics of the formulation of the problem were explored, including the use of the SINR as opposed to the PCA fitness criterion, and the effects of varying the antenna configuration and the problem's constraints.

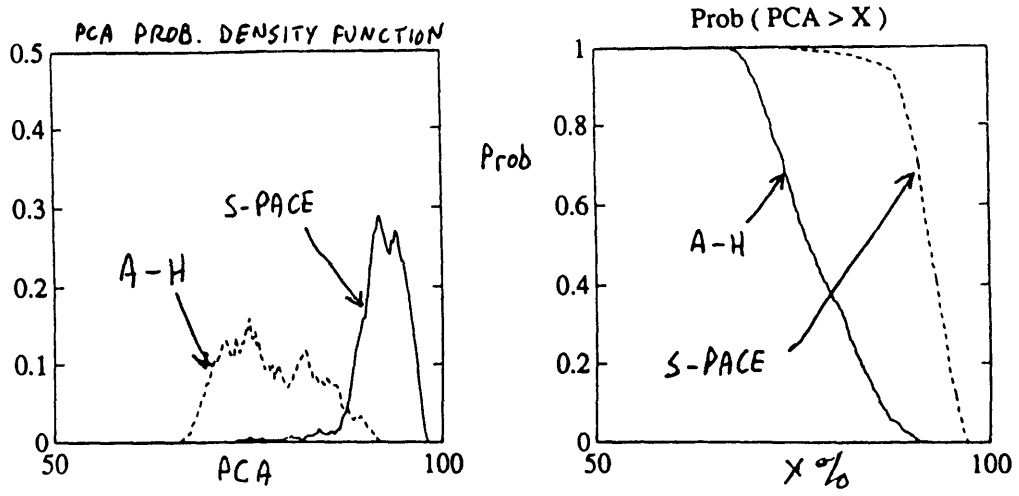
5.1 PCA Evaluation

The PCA fitness criterion was used to compare the performance of this genetic algorithm with that of the Metropoulis simulated annealing algorithm used by Jay Simon. His primary measure of the effectiveness of the algorithm involved running a large number of optimizations on random configurations of various numbers of jammers. He found that it was in the 2-jammer scenario that the S-PACE algorithm offered the most improvement; this was therefore the case which was analyzed in this project.

The jammers which were used were relatively powerful, being 40dB stronger than the noise. They were placed at random in the coverage area, with the sole constraint being that they had to be far enough apart to be resolvable as 2 separate sources (1/3 of a beamwidth was used as the criterion for this, as per the S-PACE paper). The first experiments were designed to examine the performance of the Howells-Applebaum formulation, using the quiescent weights as the weight vector $\mathbf{v}$. Simon ran the Howells-Applebaum algorithm through 340 trials, and analyzed an estimate of the probability

distribution of the PCA which resulted. This distribution is presented in Figure 5.1.1,

Figure 5 1.1: Results for non-optimized (A-H) and optimized (Metropoulis) algorithms
From Simon[6]



along with a plot of $P(\text{PCA}>x)$.

In these experiments, as in all the others that will be discussed in the chapter, the parameters were:

- Population size: 51
- Crossover: 80%
- Mutation: 0.5%

The mean PCA was 78.0%, with almost all values lying between 70% and 90%. The results obtained in this project by repeating this experiment with a sample size of about 1400 were similar (Figures 5.1.2, 5.1.3); in this case the mean was 77.3%, and the range of values was slightly larger. Note that this does not involve any optimization of the vector **v** in Equation 2.2.6; it is simply the result of premultiplying **v** by Φ^{-1}. To analyze the performance of the genetic algorithm, the same experiments were run for a total of 50 trials, with each trial lasting for 4000 fitness evaluations. The average PCA was increased to 94.6%, as compared to the 92.2% PCA achieved with the Metropoulis optimization. The majority of values lay between 88% and 98%.

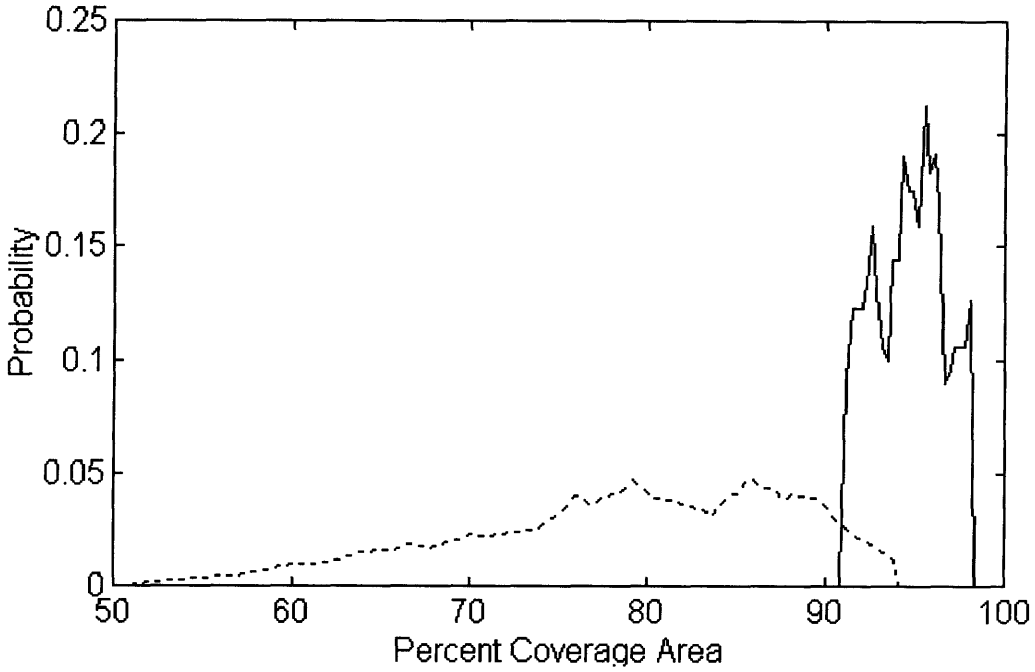


Figure 5.1.2: PCA Performance of the 7 beam antenna
Dotted line is A-H, solid line is genetic algorithm

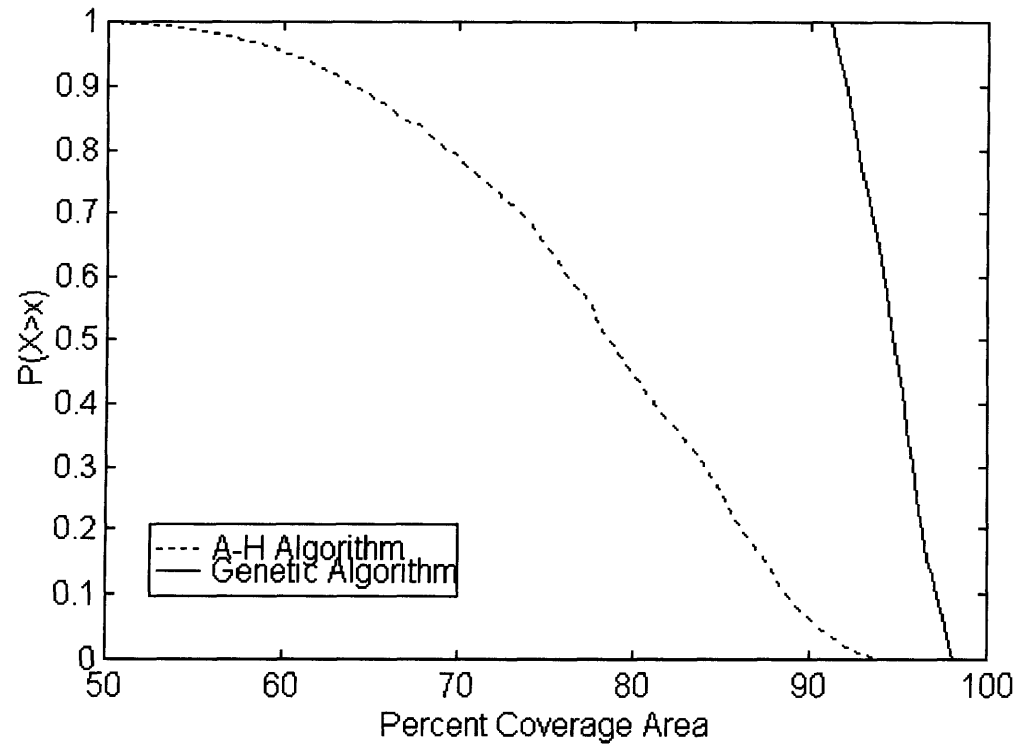


Figure 5.1.3: $P(\text{PCA}>x)$ for 7 beam antenna

To determine the effect of increasing the number of degrees of freedom which the algorithm has to work with, an antenna setup with 19 beams was run through the same test. The outer ring of 6 beams was replaced with a ring of 12, and an additional, intermediate ring was placed between this outer ring and the central beam. The gain pattern of the beams was unchanged, and the crossover gain was -1.2dB. An clearer idea of the construction of this antenna model may be obtained from Figure 5.1.4, which shows the crossover contours of the beams as well as an outline of the target. Using the Applebaum-Howells algorithm alone, the performance of this antenna was slightly better than that of the 7 beam antenna, with a mean PCA of 78.4% obtained from 300 samples. Applying the genetic algorithm to the problem resulted in a dramatic improvement in PCA, with the mean rising to 97.9% (Figures 5.1.5, 5.1.6). The algorithm was able to take excellent advantage of the additional freedom provided by the increased number of beams; a typical gain contour is shown in Figure 5.3.4.

One issue which may be raised is that although the 19 beam antenna achieves a superior solution in the same number of iterations as the 7 beam antenna, it would take about 19/7 as long to calculate one iteration. With this in mind, the 19 beam antenna was run another 50 times, but only allowed to complete 1470 gain evaluations (=4000*7/19). The results of this experiment are shown in Figure 5.1.7. The main features of this graph are (See Table A.4)

- The 19 beam, 4000 evaluation scenario produces much better results than the 7 beam, 4000 evaluation scenario.
- The 19 beam, 1470 evaluation scenario performs worse than the corresponding 4000 evaluation case, but is still a noticeable improvement over the 7 beam case.

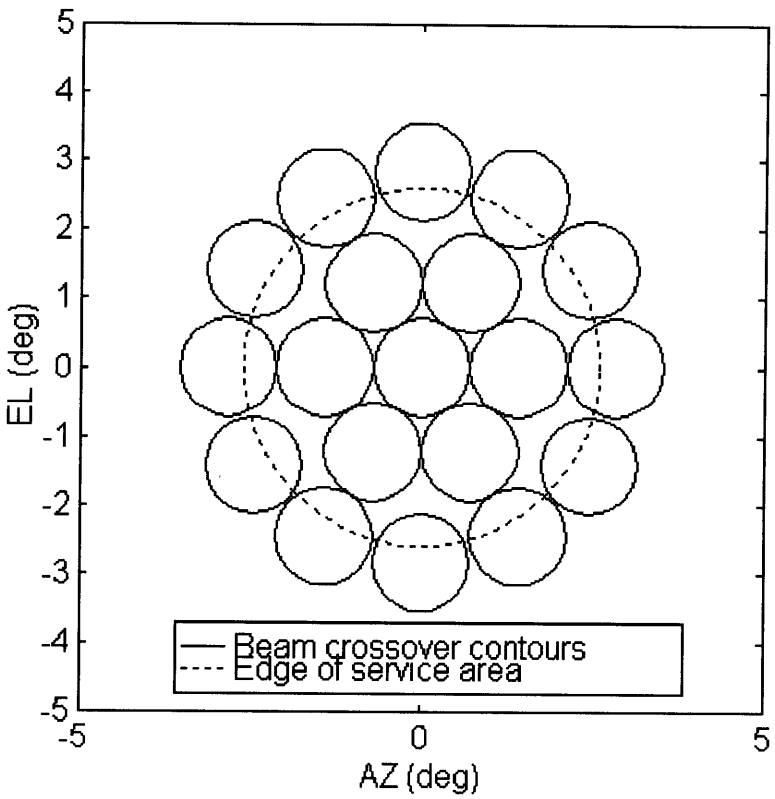


Figure 5.1.4: 19 beam antenna

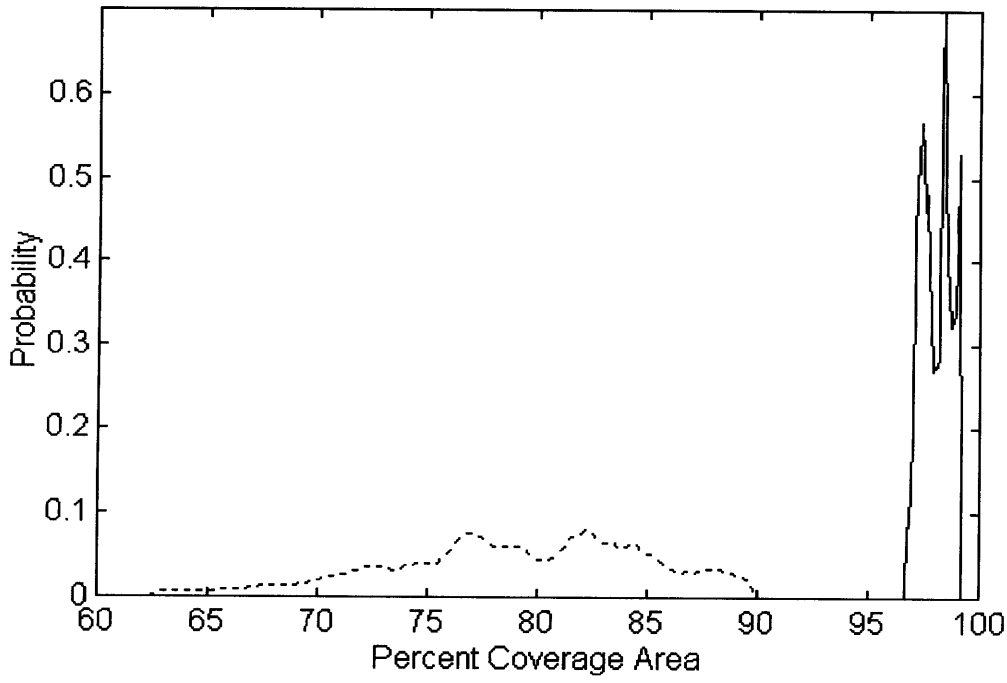


Figure 5.1.5: PCA performance of the 19 beam antenna

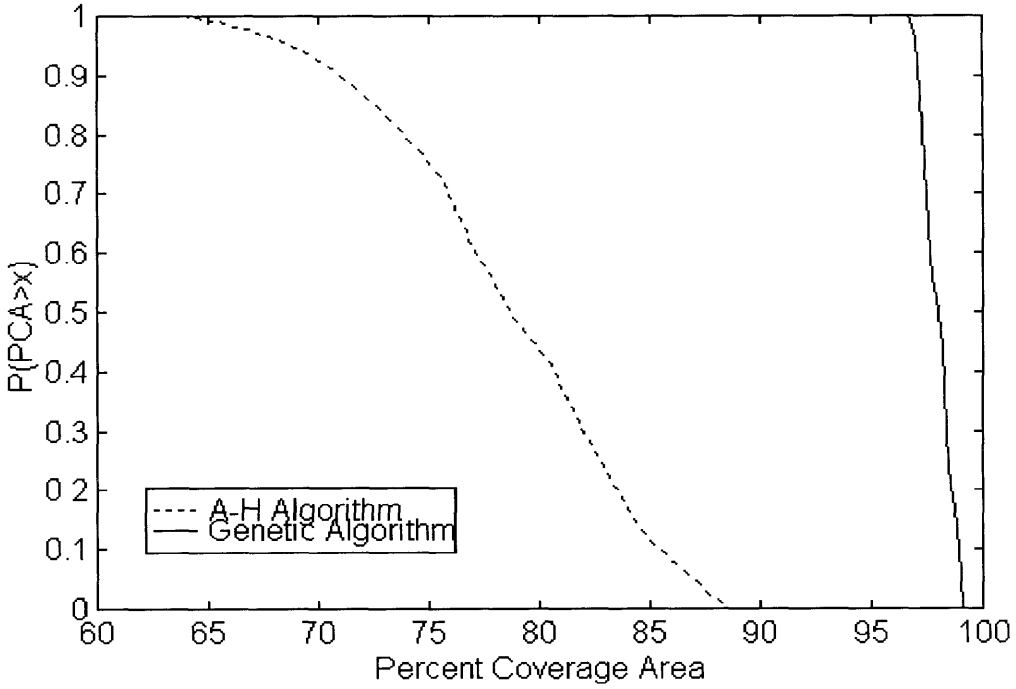


Figure 5.1.6: $P(\text{PCA}>x)$ for 19 beam antenna

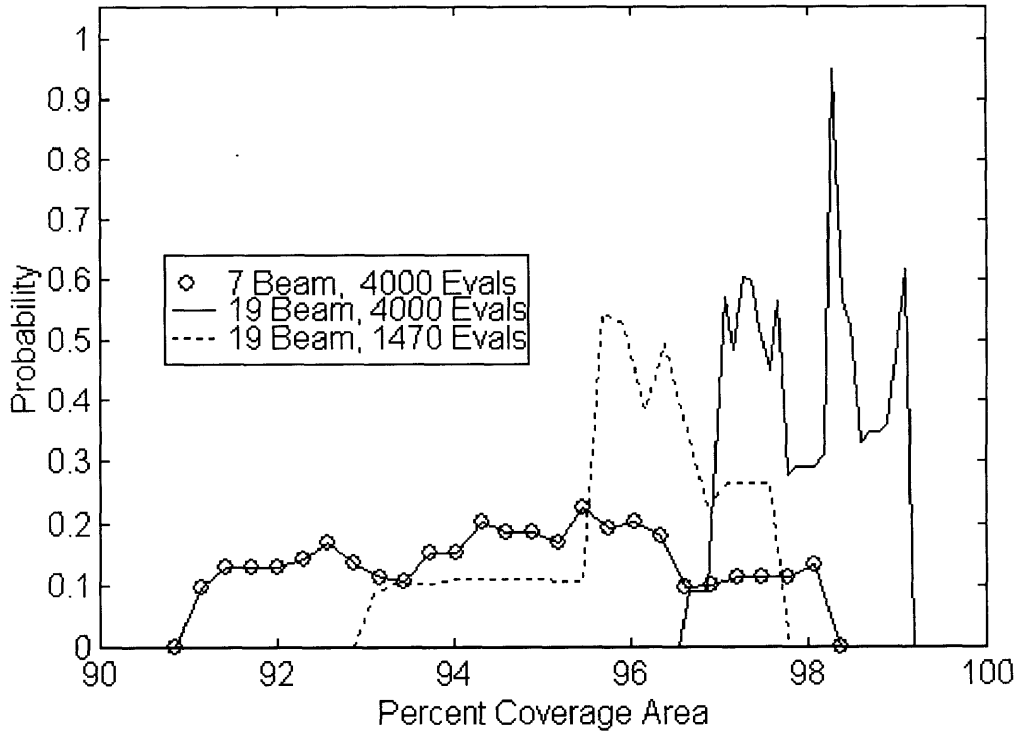


Figure 5.1.7: PCA criterion, varying numbers of beams and fitness evaluation limits

5.2 Convergence Evaluation

It is at least as important to look at the way in which the solutions are obtained as it is to look at their values. To this end, a variety of convergence histories will be presented and evaluated. There are a few general properties of these graphs which are unique to each antenna type and problem variation. First, as may be seen from Figures 5.2.1-5.2.4, it should be noted that the 19 beam antenna takes longer to reach its converged value than does the 7 beam antenna. This is not surprising, given that the 19 beam antenna both has much higher dimensionality than the 7 beam antenna, and also converges to a higher fitness value. Interestingly, it seems to converge more quickly on the PCA convergence criterion than on the SINR criterion. This may be due to the fact that it is actually an easier criterion to satisfy; since the only requirement is that the gain must remain above the threshold value, there would seem to be more flexibility in the solution. In contrast, the SINR criterion is affected by the entire coverage area, so small improvements in fitness may continue to be made even after a large number of iterations.

It appears from the convergence histories that the same parameter values appear to work well for both criteria, although no systematic parameter analysis was undertaken for the PCA fitness criterion. One interesting plot is Figure 5.2.5, which shows an initial population which had a member with very high relative fitness. It demonstrates that the algorithm was able to maintain its diversity, while at the same time discarding worthless information. The evidence for this is the slow but steady rate at which the rest of the population converges towards that exceptional individual, and the fact that the fitness continues to increase once they have reached it.

Figure 5.2.6 is an impressive demonstration of the power of genetic algorithms. In this trial, no information about the jamming environment was given to the program; it was simply directly optimizing over the 13-dimensional space spanned by the weights (14 minus one for the phase constraint.) Given 8000 gain evaluations, it was able to increase the maximum SINR fitness from an initial value of 0.0721 to a final value of 1.274. This final value is comparable to some of the poorer runs of the fully optimized genetic

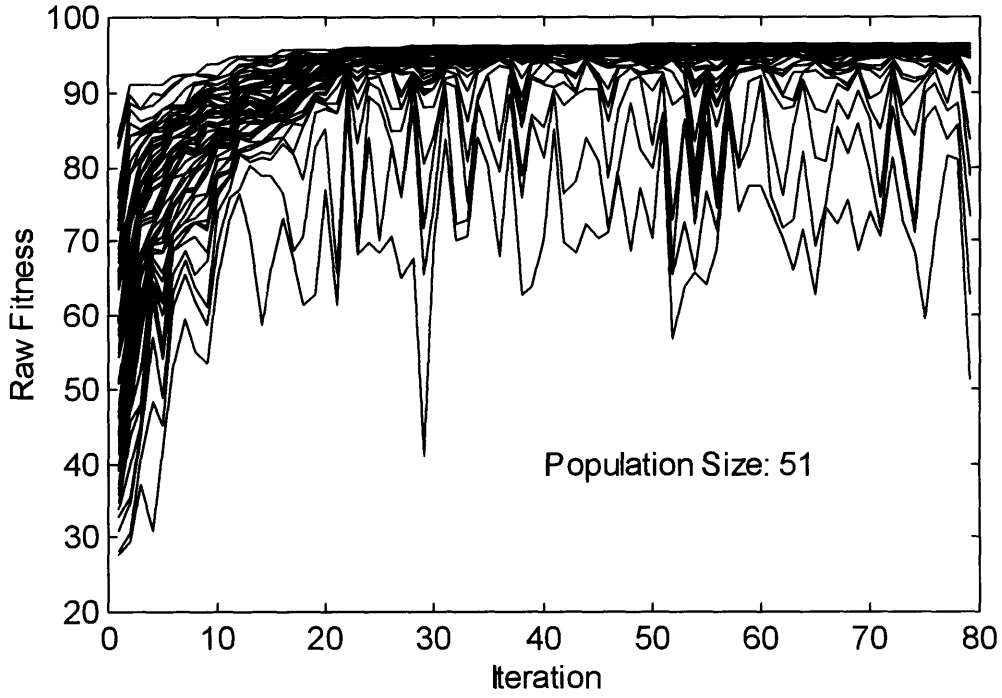


Figure 5.2.1: Convergence of 7 beam antenna for PCA fitness criterion

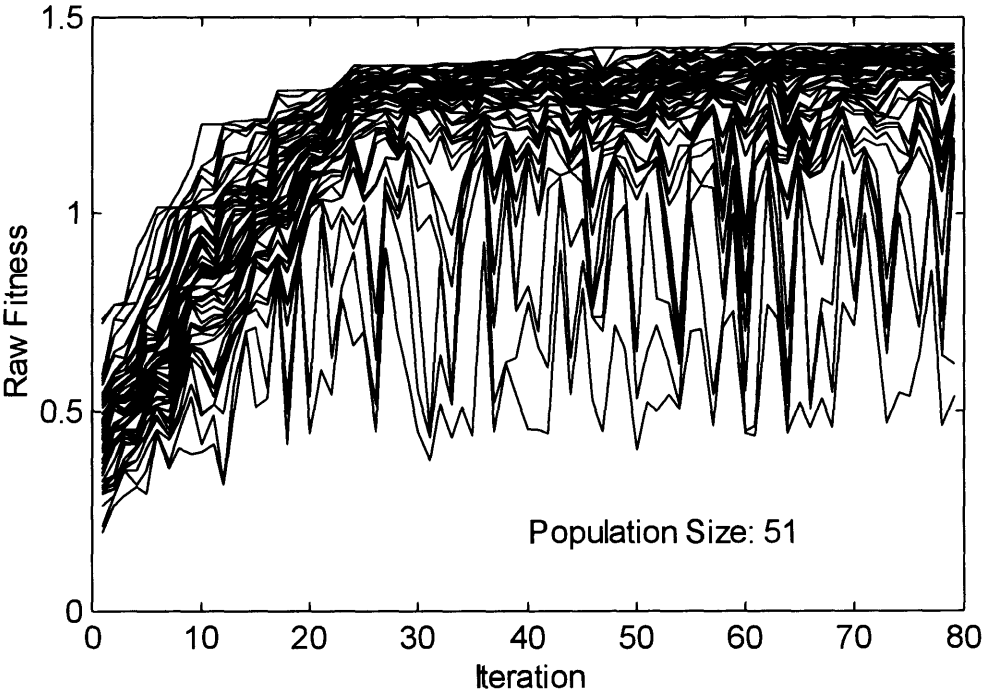


Figure 5.2.2: Convergence of 7 beam antenna for SINR fitness criterion

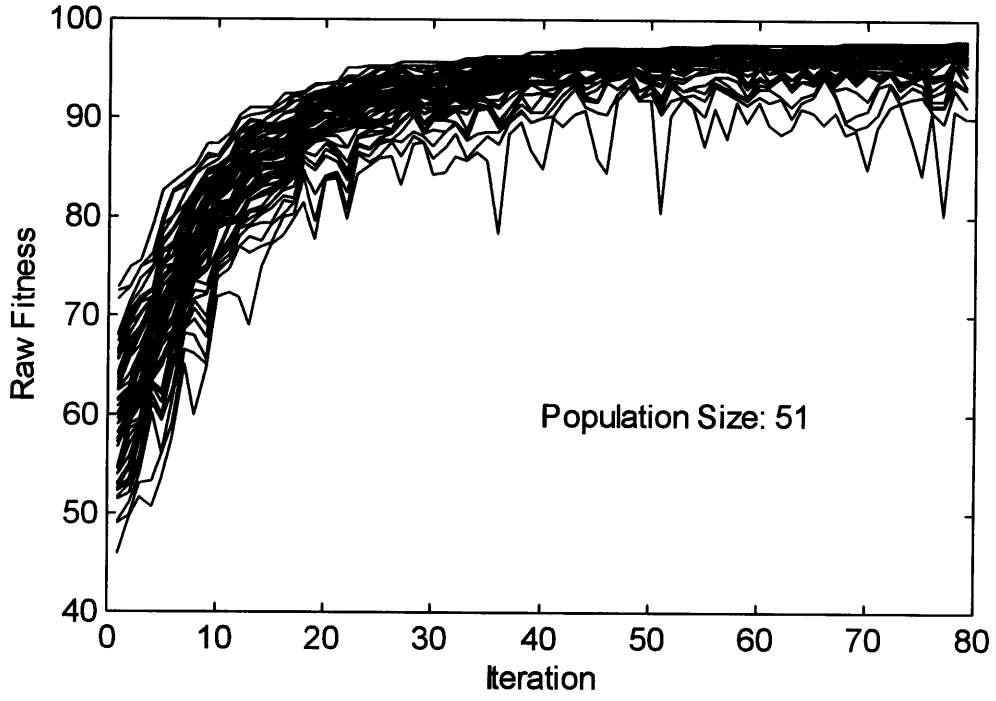


Figure 5.2.3: Convergence of 19 beam antenna for PCA fitness criterion

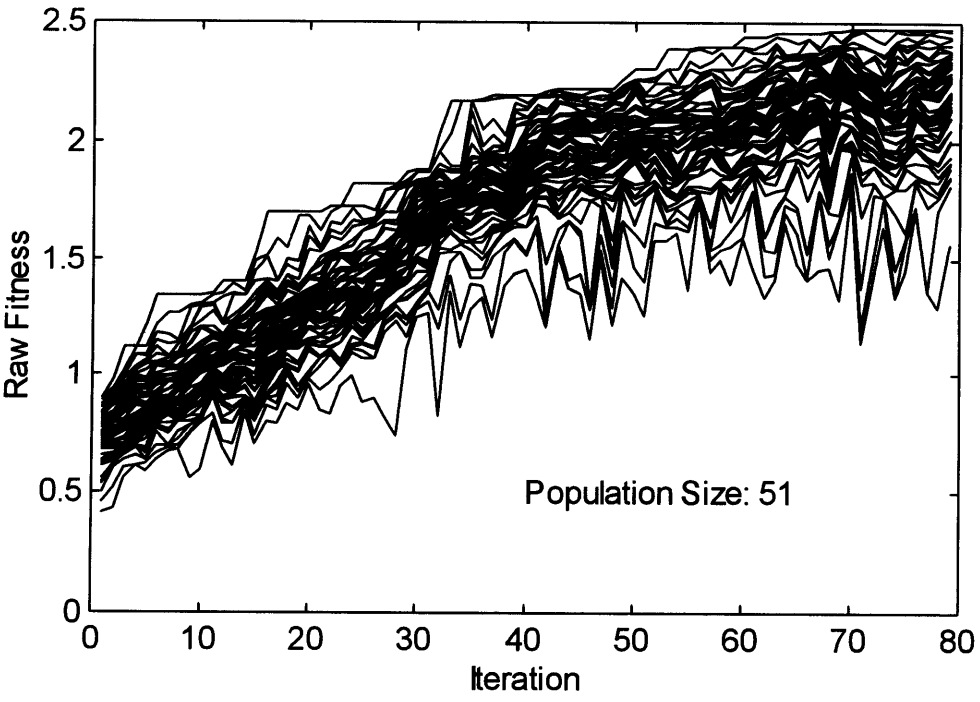


Figure 5.2.4: Convergence of the 19 beam antenna for SINR fitness criterion

algorithm. An initial period of rapid exponential growth, in which the algorithm is probably finding the nulls on the jammers, is followed by a longer period of slower, steady improvement. The contour plot of the converged gain pattern for this run is shown in Figure 5.3.1. It is apparent that it has, indeed, captured the essential elements of a good solution, namely good nulls on the jammers and fairly high, steady gain over much of the rest of the coverage area. It also interesting to note that throughout the optimization process, mutations reduce the fitness of some members to very low values which are comparable to those with which the algorithm commenced. These mutations are apparently destroying the nulls, and markedly dropping the SINR.

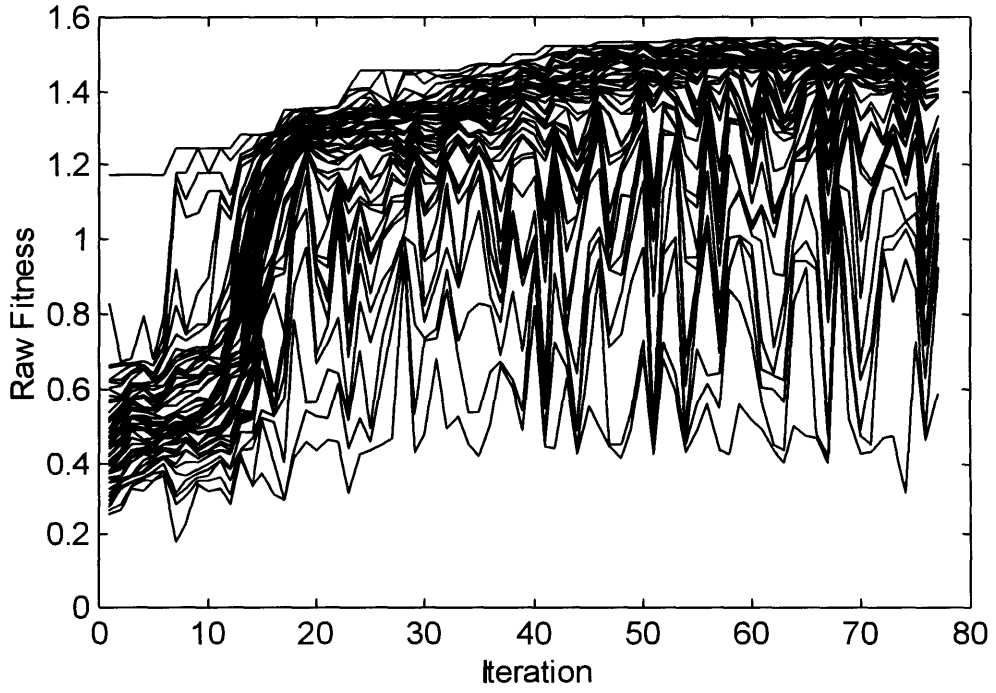


Figure 5.2.5: Convergence history with an initially very fit member

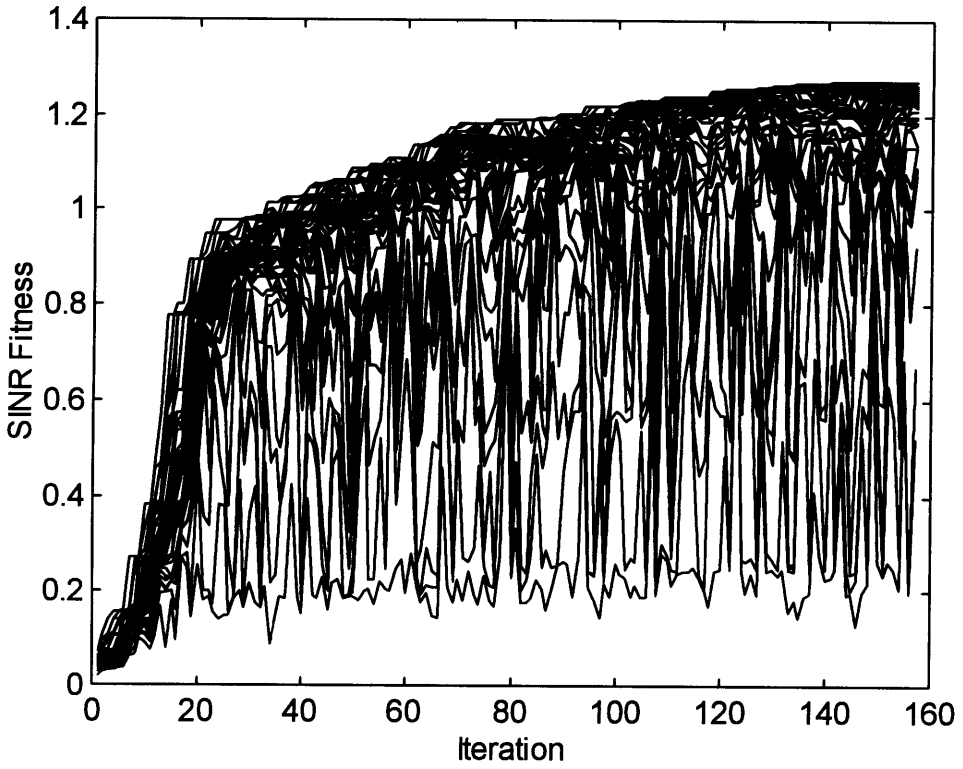


Figure 5.2.6: Unconstrained optimization

5.3 Contour Plots

The contours which are generated by the solution may also provide some valuable insight into how the algorithm is operating. From the figures presented in this section, a variety of the characteristics of the problem can be observed (Figures 5.3.1-5.3.9). In all the plots except 5.3.4 the black regions are the areas of the service area which are below the gain threshold, and the white dots mark the jammer locations.

The most obvious of these, which has already been noted, is that the 19 beam antenna performs noticeably better on both 2 and 3 jammer scenarios. Another interesting feature concerns the gain patterns of antennas which were utilizing the SINR fitness criterion as opposed to those which were using PCA. By taking a side view of the 3-D contour plot (Figures 5.3.8 and 5.3.9), it is clear that the PCA antennas are sacrificing higher maximum gain values in order to keep more of the target area above the threshold.

The SINR antennas, on the other hand, exhibit regions of significantly higher gain at the expense of larger regions which are below the threshold.

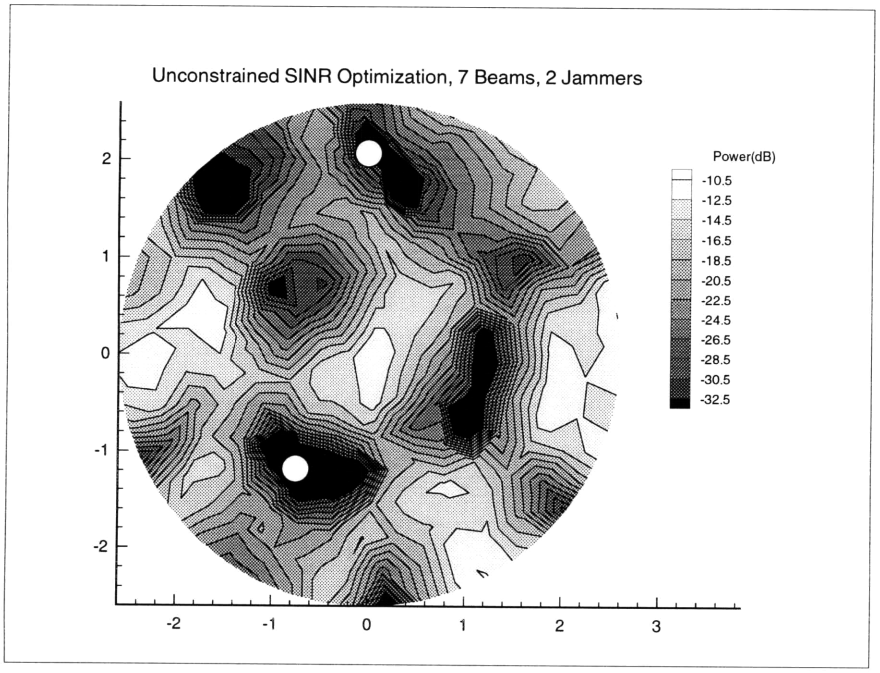


Figure 5.3.1

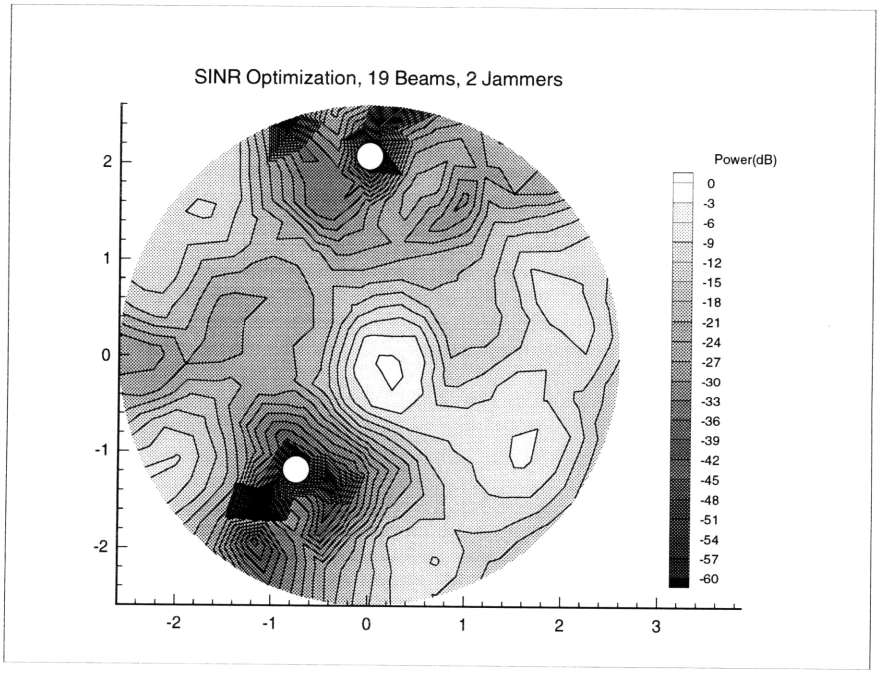


Figure 5.3.2

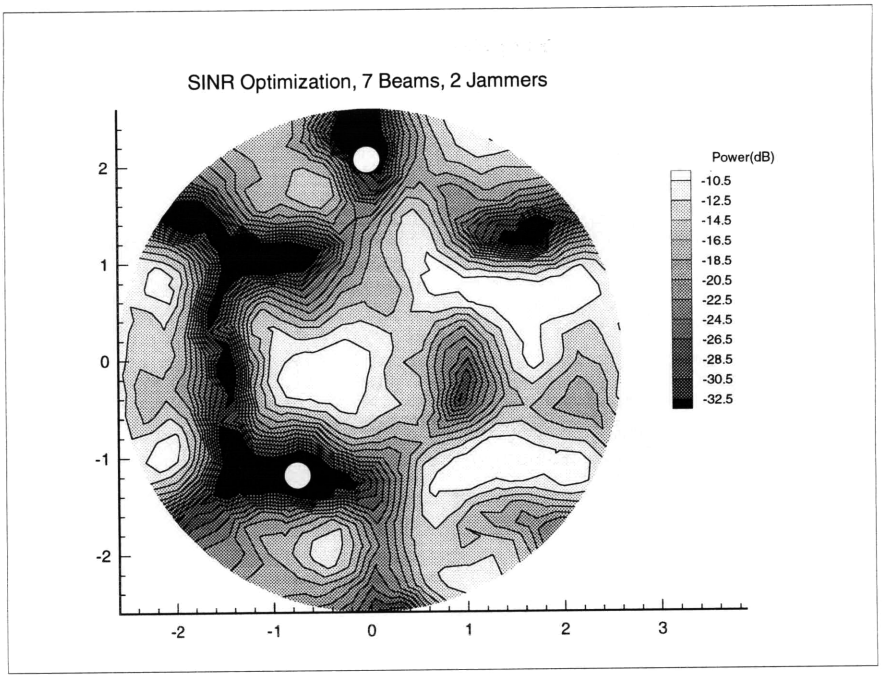


Figure 5.3.3

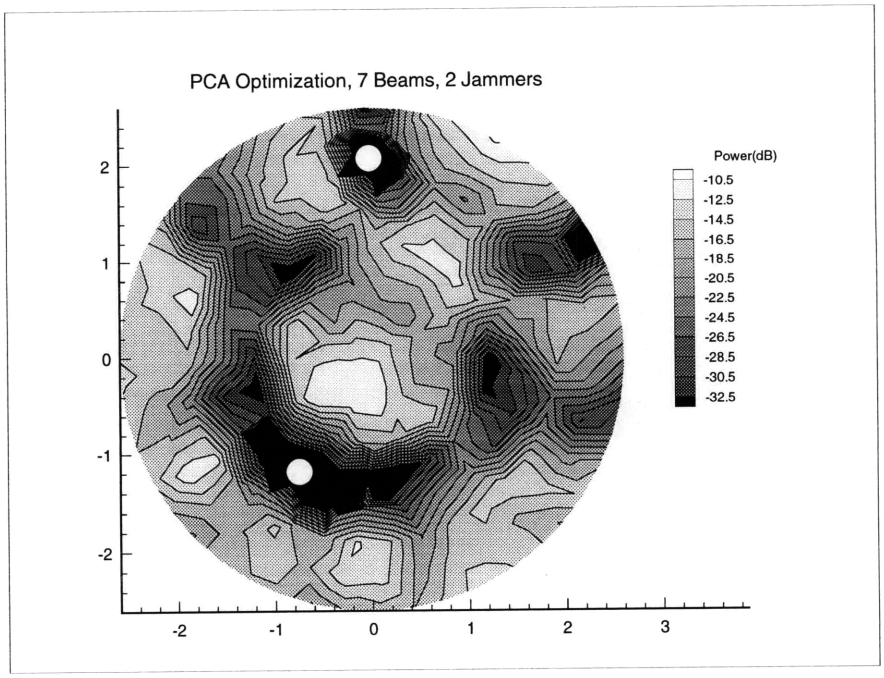


Figure 5.3.4

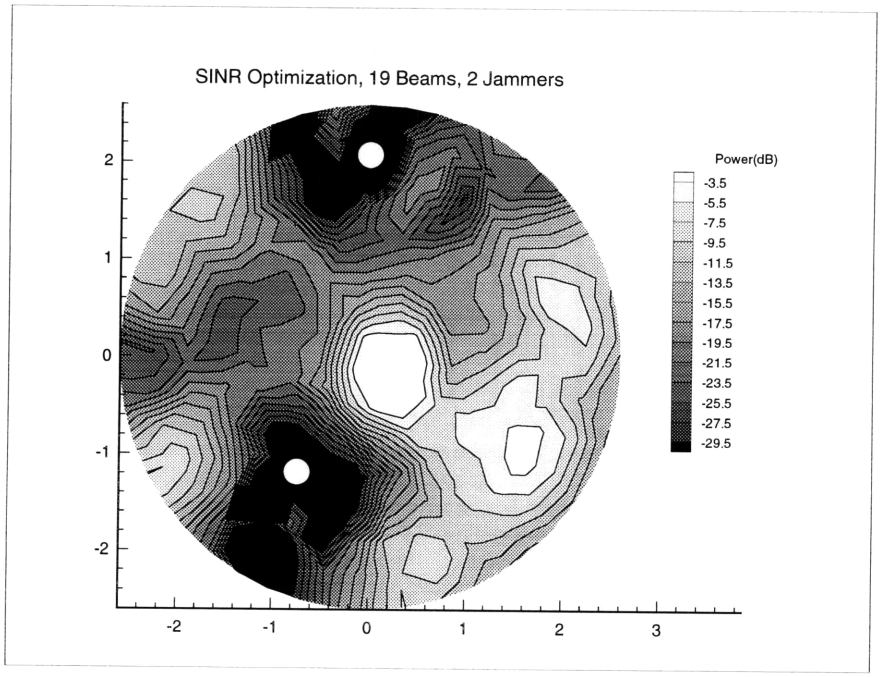


Figure 5.3.5

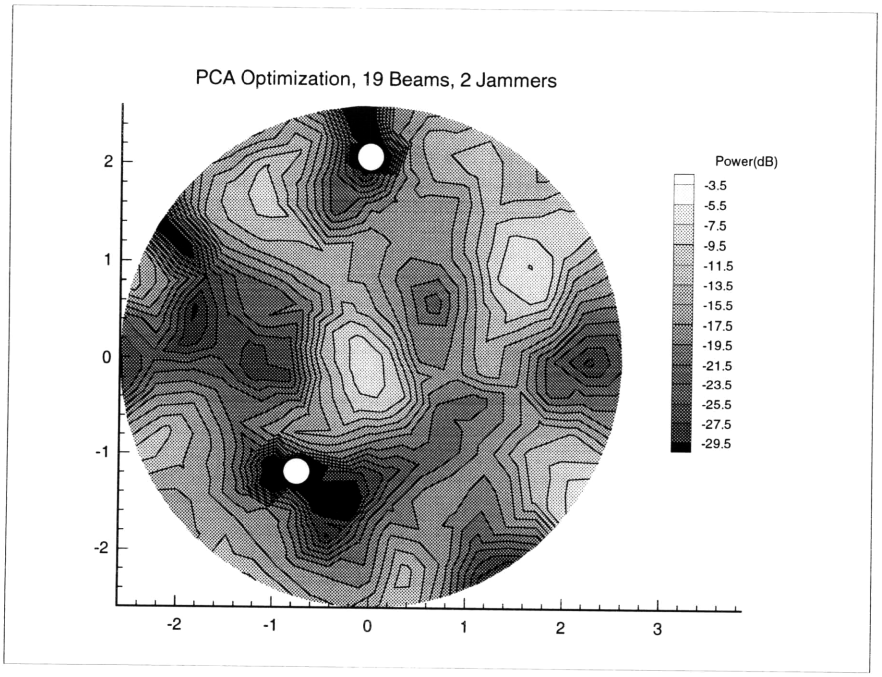


Figure 5.3.6

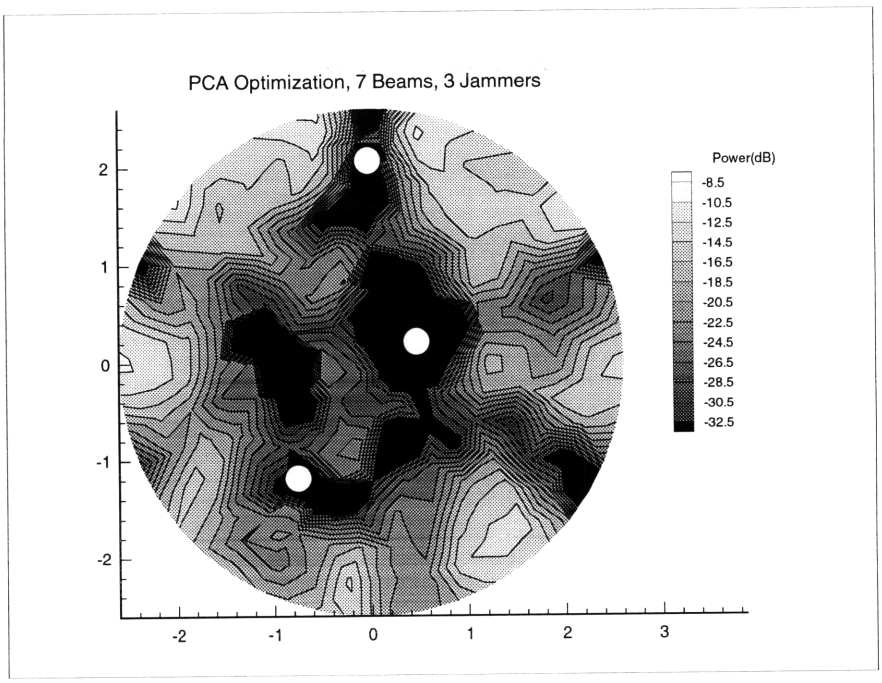


Figure 5.3.7

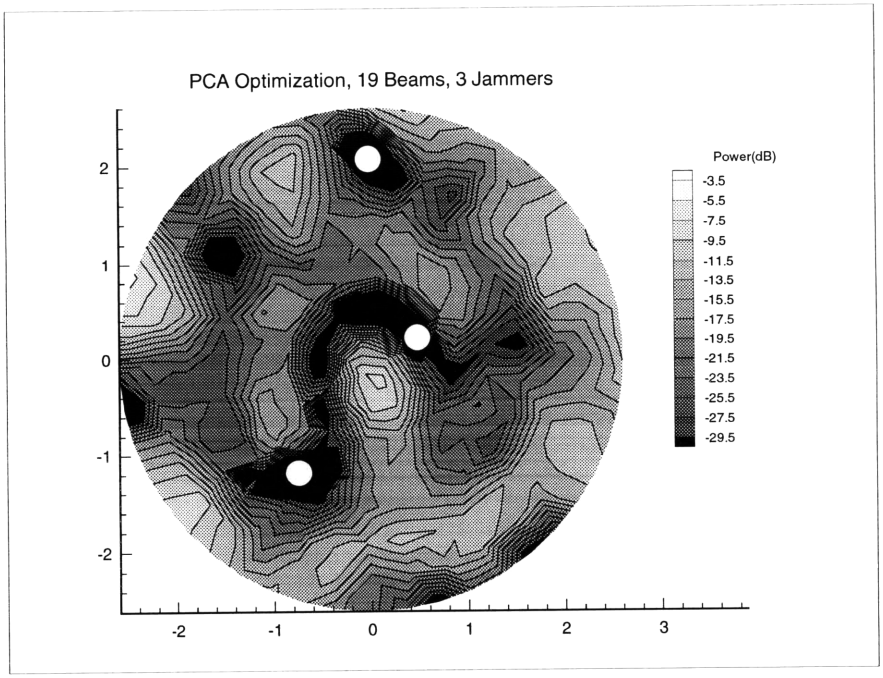


Figure 5.3.8

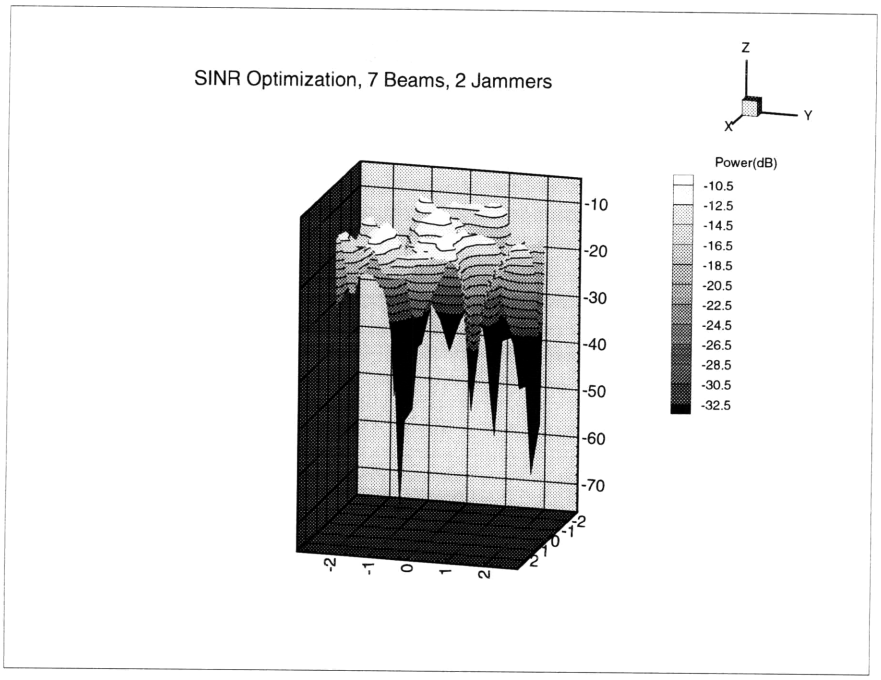


Figure 5.3.9

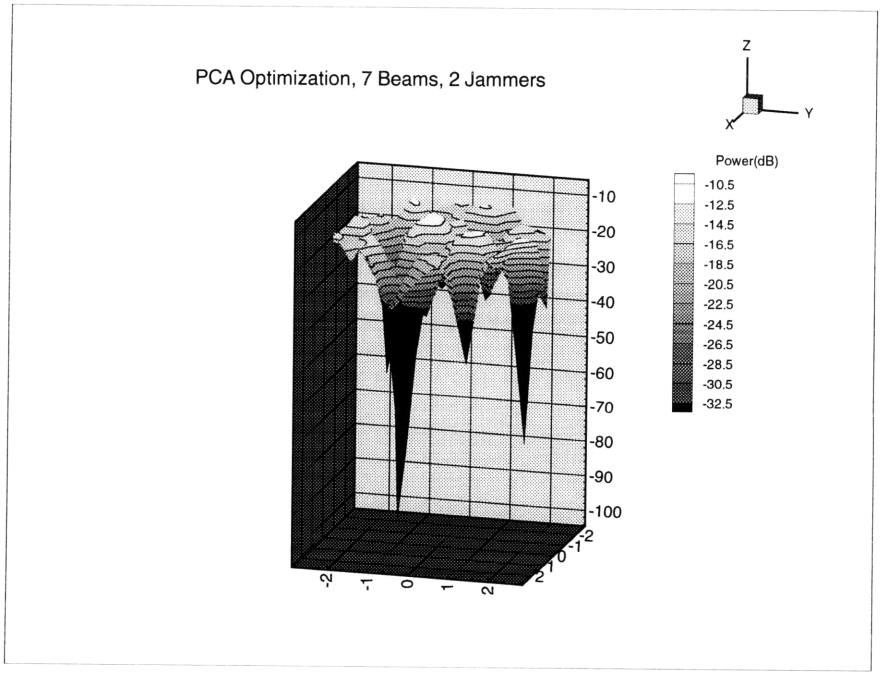


Figure 5.3.10

Chapter 6

Conclusions

6.1 Summary

A genetic algorithm was designed and implemented with the objective of maximizing a variety of optimization criteria for the area-coverage satellite antenna problem. By decomposing the parameter space in accordance with the presence of interference, it was possible to reduce the problem to the optimization of a complicated objective function in about 10 dimensions. A formulation was devised which allowed the algorithm to perform a genetic search over this reduced parameter space, subject to an arbitrary fitness function and a variety of constraints.

The genetic algorithm appears to be well suited to this problem, based on the fitness values which were obtained for both the Percent Coverage Area and Signal-to-Interference-plus-Noise Ratio criteria, and the speed and consistency with which these values were obtained. A reasonable, constrained solution is available immediately due to the S-PACE formulation, and the operation of the algorithm produces steadily improving fitness values which may be expected to converge to a reasonably good solution in less than 1000 fitness evaluations in almost all cases. The genetic algorithm produces fitnesses which are superior to those obtained from the Metropoulis simulated annealing algorithm used previously, although the relatively small sample sizes (about 50 trials) do not establish this result with certainty.

An analysis of the parameters of the algorithm revealed good performance over a fairly wide range of algorithm parameters, and the characteristics of the performance using a wide variety of these parameters was consistent with the efficient operation of genetic algorithms in general.

6.2 Recommendations

This thesis represents a fairly limited trial of the genetic algorithm for the area coverage problem. Several aspects of the problem were not considered, including:

- The interaction of the genetic algorithm with a follow-on nulling stage which could be expected to refine the solutions further
- Non-idealities in the antenna itself and in the jamming environment
- More variable numbers of jammers, and dynamic issues such as the appearance and disappearance of interference sources
- Other beam gain patterns
- The interaction of the algorithm with existing antenna hardware and software

However, from evidence such as the fact that the algorithm was able to produce reasonable results even for a completely unconstrained problem formulation, it seems unlikely that these considerations would severely inhibit the performance of the algorithm.

The genetic algorithm is a very compute-intensive process; however, given the complexity of the problem this is not an unusual characteristic. The problem of parallelizing genetic algorithms is a subject which has been receiving increasing amounts of attention, and methods such as these can effect significant reductions in the run time of the algorithm. As it now stands, 4000 evaluations of a 7 beam, 2 jammer problem using about 240 target points takes about 4 minutes on a DEC 5000. The coding for this project was done in C++, which while being a high-level, flexible language also contains a significant amount of overhead. It is reasonable to expect that optimizations of the algorithm could reduce the running time dramatically. Such an optimization would have to be undertaken if the genetic algorithm were to be a viable alternative for the area coverage problem in real time.

The structure of the algorithm was fairly straightforward, although care had to be taken that the parameters were represented in an efficient manner which satisfied the preconditions of the fundamental theorem of genetic algorithms. Good values for the

algorithm parameters were not difficult to find, however, and the algorithm gave good results as soon as the details of problem representation had been resolved.

Appendix

Tables of Parameter Optimization Test Results

Crossover Rate (%)	Mean Fitness	Std. Deviation
20	1.258	0.1174
40	1.400	0.0810
60	1.438	0.0774
80	1.475	0.0592

Table A.1: Crossover rate test results
Population Size: 51, Mutation rate: 0.5%

Population Size	Mutation Rate (%)	Mean Fitness	Std. Deviation
21	1	1.463	0.0791
21	0.5	1.450	0.1301
21	0.1	1.342	0.1563
51	1	1.520	0.0738
51	0.5	1.492	0.0958
51	0.1	1.406	0.1172
51	0.05	1.303	0.1153
81	1	1.513	0.0647
81	0.5	1.504	0.0701
81	0.1	1.438	0.1012
81	0.05	1.394	0.0933
111	1	1.441	0.0933
111	0.5	1.447	0.0785
111	0.1	1.387	0.0953

Table A.2: Mutation rate and population size test results
Crossover Rate: 80%

Population Size	Mutation Rate (%)	Mean Conv. Time	Std. Deviation
21	0.1	1970	1130
21	0.5	888	703
51	0.1	759	313
51	0.5	559	220
81	0.1	1035	398
81	0.5	923	289

Table A.3: Convergence time test results
Crossover rate: 80%

Table of PCA Fitness Tests

Optimization Method	Number of Beams	Number of Gain Evals.	Mean PCA	Standard Deviation
A-H (None)	7	1	77.3	9.8
A-H (None)	19	1	78.4	6.6
Metropoulis	7	?	92.2	?
Genetic Algorithm	7	4000	94.6	2.4
Genetic Algorithm	19	4000	97.9	0.9
Genetic Algorithm	19	1470	95.7	1.8

Table A.4: PCA results
Crossover rate: 80%, Mutation rate: 0.5%, Population size: 51

Bibliography

1. Baker, James E. *Reducing Bias and Inefficiency in the Selection Algorithm.* Proceedings of the Second International Conference on Genetic Algorithms. Lawrence Erlbaum Associates. pp. 14-21, 1987.
2. Bramlette, Mark F. *Initialization, Mutation and Selection Methods in Genetic Algorithms for Function Optimization.* Proceedings of the Fourth International Conference on Genetic Algorithms. pp. 100-107. Morgan Kaufman, 1991.
3. Compton, R.T. Jr. *Antennas: Concepts and Performance.* Prentice-Hall Inc., 1988.
4. Davis, Lawrence. *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, 1991.
5. Hillis, W. Daniel. *Co-Evolving Parasites Improve Simulated Evolution as an Optimization Procedure.* Physics D, 42:228-234, 1990.
6. Simon, Jay. *Superresolution Post-Adaptive Coverage Enhancement (S-PACE).* MIT Lincoln Laboratory Project Memorandum No. 63PM-Nulling-0002 Revision 1. May 20, 1994.
7. Watson, D.F. *Computing the n-dimensional Delauney Triangulation With Application to Vornoi Prototypes.* Computing Journal, #24 1981.
8. Van Deventer, Paul G. *Flight Control Command Generation in a Real-Time Mission Planning System Using Constrained Genetic Optimization.* Master's Thesis, Massachusetts Institute of Technology, June 1992.