Carnegie Mellon University

CARNEGIE INSTITUTE OF TECHNOLOGY

THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF Doctor of Philosophy

TITLE **Dynamic Beamforming Optimization for Anti-Jamming and Hardware Fault Recovery**

PRESENTED BY **Jonathan Becker**

ACCEPTED BY THE DEPARTMENT OF

Electrical and Computer Engineering

___Jason Lohn_____ ___5/14/14_____
ADVISOR, MAJOR PROFESSOR DATE

___Jelena Kovacevic_____ ____5/14/14_____
DEPARTMENT HEAD DATE

APPROVED BY THE COLLEGE COUNCIL

__Vijayakumar Bhagavatula_____ ___5/14/14_____
DEAN DATE

Dynamic Beamforming Optimization for Anti-Jamming and Hardware Fault Recovery

Submitted in partial fulfillment of the requirements for

the degree of

Doctor of Philosophy

in

Electrical and Computer Engineering

Jonathan M. Becker

B.S., Electrical Engineering, California Polytechnic State University
M.S., Electrical Engineering, University of Southern California

Carnegie Mellon University
Pittsburgh, PA

May, 2014

Abstract

In recent years there has been a rapid increase in the number of wireless devices for both commercial and defense applications. Such unprecedented demand has increased device cost and complexity and also added a strain on the spectrum utilization of wireless communication systems. This thesis addresses these issues, from an antenna system perspective, by developing new techniques to dynamically optimize adaptive beamforming arrays for improved anti-jamming and reliability.

Available frequency spectrum is a scarce resource, and therefore increased interference will occur as the wireless spectrum saturates. To mitigate unintentional interference, or intentional interference from a jamming source, antenna arrays are used to focus electromagnetic energy on a signal of interest while simultaneously minimizing radio frequency energy in directions of interfering signals. The reliability of such arrays, especially in commercial satellite and defense applications, can be addressed by hardware redundancy, but at the expense of increased volume, mass as well as component and design cost.

This thesis proposes the development of new models and optimization algorithms to dynamically adapt beamforming arrays to mitigate interference and increase hardware reliability. The contributions of this research are as follows. First, analytical models are developed and experimental results show that small antenna arrays can

thwart interference using dynamically applied stochastic algorithms. This type of *in-situ* optimization, with an algorithm dynamically optimizing a beamformer to thwart interference sources with unknown positions, inside of an anechoic chamber has not been done before to our knowledge. Second, it is shown that these algorithms can recover from hardware failures and localized faults in the array. Experiments were performed with a proof-of-concept four-antenna array. This is the first hardware demonstration showing an antenna array with live hardware fault recovery that is adapted by stochastic algorithms in an anechoic chamber. We also compare multiple stochastic algorithms in performing both anti-jamming and hardware fault recovery. Third, we show that stochastic algorithms can be used to continuously track and mitigate interfering signals that continuously move in an additive white Gaussian noise wireless channel.

Acknowledgments

It has been said that it takes a village to raise a child. From my experiences, it takes a department to create a successful PhD. I would like to thank my academic advisor, Professor Jason Lohn, who guided me through the process of earning my PhD. I would also like to thank my committee members from Silicon Valley, Professors Patrick Tague and Ole Mengshoel, whose advice helped me complete my thesis. I would also like to thank my external committee member, Dr. Derek Linden, for his advice and inspiration. I am appreciative that Dr. Linden allowed me to use his AntNet software with my antenna array optimization simulations with WIPL-D and for his tips on properly using both software packages.

In addition, my research was funded in part by Cylab at Carnegie Mellon University under grant DAAD19-02-1-0389 from the Army Research Office, the Navy, and by the Electrical and Computer Engineering Department at Carnegie Mellon University. I would like to thank these organizations for their generosity and for their support.

Of course, I also thank several other professors that I met during my studies and research at CMU: Professor Martin Griss and Professor Bob Iannucci for their support as the Silicon Valley Campus chairs, Professor Ozan Tonguz for his guidance during my preparation for the qualifying exams, Professor Emeritus James Hoburg for the opportunity to be a teaching assistant for his electro- and magneto-quasi stat-

ics undergraduate course and his support in helping me prepare for my qualifying exams, Professor James Bain for allowing me to be a teaching assistant for his electromagnetics waves course, Professors David Ricketts and David Greve for having me as their teaching assistants for their RF/wireless capstone courses, and Professor Vijayakumar Bhagavatula for teaching the undergraduate noisy signals representation course that was fundamental to and greatly helped me in passing my second Qualifying Exam. Without your support and advice, I would not have been successful in earning my PhD.

Furthermore, I would like to thank my friends that I made at CMU. I thank Dr. James Downey for his advice and humor both before and after he completed his PhD, Professor Joshua Griffin for his support and advice during and after my graduate internship at Disney Research Pittsburgh (DRP), Dr. Matthew Trotter for his support and advice during and after my time at DRP, Dr. Joseph Fernandez for his friendship and advice throughout my PhD, Dr. Reginald Cooper for his friendship and support, Jon Smereka for his friendship and well branded humor, Dr. Joel Harley for his guidance and friendship as a fellow EGO officer, Dr. Shahriyar Amini for encouraging me to be more active in EGO as its Secretary, and Matthew Beckler for his years of service as EGO secretary and support in guiding me into following his footsteps as an EGO secretary.

Finally, I would also like to thank my family who supported me along the way. First and foremost, I send thanks to my girlfriend, Heather Mallet, who put up with

my long hours dedicated to completing my PhD and the inevitable stress that we both endured together. Her wisdom helped me through the stress because she knew that my hard work would pay off in the end. I would also like to thank her parents, Herbert and Adrienne Mallet, as well as her brother and sister-in-law, Matthew and Laine-Guttman Mallet, who gave their support and guidance while I worked hard towards graduation. I am truly appreciative that you all support me and want to see me succeed. I also thank my family for their support and guidance: My parents Tully and Frances Davis Becker, my brother George, my sister Hannah, and my brother-in-law Brent Bremer. Mom, I wish that you could have been here to see me complete my PhD. However, I know that you are in a better place, and you are very proud of what I have accomplished.

Contents

List of Tables

List of Figures

List of Acronyms

AJBF	Anti-Jamming Beamforming
AF	Array Factor
AUT	Array Under Test
BER	Bit Error Rate
CCA	Concentric Circular Array
CGM	Conjugate Gradient Method
DOA	Direction of Arrival
EM	Electromagnetic
EF	Element Factor
GA	Genetic Algorithm
HCA	Hill Climbing Algorithm
HR	Hardware Redundnacy
LMS	Least Mean Squares
LOS	Line of Sight
MIMO	Multiple Input Multiple Output
MMSE	Minimum Mean Square Error
MTBF	Mean Time Between Failure
MTTR	Mean Time To Repair
MOM	Method of Moments
MVDR	Minimum Variance Distortionless Response
NLOS	Non Line of Sight
OFDM	Orthogonal Frequency Division Multiplexing
PSLL	Peak Side Lobe Level
PAA	Phased Antenna Array
PDF	Probability Density Function

PN	Pseudo-Random Noise
POC	Proof of Concept
RSGA	Rejuvenated Simple Genetic Algorithm
SLL	Side Lobe Level
SNR	Signal to Noise Ratio
SOI	Signal of Interest
SGA	Simple Genetic Algorithm
SINR	Signal to Interference and Noise Ratio
SIR	Signal to Interference Ratio
SA	Simulated Annealing
SOA	State of the art
TDGA	Triallelic Diploid Genetic Algorithm
ULA	Uniform Linear Array
UCA	Uniform Circular Array
VNA	Vector Network Analyzer

Chapter 1

Introduction

In recent years there has been a rapid increase in the number of wireless devices for both commercial and defense applications. This has been adding strain on the spectrum utilization of wireless communication systems. Because there exists a limited amount of available frequency spectrum, interference is bound to occur as the spectrum saturates [1]. In the military realm, adversaries jam signals used to guide military equipment including smart bombs and unmanned aerial drones. As a countermeasure to unintentional interference and adversarial jamming, antenna arrays are used in wireless communications to focus electromagnetic energy on a signal of interest (SOI) while simultaneously minimizing energy in jammer directions. However, sophisticated antenna arrays cost millions of dollars and are designed for defense applications that are not suitable for lightweight drones or consumer mobile devices. These systems utilized standard optimization techniques, generally gradient-based algorithms, and consist of tens to hundreds of antenna elements rendering them unsuitable for consumer wireless communication systems.

Much of the hardware incorporated into sophisticated antenna arrays are used to implement gradient-based algorithms for optimizing the array. A goal of this thesis is to show that it is possible to shift hardware based optimization into software by using optimization algorithms. It is also possible to achieve anti-jamming beamforming for commercial wireless applications with fewer antennas compared to beamforming systems used for defense applications. By shifting optimization to software and using fewer antennas, the cost of anti-jamming beamforming systems can be reduced from millions of dollars to thousands of dollars making them more feasible for commercial applications. In addition, this shift from hardware to software also reduces the physical size of the array which is also desirable for commercial applications.

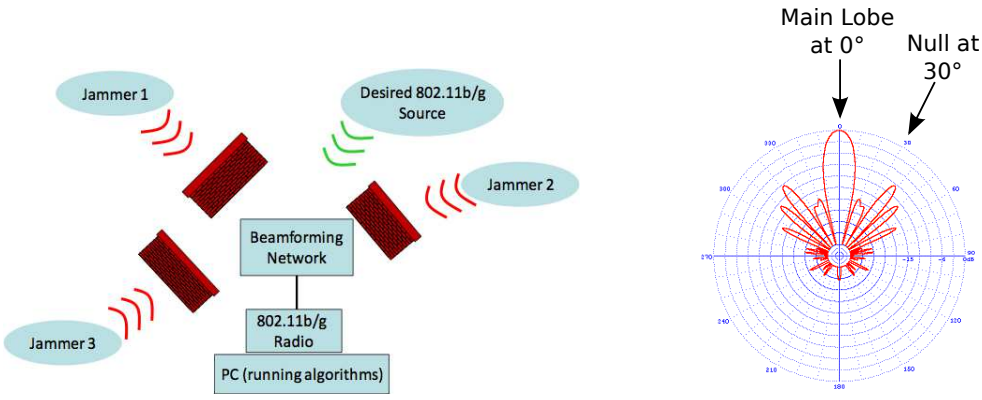
In addition, many optimization algorithms applied to beamforming arrays assume that the signals do not change direction. If the signal directions can be determined, the algorithms used to optimize these arrays are run once and the optimized settings are used indefinitely [2]. This is often an invalid assumption, as many wireless devices are mobile with time-varying directions of arrivals (TVDOAs) at the antenna array. Furthermore, the wireless channel can also vary with time. Objects, such as people and vehicles, can momentarily move between the transmitters and the receiving beamformer thereby causing temporary fades in the received signals. The optimization algorithm needs to quickly adapt to these signals and to changing environment conditions, or anti-jamming beamforming will be difficult to obtain.

The reliability of such arrays, especially in commercial satellite and defense appli-

cations, can be addressed by hardware redundancy, but at the expense of additional volume and mass with a greater component and design cost. Commercial systems are often limited by cost and available space, so hardware redundancy is not a viable option. Some defense systems, such as Unmanned Aerial Vehicles (UAVs) are limited by volume and mass, so hardware redundancy is unfeasible as well. In both cases, anti-jamming functionality is lost if a hardware component faults. Redundancy must therefore be obtained via software techniques that can detect and re-optimize the antenna array for fault recovery. This thesis addresses these issues, from an antenna system perspective, by developing new techniques to dynamically optimize adaptive beamforming arrays for anti-jamming and reliability.

A high-level overview of a system of three jamming devices is depicted in Figure 1.1a. The system uses Stochastic Algorithms to perform anti-jamming and hardware fault recovery. The research in this thesis focuses primarily on the Genetic Algorithm (GA) – a population-based stochastic search technique built around the concepts of survival of the fittest mate selection, chromosomal crossovers, and bitwise mutation. The GA's goal is to maximize a fitness function (i.e., Signal to Interference and Noise Ratio, SINR). The performance of the GA is compared to the Simulated Annealing (SA) algorithm and the Hill Climbing Algorithm (HCA). The algorithms operate on a computer as depicted in Figure 1.1a.

Beamforming is an important technique in modern RF systems. In most communication based applications, a transmitting system directs RF energy at an intended



(a) (b)

Figure 1.1: Beamforming Examples: (a) Overview of an Anti-jamming System, (b) An example of beamforming with main lobe at 0°and a null at 30°.

receiving antenna system. In other applications, the receiving antenna system steers nulls (deep negative decibel gains) towards jammers. A beamforming network is comprised of radio frequency (RF) or microwave circuitry that fed each antenna element in an antenna array with an external device that sets the complex array weights (amplitudes and phases) to obtain a desired radiation pattern. Figure 1.1b shows an example of beamforming in which a null is directed at 30 degrees. The main lobe represents the direction where incoming signals are received, whereas very little signal is received in the null directions (i.e., at 30 degrees). An anti-jamming beamforming array is similar to a Phased Antenna Array (PAA) except the PAA actively adjusts phases to form a mainbeam in a desired direction with low sidelobes in relation to the mainlobe. Prior to steering the beam, PAAs set weight amplitudes to create the desired sidelobe levels (SLLs) in relation to the mainbeam. The weight amplitudes are not modified while the array's mainbeam is steered.

1.1 Limitations of Current Phased Array Technology

Current phased array technology is developed for antenna arrays with dozens to hundreds of antennas. These beamforming arrays are phased such that they focuse a narrow beam pointed in the direction of a known SOI using radiation pattern mask that guarantees that the sidelobe levels are below a certain level with respect to the SOI (i.e., $\leq$ -20 dB) [3,4]. They are typically designed for military use (such as radar arrays) and are large and typically too expensive for commercial wireless applications.

Given known SOIs, it can be shown that at least 50 antennas would be needed to achieve SLLs of at least -20dB with a Uniform Linear Array (ULA) with linear phased weights [4]. This SLL is sufficient to prevent interference from jamming the SOI because the output interference power will be less than one-hundredth of its input SOI power level. However, these SLLs are not achievable for small antenna arrays. For example, Manteghi *et al.* [5] developed a low-cost antenna array and shows that a four-antenna linear array had SLLs of -12 dB with respect to SOI. A 64 element linear achieves SLLs greater than -20 dB with respect to SOI.

Although Bevelacqua [6] shows that a six element linear array with Dolph-Chebyshev phase weights can achieve SLLs of -30 dB,[1] an issue with linear arrays is that they have a grating lobe opposite the main lobe. This is not desirable, as the array will not block interference that arrives 180° with respect to the SOI. The

[1] Bevelacqua did not calculate the minimum achievable SLL for a four element linear array using Dolph-Chebyshev phase weights.

practice is to place a ground plane near the linear array to block EM radiation behind the array [4]. However, this ground plane adds weight to and increases the size of the array because the ground plane must be several wavelengths larger than the array in both the X and Y directions.[2]

Unlike linear arrays, circular arrays have an advantage in that they possess symmetry in the azimuth plane [4]. Although Belevacqua [6] optimizes the geometry of hexagonal arrays[3] and other planar array configurations, that method is unable to optimize planar arrays for SLLs less than -20 dB for all evaluated test cases. It can also be noted that the analysis is based on array factor calculations, so they do not include mutual coupling between elements which tend to increase SLLs.

The above beamforming methods assume that the SOI and interference directions are known *a priori*. However, it is not always possible to know SOI and interference DOAs ahead of time. For example, Zhang *et al.* [8] notes that GPS systems operate at a cold start without any prior knowledge of its position and orientation, so the system must determine the SOI's DOA at startup. However, the GPS system can be jammed at startup, and the jammers need to be suppressed before the system can locate the SOI. Adaptive algorithms are needed to steer nulls in the jammers' directions while simultaneously focusing EM energy towards the SOI. However, the classical Least Means Squares (LMS) algorithm is known to have slow convergence

[2]The ground plane theoretically halves the input impedance of the array as noted through the use of image theory [7].

[3]Hexagonal arrays can be considered to be a limiting form of the circular array with a maximum of seven elements where the seventh element was placed in the array's center.

with poor multimodal performance [9]. Although the Conjugate Gradient Method (CGM) converges much faster than LMS, SOI and jammer directions are needed for it to optimize the array pattern for anti-jamming [9].

1.2 Importance of Anti-Jamming Beamforming with Hardware Fault Recovery

Hardware fault recovery is important in anti-jamming beamforming arrays because they typically operate in the field as part of a wireless network and can not be easily repaired. Figure 1.2 shows a hypothetical situation where an anti-jamming beamforming array has been optimized to protect a wireless Wide Array Network (WAN) that connects to several Local Area Networks (LANs) as well as mobile infrastructure (i.e., buses, ambulances, automobiles, etc.). This WAN is part of a larger terrestrial wireless communications network that connects a central uplink transmitter to multiple WANs and aircraft via a communications satellite.

If the anti-jamming beamforming array can not recover from hardware faults with some form of hardware fault tolerance, a hardware fault will cause the array to fail in the sense that it no longer protects the wireless WAN. The jammer prevents the central uplink transmitter from communicating with that WAN's multiple wireless LANs that in turn no longer forwards broadcast information to their respective users and wireless sensor networks (Figure 1.3). Chiti *et al.* [10] notes that fault tolerance is important when multiple wireless systems are interconnected in an ad-hoc network aimed at providing disaster management over a wide area. Failures within any part

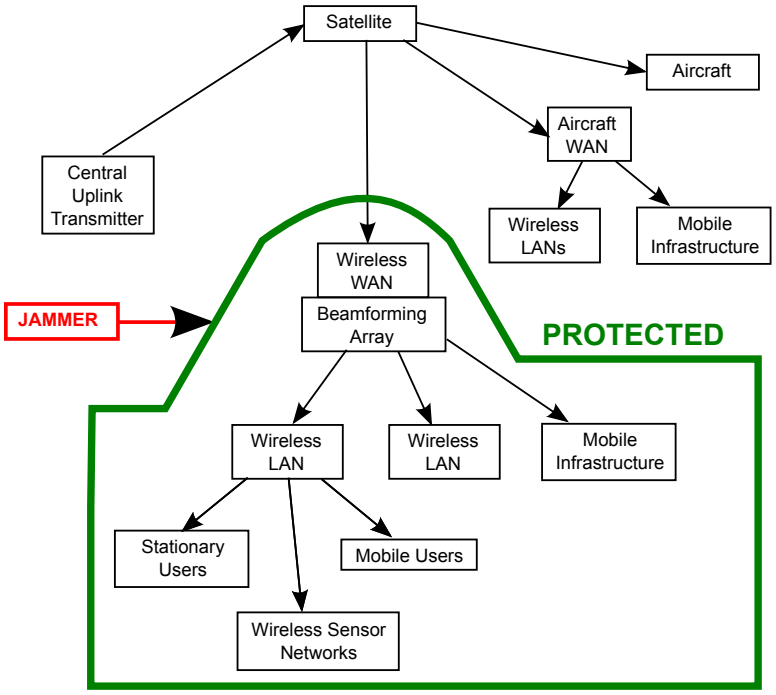


Figure 1.2: A wireless WAN section of a hypothetical terrestrial wireless communications network protected by an anti-jamming beamforming array.

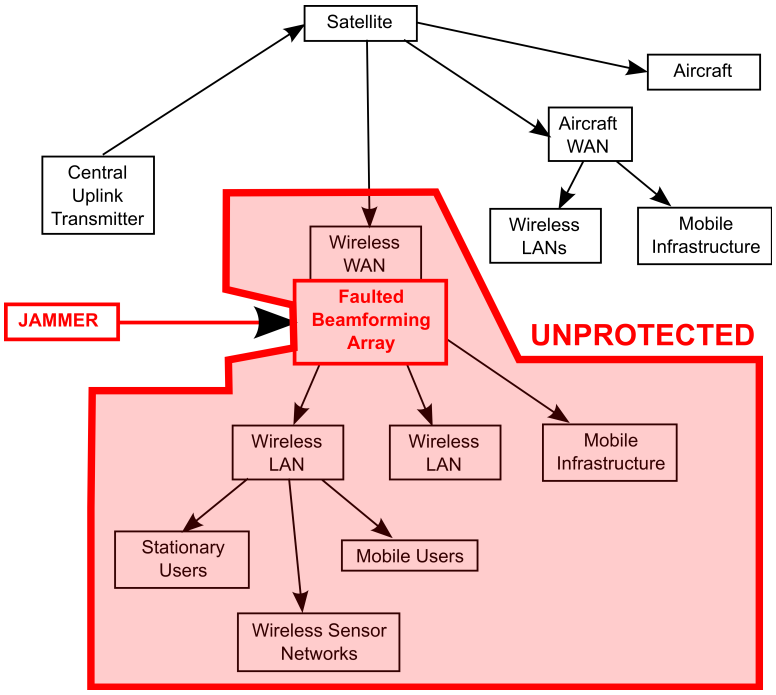


Figure 1.3: The wireless WAN section of a hypothetical terrestrial wireless communications network unprotected and jammed due to a faulty beamforming array.

of the network prevent vital information from being communicated between various first-responders, governmental agencies, and concerned peoples within the network's reach. They also observe that the ability to detect inter-network failures is important not only to implement fault tolerance, but it is also important to ensure people's safety within the network's coverage area. Khatib [11] describes fault tolerance issues in a military terrestrial network similar to the one illustrated in Figure 1.2.

As in the civilian case previously discussed, a faulty PAA in a long-range UAV can prevent communication of vital commands from reaching military personnel on the ground. Rabbath and Léchevin commented in [12] that an information flow fault (i.e., "a temporary or permanent loss of information between two or more UAVs, or between UAVs and the operating crew") can be caused by communication failures due to situations including jamming and TX/RX failures. As such, ground personnel cannot submit tactical reports to the central receiver (co-located with the central uplink transmitter) because the enemy jams their communications because the long-range UAV's PAA failed and did not recover functionality. Clearly, it would be of vital importance to detect and recover from faults in that PAA to regain anti-jamming functionality.

Possible situations where beamforming array failures can cause serious issues are listed in Table 1.1. The motivation for hardware fault recovery in antenna arrays for long-range military communication and civilian disaster management is clear [10, 11]. For the case of arrays used with MTI radars, Klemm notes that if array errors are

known *a priori*, they can be incorporated into the beamforming weights. However, if the errors are unknown (as would be the case for an antenna element fault), the beam would be skewed away from the target, and its clutter performance will suffer [13]. It is possible that the interference null performance would suffer as well because unknown array errors detuned the array's radiation pattern causing increased SLLs.

Table 1.1: Situations where beamforming array failures cause serious issues

Situation	**Effects of Array Failure**
Long-range military communication	Trickle down loss of communication in chain of command with decreased safety and possible loss of life in the field
Civilian Disaster / Emergency Management	First responders' safety at risk due to lack of situation awareness along with confusion in and out of disaster arena due to non-dispersion of information
Airborne Moving Target Indicator (MTI) Radars	Degradation of performance due to poor clutter and interference rejection
Consumer Mobile Communication	Degradation of cell performance and increased cost from sending field engineers to perform repairs
Spaceborne Communications	Early termination of mission due to array fault induced communication system failure
Aircraft & Weather Surveillance Radar	Increased risk of aircraft collisions with loss of life and delayed passenger traffic

Although a single antenna element fault is unlikely to cause severe issues in a large array with hundreds or thousands of elements, an array failure is likely to occur when a significant portion (i.e., 10%) of the elements are at fault [14]. Agrawal *et al.* [15] notes that the purpose of redundancy in beamforming arrays was to increase the array's Mean Time Between Failure (MTBF) beyond that of its individual components. They

discuss redundancy techniques to increase system MTBF for an 8,000 element circular antenna array. A failed array is described as one that had a 3 dB increase in SLL. Antenna elements include Transmit/Receive Modules (TRMs), control modules, and power supplies. A fault in any of those subsystems can cause an antenna element to fail. Out of the 8000 elements, their analysis shows that it took 3.2% (or 256 out of 8000) element failures to cause an array failure when the each antenna had its own TRM, control module, and power supply. The above example indicates that hardware fault recovery is important for array applications that typically use large arrays such as satellites and aircraft and weather surveillance radar.

As the size of the platform decreases, the available space for an antenna array decreases making it infeasible to incorporate an array with hundreds or thousands of elements. Even in military applications that typically use large arrays, small UAVs can have their own PAAs [11]. Such arrays are limited to dozens of antennas, so fault recovery is more critical.

1.3 Motivation for Implementing Array Redundancy via Algorithms

Since previous antenna arrays operate on large platforms and are supported by government funds, it is feasible to build large arrays with hardware redundancy. Hardware redundancy allows arrays to operate beyond individual component lifespans, as standby subsystems would takeover when primary subsystems fail. However, commercial wireless systems are both space and funding limited, so it is not feasible to

build large arrays with hardware redundancy. Redundancy must therefore be placed in software algorithms that can detect and recover from hardware failures to regain anti-jamming performance.

As discussed above, an array failure due to a hardware fault can cause a ripple effect in the sense that wireless communication systems that connect to a beamformer could cease functioning if the beamformer faults and fails to thwart interference. This is critical because wireless systems can be connected indirectly to the faulty beamformer protected system via wireless links of their own. The fault recovery algorithms must detect faults and recover anti-jamming beamforming functionality.

These algorithms should localize faults, as fault detection is necessary for future repair or replacement of the system by a field engineer. A primary motivator of fault tolerance is to increase the Mean Time Between Failure (MTBF) of a system beyond that of the MTBF of its components given that the Mean Time to Repair (MTTR) was greater than the component MTBF [16]. Although fault localization is not critical as fault tolerance, fault localization is still important because it decreases system repair time, and downtime reduction increases the system's availability.

A high-level flowchart showing how a hardware fault detection, recovery, and localization algorithm could be incorporated into a system is shown in Figure 1.4. An assumption is generally made that the system starts with normal operation, a hardware fault causes a system failure, the algorithm detects a failure, and the algorithm recovers system functionality at least partially. However, this assumption that faults

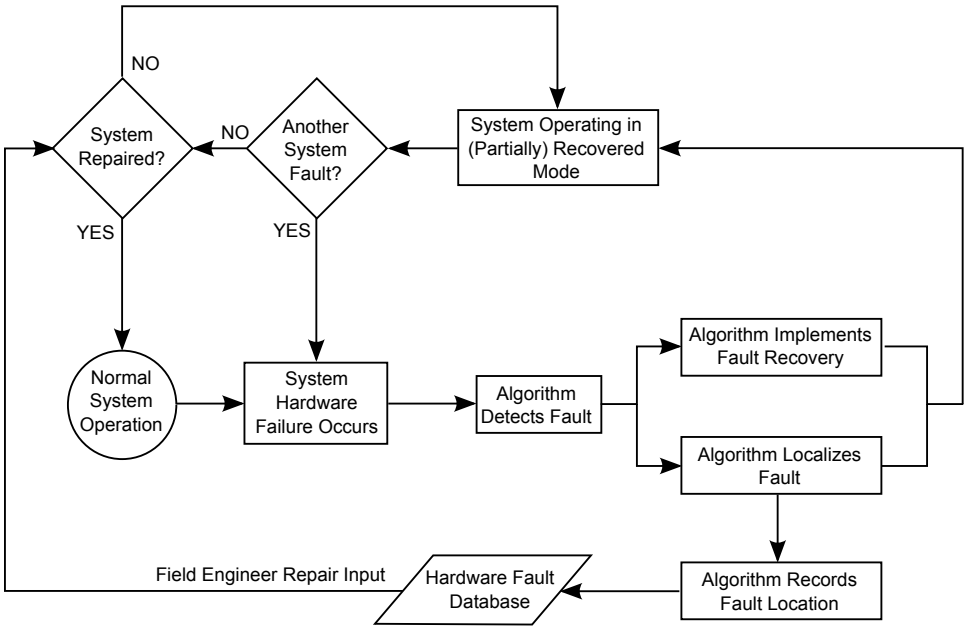


Figure 1.4: A high-level flowchart showing hardware fault detection, localization, and recovery.

cause system failures is not always valid [16]. For example, an antenna fault would not cause a beamformering failure (i.e., loss of anti-jamming) if the antenna element does not have a significant impact on the array's radiation pattern prior to the fault.

It is not always possible for the algorithm to recover anti-jamming functionality after the occurrence of each fault. A general rule of thumb in anti-jamming beamforming arrays is that a normally operating array with N antennas can anti-jam at most $N - 1$ jamming signals (where the N^{th} signal is the SOI). If one SOI and $N - 3$ jammers are present, and three antenna elements become faulty, it is impossible for any fault recovery algorithm to recover anti-jamming functionality. The three faults effectively reduce the array size from N elements to $N - 3$ elements, so there are more signals than normally operating antenna elements.

Because of the critical nature of hardware failures in general, it is desirable to

maximize the fault detection probability. It is also possible for the algorithm to give false alarms and detect faults when anti-jamming failures do not occur. Although false alarms are not as critical as missed faults, the algorithm should minimize the probability of false alarms. If too many false alarms occur, the end users will not trust the algorithm's performance, and correct detection of faults will be ignored.

1.4 Stochastic Algorithms for Anti-Jamming Beamforming and Hardware Fault Recovery

This thesis compares multiple stochastic algorithms in performing both anti-jamming and hardware fault recovery. Stochastic algorithms use random variations guided by fitness functions to find optimal solutions in large, multimodal search spaces. In anti-jamming beamforming applications with hardware fault recovery, the fitness function to be maximized is the Signal to Interference and Noise Ratio (SINR). Because the Genetic Algorithm (GA) and the Simulated Annealing (SA) algorithm are investigated in performing anti-jamming beamforming and hardware fault recovery, those algorithms will be discussed below. The Hill Climbing Algorithm (HCA) is a special case of SA.

1.4.1 Genetic Algorithms in Anti-Jamming Beamforming with Hardware Fault Recovery

GAs are useful for solving combinatorial optimization problems with multimodal landscapes. GAs have been used to solve electromagnetic problems including antenna design [17–20], phased antenna array synthesis [21], and array synthesis recovery fol-

lowing occurrence of element faults [22, 23]. The GA is a stochastic search algorithm based on three concepts found in nature: survival of the fittest mate selection, chromosomal crossovers, and chromosomal bitwise mutations. There are additional operators that the GA can use, but a GA that operates only on these three operators is known as a Simple Genetic Algorithm (SGA) [24–26]. The GA maximizes a fitness function chosen to represent the problem at hand. As will be shown in Chapter 5, the problem of optimizing complex array weights to focus energy on a SOI and null multiple jammers after a fault occurred is a combinatorial one with a multimodal landscape suitable for optimization with a GA. SINR is a valid fitness function because it measures how well the array's radiation pattern focuses energy on a SOI and steers nulls towards interference. A flowchart of an SGA adapted to maximize an anti-jamming beamforming array's SINR is shown in Figure 1.5. Terminology related to genetic algorithms are defined in Table 1.2.

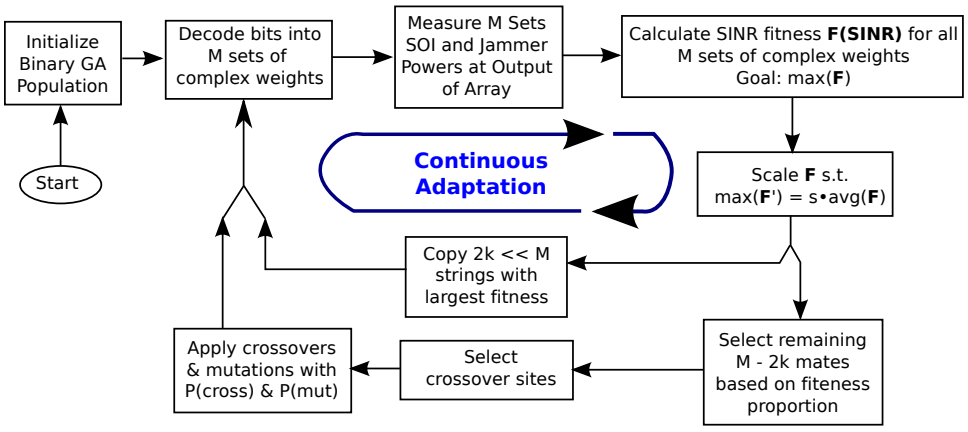


Figure 1.5: Flowchart of a SGA adapted to maximize an anti-jamming beamforming array's SINR.

At generation zero, the GA randomly chooses a population of M binary strings

Table 1.2: Terminology used to describe Genetic Algorithms

Terminology	**Definition**
String	Genetic representation (typically binary) of the GA's input parameters.
Chromosome	See "String"
Genome	Set of genetic strings
Phenome	Problem specific decoding of the Genome (i.e., conversion of binary strings to array attenuator and phase shifter settings)
Genes	Individual bits in a string
Allele	Set of all possible values that a gene can have
Triallelic	Refers to alleles having values of (-1, 0, +1) where '-1s' are recessive '1s'
Haploid	Single string representation
Diploid	Dual string representation
Mate-selection	Process of selecting mate pairs based on string fitness
Crossover	Process of slicing, swapping, and splicing a pair of strings akin to chromosomal crossover
Mutation	Random changes to a small percentage ($\leq$ 2%) of genes in a population

of length L. Each string representes the settings for the array's complex weights (i.e., amplitudes and phase). The complex weight search space is constrained such that $\mathcal{R}\{\hat{\mathbf{a}}\} \in (0, 1]$ and $\mathcal{I}\{\hat{\mathbf{a}}\} \in [0, 2\pi)$. Although there are an infinite number of possible analog solutions in this search space, the decimal to binary conversion creates a finite search space.

Although the initial population is chosen randomly, the GA performs a guided search through the binary parameter space. Strings with higher SINR fitness are

chosen more often than strings with low SINR fitness. The crossover operation shares the good qualities of fit strings and created children with better fitness compared to their parents. Mutation reintroduces genetic material into the population that is lost during the mate selection process.

The SGA performs best when desired solutions did not change with time. In theory, the SGA converges to a steady-state population with little diversity [27]. This is problematic in situations when signal directions change because the SGA theoretically converges to the first signal set, and it will be unable to adapt to the new set of directions without re-initializing the algorithm. This requires additional time for the SGA to converge to the new set of directions. In many situations as in the case of portable WiFi applications, the SOI's DOA remains constant once located, but interference can change directions (i.e., mobile jammers). With other applications including ground to UAV communications, the SOI can change directions because the SOI's transmitter constantly moves.

There are other operators and variations that improve the GA's performance against mobile signals by preserving generational population diversity. One variation of the GA uses diploid strings with triallelic values (i.e., TDGA) as previously discussed by [24, 28] and shown in Figure 1.6[4]. The hardware settings are first encoded into a dual string representation with tertiary values. This allows for long-term genetic memory because the dual-string representation allows dominant (i.e., expressed)

[4]The SGA shown in Figure 1.5 uses binary haploid (i.e., single) strings to encode the array's complex weights

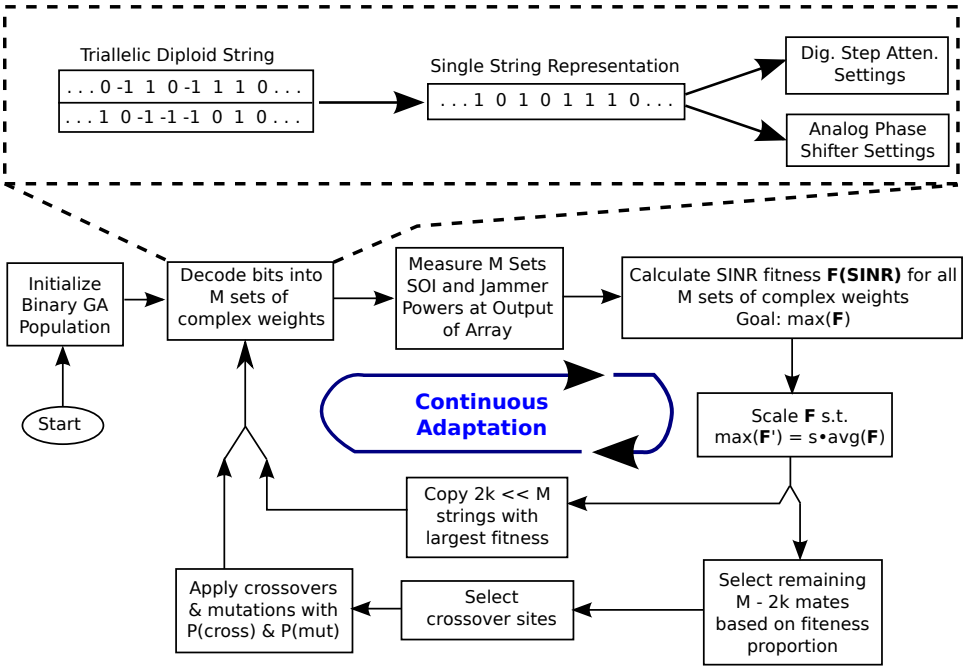


Figure 1.6: Flowchart of a TDGA adapted to maximize an anti-jamming beamforming array's SINR.

and recessive (hidden) properties to be stored for many generations, and the tertiary values allows hidden properties to switch places with expressed properties when the environment changes. The diploid strings in Figure 1.5 are encoded into single string representations using these dominance relations. The encoded single string representation is then decoded into the array's complex weights (i.e., attenuators and phase shifters).

The primary focus of this thesis will be on evaluating the SGA and TDGA in optimizing an anti-jamming beamforming array and perform hardware fault recovery. It is the intent to compare GA results with Simulated Annealing as well as the Hill Climbing Algorithm to create a baseline in Chapters 5 and 6. The thesis discusses how Simulated Annealing is applied to anti-jamming beamforming arrays with hardware

fault recovery in the following subsection.

1.4.2 Simulated Annealing in Anti-Jamming Beamforming with Hardware Fault Recovery

Another stochastic search algorithm used is Simulated Annealing (SA). SA is an algorithm based on the concept of annealing metallic objects [29]. At high temperatures (i.e., 1000 K), the atoms in a metal object move rapidly in random directions and bounced off each other. As the metal is slowly cooled to 0 °K, the atoms' velocities decrease until they formed a perfect crystalline lattice. The SA algorithm models this process to guide the search process. SA is also useful for solving multimodal combinatorial optimization problems because its random nature at high temperatures prevents the algorithm from getting stuck at local optima early in its search. The SA algorithm's applications to electromagnetic problems includes optimal array design [30, 31] and calibration of sparse antenna arrays [32]. To the best of the author's knowledge, SA has not been applied to optimizing complex array weights for anti-jamming beamforming after antenna elements have faulted.

The application of simulated annealing to anti-jamming beamforming with hardware fault recovery arrays is similar to that of the GA, and a block diagram of simulated annealing is shown in Figure 1.7. The SA is a form of a hill climbing algorithm (HCA) that modifies solutions based on a temperature schedule and always accepts better solutions. The temperature schedule controls the mutation rate with a high probability of mutation at evaluation $v = 0$, and the mutation rate approaches zero

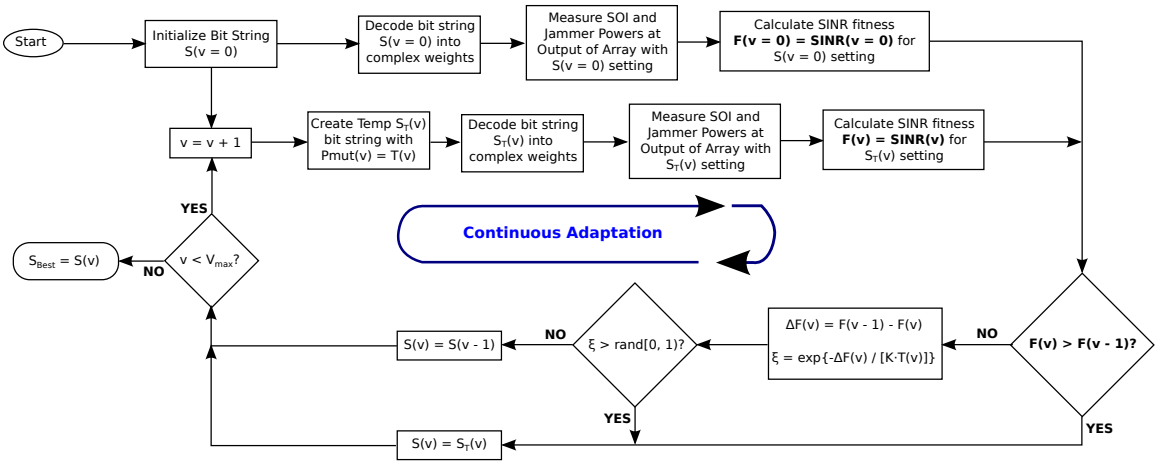


Figure 1.7: Flowchart of Simulated Annealing adapted to maximize an AJBF array's SINR when stationary signals are present.

when the SA terminates at evaluation V_{f}. A sigmoid function with cooling rate τ is typically used as SA's temperature schedule. SA uses a second parameter known as the Metropolis condition that allows the SA to accept worse solutions dependent on the cooling schedule [29]. The difference between SA and HCA is that HCA never accepts worse solutions because the Metropolis condition is disabled in HCA.

1.5 Thesis Contributions

The necessity for research on stochastic optimization of anti-jamming beamforming arrays with hardware fault recovery is demonstrated in this thesis. SOI and jammer DOAs are unknown, so the parameter search space is combinatorial with trillions of possible solution, and the fitness landscape is multi-modal. This renders classical optimization algorithms such as Least Mean Squares (LMS) ineffective because they would stagnate in one of many local optima. Use of stochastic algorithms is proposed because they are effective in optimizing combinatorial search problems with large

parameter search spaces. The contributions of this thesis are:

1. Analytical models are developed and experimental results show that small antenna arrays can thwart interference using dynamically applied stochastic algorithms. This type of in-situ optimization, with an algorithm dynamically optimizing a beamformer to thwart interference sources with unknown positions, inside of a anechoic chamber has not been done before to the author's knowledge.
2. It is shown that these algorithms can recover from hardware failures and localized faults in the array. Experiments were performed with a proof-of-concept four-antenna array. This is the first hardware demonstration showing an antenna array with live hardware fault recovery that is adapted by stochastic algorithms in an anechoic chamber.
3. A comparison of multiple stochastic algorithms in performing both anti-jamming and hardware fault recovery. The algorithms to be compared include genetic algorithms and simulated annealing.
4. Demonstration via simulations showing that stochastic algorithms can be used to continuously track and mitigate interfering signals that continuously move in an additive white Gaussian noise wireless (AWGN) channel.
5. Demonstration via simulations showing that the SGA can adapt to mobile signals and perform hardware fault recovery in an AWGN channel even after convergence. This result is novel because current theory predicts that the SGA

cannot adapt to mobile signals and hardware faults after convergence in any situation. The results demonstrate a case where the SGA's mutation operator introduces sufficient diversity and allows the SGA to continue its search without the need to randomly generate a new population.

1.6 Thesis Outline

The remainder of this thesis is organized as follows. Chapter 2 discusses the background and previous research in anti-jamming beamforming antenna arrays with hardware fault recovery optimized by stochastic algorithms. Chapter 3 discusses new models for anti-jamming beamforming. Chapter 4 discusses new models for antenna array fault detection, recovery, and localization. Chapter 5 discusses dynamic optimization of anti-jamming beamforming arrays with both simulated and *in-situ* results. Chapter 6 discusses *in-situ* experiments and simulation analysis of hardware fault recovery. Finally, Chapter 7 concludes this thesis and discusses future work related to anti-jamming beamforming arrays with hardware fault recovery.

Chapter 2

Background and Previous Research in Beamforming Arrays

There is a wide range of research in antenna beamforming with hardware fault recovery. This research has generally been on large arrays with dozens to hundreds or thousand of antennas. The authors behind these research papers generally assume that the SOI's DOA is known *a priori*, and this simplifies the problem significantly. The resulting problem becomes one of finding the optimal complex antenna weights that steer the array's main beam towards the SOI while simultaneously keeping the SLLs below a predefined value such as -20 dB. If the antenna array consists of uniformly spaced elements arranged on a line, the array can be tuned by applying uniform, Binomial, or Dolph-Chebyshev weights [3, 33]. Array patterns can also be synthesized whether the arrays are linear or planar.

The problem becomes slightly more difficult if the SOI changes directions or if the wireless channel faded. Adaptive algorithms such as Least Means Squares (LMS) can be used to form and move the main beam in relation to the SOI thereby tracking its movements [9]. This set of problems is still relatively simple to solve, as the SOI's initial direction is known *a priori*. Furthermore, interference in this problem set is either lacking, or the arrays are large enough such that the SLLs can be tuned to at

least -20 dB in relation to the main beam at all times.

The beamforming problem becomes difficult to solve when the SOI's DOA is not previously known, interference is present with unknown directions, and the antenna array is small[1]. There are commercial applications where the SOI's and interference's DOAs are not known *a priori* such as portable 802.11 WiFi communications in crowded public spaces with many people using various wireless devices. There are also military applications such as communication with small UAVs where the UAV does not initially know its basestation's location, and adversaries jam the UAV from the moment it is turned on. In these cases, the problem search space that consists of the complex antenna weights becomes multimodal [34], and classical adaptive algorithms such as LMS are rendered ineffective because they converge on local optima [35]. A categorization of beamforming problems are summarized in Table 2.1. Note that these solutions assume that the antenna elements are infinitesimal dipoles and generate first order approximations of the antenna array radiation patterns.

Reliability and fault tolerance are also important in antenna arrays because they typically operate in the field and cannot be easily repaired. Agrawal *et al.* [15] discuss redundancy techniques to increase active phased array Mean Time Between Failure (MTBF) for an 8,000 element circular antenna array with 40dB Taylor low sidelobe taper. An antenna array's components are divided into passive and active components. Agrawal assumes that passive components such as transmission lines

[1]Small in the sense that the number of signals present (both SOI and interference with multipath reflections) is comparable to the number of antennas in the array typically 12 antennas or less.

Table 2.1: Classes of beamforming problems

Array Type	**Signal States**	**Method(s) to Find Weights**
Large ULA	SOI DOA known & constant	Uniform, Binomial, Dolph-Chebyshev weighting or array synthesis
Large ULA	SOI DOA known & variable	Adaptive algorithms such as LMS, CGM, or MVDR. Stochastic search algorithms also viable.
Circular array	SOI DOA known & constant	Similar to ULA except array factor modeled using Bessel functions [3, 33]
Small planar array or ULA	SOI DOA known & variable with single interference with known DOA (i.e., known noncoherent multipath reflection)	Adaptive algorithms such as LMS, CGM, or MVDR. Stochastic search algorithms also viable.
Small planar array or ULA	SOI DOA unknown & interference DOAs unknown	Stochastic search algorithms

and power dividers have infinite MTBF. Active components included power supplies, Transmit/Receive (T/R) modules, and T/R control modules.

In [15], the authors assign finite MTBFs to active components with values based on the MIL-217 standard. An array failure occurred when the sidelobe level (SLL) increased by 3 dB. They found that it requires less elements to cause a 3 dB SLL increase when the antennas are clustered with eight T/R modules per control module compared to one T/R module per control module. Although the no-cluster configuration has a 1,120 hour average per antenna MTBF, and the eight antenna cluster

configuration has a 162 hour average antenna MTBF, they note that a configuration with redundant power supplies and dual antenna clusters has the best 1,882 hours average antenna MTBF. An issue with antenna arrays used in commercial wireless communication systems is that they cannot have hundreds or thousands of antenna elements because such arrays are limited in hardware cost, mechanical volume, and weight. Even in military wireless communications, it is desirable to use smaller arrays. for example due to the existence of small UAVs in the communications chain [11]. Thus, there is a need to investigate redundancy via software algorithms in the form of reconfiguration (i.e., self-healing) of remaining hardware subsystems to compensate for hardware failures.

Hardware fault detection and recovery can be divided into several areas: fault detection, recovery from faults, and localization of faults. Detection is clearly important because a failure cannot be compensated if it is unknown to the algorithm. The primary goal of fault recovery is to ensure that the array continues operating (although in a possibly hindered state) until it can be repaired. This in essence extends the array's MTBF beyond that of individual components' MTBFs. Fault localization is necessary to aid in the repair of the array thereby reducing the array's MTBR. It follows that fault localization can aid in speeding up fault recovery by excluding faulty hardware during the self-healing process (i.e., performing an in-situ swap of a faulty component with a standby component). Fault localization for the purposes of *in-situ* replacement of faulty components is beyond the scope of this thesis.

The ability to detect, localize, and correct for hardware faults in an antenna array has been demonstrated. Lee *et al.* [36] implement a radar beamforming system with a built-in pulsed signal injection system for fault detection, localization of faults, and fault correction. The beamforming system consists of ten columns of eight monopole antennas that connect to computer controlled T/R modules. Their system performs fault detection and localization by injecting pulsed RF signals into each of the monopole antennas via microstrip transmission lines mounted in proximity to the antenna banks. The injected signals consist of pulse trains that are fed through a ten way divider and phase toggled such that the amplitude and phase of each element can be determined. Once the amplitude and phase of each element is known, their method computes array performance metrics including gain, null depths, and side lobe patterns. They also generate fault detection maps to isolate faults and implemented a fault correction algorithm to retune the remaining element amplitude and phase weights to regenerate desirable beam patterns and SLLs.

However, the method demonstrated by [36] requires additional hardware to create and inject phased pulse trains into the array at the RF frequency. Their system also requires receiver calibration as reference for insertion loss and phase measurements, system level calibration of the phase-toggling circuitry to determine its accuracy, and calibration of antenna element phase shifter bits to determine accuracy of each phase shift bit. Because the system is built as a radar beamformer, it should be noted that the mainbeam direction and SLLs are known previously, so pre-fault amplitude

and phase weights are calculated by default and used as references for comparison to the faulted array's parameters. Because fault detection hardware can fail and prevent array fault detection, it is desirable to perform fault detection, localization, and recovery functions in software without the need for additional hardware.

The remainder of this chapter is as follows. Section 2.1 discusses the assumptions used in solving the anti-jammer beamforming with hardware fault recovery problem and included an upper level algorithm comparison. Section 2.2 explains the Array Factor method used to calculate first-order approximations of antenna array radiated fields by classical beamforming weighting methods and optimization algorithms.Section 2.3 summarizes and compares several array weighting methods commonly used in radar beamforming. Although these techniques are not used in anti-jamming beamforming, they are relevant to antenna array hardware fault recovery discussed later in this chapter. Section 2.4 reviews techniques in beamforming using LMS and related algorithms. Section 2.5 discusses previous research in beamforming using GAs, and Section 2.6 discusses previous research in beamforming using simulated annealing. Section 2.7 discusses previous research in antenna array hardware fault detection, recovery, and localization. Finally, Section 2.8 summarizes this chapter.

2.1 Assumptions and Algorithm Comparisons

Much of the previous beamforming research relates to beamforming in the sense of focusing a main beam towards an SOI. Applications of such beamformers include

radar systems in which the SOI's DOA is known, as it consists of reflections from targets whose properties (such as position, velocity, and relative size) are desired. The array is electronically steered towards a direction where a target might be present. These arrays are often large and consist of dozens to hundreds or thousands of antennas. If interference is present, the beamformer's radiation pattern can be tuned such that the SLLs are at least 20 dB below the main beam pointed at the SOI.

In contrast, anti-jamming beamformers researched in this thesis are used with wireless communication systems. The SOI's DOA is most often unknown *a priori*, and interference exists that prevent the receiver from decoding information transmitted by the SOI's source. Because the systems are often commercial, the arrays are limited in size due to budget and space constraints. As such, the following assumptions are made throughout this thesis:

1. The signal directions are unknown prior to system startup.
2. The SOI is jammed when the receiver turned on.
3. The SOI and interference are potentially mobile, so an adaptive solution is required.
4. The problem is combinatorial because the complex array weights are digitized to create a finite search space.
5. The array consists of hardware components with finite MTBF, so it is possible for hardware faults to occur during or after optimization of the array. As such, it is necessary for the algorithms to perform hardware fault recovery during the

optimization process.

Several stochastic search algorithms are compared in Table 2.2. Because the problem being solved is multimodal, LMS and CGM would not perform well because they both tend to get stuck in local maxima. CGM differs from LMS in that, at each iteration, CGM searches the parameter space in a direction orthogonal to its previous search step [9]. Once LMS and CGM converge to a solution, the search steps are effectively zero, so neither algorithm can find a better solution if one exists. The evaluation and comparison of all possible search algorithms used in beamforming is outside the scope of this thesis.

Table 2.2: Comparison of stochastic search algorithms used in beamforming

Approach	**Features**	**Drawbacks**
Least Mean Squares	Adaptive feedback	Local search with poor multi-modal performance
Conjugate Gradient Method	Searches parameter space using conjugate directions	Signal directions needed, poor multi-modal performance, $\mathcal{O}(N^2)$
Genetic Algorithms	Population based global search	Run-time is problem dependent
Simulated Annealing	Evaluates solutions sequentially	Convergence is cooling schedule dependent

2.2 Array Factor Method

The array factor (AF) is a means of estimating an antenna array's radiation pattern via its electric field. The AF calculation assumes that the electromagnetic waves

are steady sinusoidal state in free space, and the observation point is located in the far-field. The antenna elements are assumed to be infinitesimal dipoles. The antenna array's electric field can be expressed via (2.1) ([3, 37]) as

$$\hat{\bar{E}}\left(R,\theta,\phi\right) \;=\; \hat{\bar{F}}\left(R,\theta,\phi\right)\hat{A}\left(\theta,\phi\right) \tag{2.1}$$

where $\hat{\bar{F}}\left(R,\theta,\phi\right)$ is the element factor that represents individual element contribution to the radiation pattern. For an infinitesimal dipole element, the element factor can be expressed as

$$\hat{\bar{F}}\left(R,\theta,\phi\right) \;=\; \hat{\bar{F}}\left(R,\theta\right) \;=\; \bar{a}_{\theta}\,\frac{\mu_{\mathrm{o}} j\beta e^{-j\beta R}}{4\pi R}\sin\left(\theta\right)\; I_{\mathrm{o}}\mathrm{d}\ell \tag{2.2}$$

The array factor $\hat{A}\left(\theta,\phi\right)$ is the sum of elements contribution to the radiation pattern and can be expressed as

$$\hat{A}\left(\theta,\phi\right) \;=\; \sum_{i=1}^{N}\hat{a}_{\mathrm{i}}e^{j\beta\bar{a}_{\mathrm{R}}\bullet\bar{d}_{\mathrm{i}}} \tag{2.3}$$

where $\beta \;=\; 2\pi/\lambda_{\mathrm{o}}$ is the free-space wavenumber. The array factor described in equation (2.3) has three important parts:

1. $\left\{\hat{a}_{\mathrm{i}}\right\},\ i\in[1,N]$ are complex weights added to each antenna element.
2. $\left\{\bar{d}_{\mathrm{i}}\right\},\ i\in[1,N]$ are the X,Y,Z coordinates of the N antennas
3. $\bar{a}_{\mathrm{R}}$ is the spherical coordinate radial unit vector expressed in terms of its X,Y,Z coordinate projections.

It can be seen from (2.3) that an array's radiation pattern can be controlled by either moving the antennas or by adjusting the complex weights. The former method is known as reconfigurable antenna arrays, and the later method as complex weighted beamforming. Phased Antenna Arrays (PAAs) are a subset of the later method where the complex weights are purely imaginary. PAAs are used when the SOI direction is

known *a priori*, and when either there is no interference present, or the array has a sufficient number of elements such that the SLLs can be reduced to a level where the interference does not pose a threat. If there are not enough elements to keep the SLLs below -20 dB with respect to the main lobe, and/or it is necessary to steer nulls in directions of jammers, amplitude portions of the complex weights $\{\hat{a}_{\mathrm{i}}\}$ add additional degrees of freedom in steering the main beam towards the SOI while simultaneously steering nulls towards jammers.

Antenna radiation patterns are typically expressed in terms of radiated power. It can be seen from equation (2.1) that an array's radiated power (i.e., directivity) can be expressed as

$$D(\theta,\phi) \;=\; D_{\mathrm{EF}}(\theta,\phi)\left|\hat{A}(\theta,\phi)\right|^2 \tag{2.4}$$

If the antenna elements are infinitesimal dipoles, the directivity due to the element factor $D_{\mathrm{EF}}(\theta,\phi)$ reduces to [3]

$$D_{\mathrm{EF}}(\theta,\phi) \;=\; D_{\mathrm{EF}}(\theta) \;=\; \sin^2(\theta) \tag{2.5}$$

and equation (2.4) can be expressed as

$$D(\theta,\phi) \;=\; \sin^2(\theta)\left|\hat{A}(\theta,\phi)\right|^2 \tag{2.6}$$

The research discussed in this thesis focuses on azimuth (i.e., $\theta = \pi/2$) radiation patterns, so it can easily be seen that equation (2.6) for the azimuth plane reduces to

$$D_{\mathrm{AZ}}(\phi) \;=\; \left|\hat{A}(\theta = \pi/2,\phi)\right|^2 \tag{2.7}$$

The power and simplicity of using the array factor for calculating array radiation patterns when modeling its elements as infinitesimal dipoles can be seen through equations equations (2.4) – (2.7). The array's radiation pattern involves a calcu-

lation that is $\mathrm{O}(N)$, and integration of the electric fields over $\theta \in [-\pi/2, \pi/2]$ and $\phi \in [-\pi, \pi]$ is unnecessary. If the azimuth radiation patterns are desired, the calculation of the array's radiation pattern depends only on the antenna elements' locations and complex weight values. The drawback to the array factor method, however, is that antenna elements have finite lengths, and mutual coupling exists between elements. The array factor does not account for mutual coupling, but it is generally acceptable for theoretical modeling of antenna arrays used by classical beamforming algorithms and beampattern synthesis techniques.

2.3 Array Weighting Methods and Beampattern Synthesis

There are multiple beamforming array weighting methods that include uniform amplitude phasing, binomial weighting, and Dolph-Chebyshev weighting. The methods induce tradeoffs between mainlobe gain, mainlobe beamwidth, and SLLs as described in great detail by [9,33]. These methods can be used when the SOI's DOA is known *a priori*, the SOI does not move, and there are sufficient elements in the array such that the SLLs can be tuned below an acceptable level (i.e., -20 dB) with respect to the SOI's mainbeam. Table 2.3 summarizes the properties of various weighting methods as described in [9,33] for an N element linear array with spacing d between elements. The beamwidths and SLLs are approximations assuming large N.

The term f_{bbf} found in the Dolph-Chebyshev mainbeam gain and 3 dB beamwidth approximations is a beam broadening factor defined in [3] as

Table 2.3: Comparison of various array weighting methods

Method	**Mainbeam Gain**	**3dB Beamwidth**	**PSLL (dB)**
Uniform Broadside	$2N\left(\frac{d}{\lambda}\right)$	$\approx 2\left(\frac{1.391\lambda}{\pi N d}\right)$	-13.5
Binomial ($d = \lambda/2$)	$\approx 1.77\sqrt{N}$	$\approx 1.06\left(N-1\right)^{-\frac{1}{2}}$	$-\infty$
Dolph-Chebyshev (SLL = $\mathrm{R_o}$, unitless)	$\approx \left(\frac{1}{f_{\mathrm{bbf}}}\right)(2N)\left(\frac{d}{\lambda}\right)$	$\approx 2 f_{\mathrm{bbf}}\left(\frac{1.391\lambda}{\pi N d}\right)$	$\mathrm{R_o}$ (dB)

$$f_{\mathrm{bbf}} \;=\; 1 \;+\; 0.636\left\{\frac{2}{\mathrm{R_o}}\cosh\left[\sqrt{\left(\cosh^{-1}\mathrm{R_o}\right)^2 \;-\; \pi^2}\right]\right\}^2 \tag{2.8}$$

The beam broadening factor $f_{\mathrm{bbf}} \approx 1$ for a SLL of 15 dB down from the mainlobe gain, and it approaches 1.7 for a SLL of 60 dB (see Figure 6.24(a) in [3]). The approximation of the Dolph-Chebyshev mainbeam gain in Table 2.3 assumes that $\mathrm{R_o^2} \gg 1$ which is clearly valid for SLLs $\geq$ 15 dB. Thus, Dolph-Chebyshev arrays sacrifice mainbeam gains and 3dB beamwidths for SLLs greater than 15 dB, and the Dolph-Chebyshev array approaches the Binomial array as SLLs approach infinity. It can be seen from Table 2.3 that the uniform weighted array have the smallest 3dB beamwidth followed by the Dolph-Chebyshev and binomial weightings. This is in agreement with [3].

Although Table 2.3 approximates the mainbeam gain of a Dolph-Chebyshev ULA, [3] gives a more exact formula for an array scanned near broadside that is

valid for $-60dB \leq SLL \leq -20dB$ in (2.9):

$$\mathrm{G_o} \;=\; \frac{2\mathrm{R_o^2}}{1 \;+\; (\mathrm{R_o^2} \;-\; 1)\,\mathrm{f_{bbf}}\frac{\lambda}{(\mathrm{L} \;+\; \mathrm{d})}} \tag{2.9}$$

$$L \;=\; (N \;-\; 1)\,d \tag{2.10}$$

where d is the spacing between elements. The design of Dolph-Chebyshev ULAs is made simpler by use of the z-transform where $z \;=\; e^{j\Psi}$ [33] such that (2.3) for ULAs is expressed in simpler terms as

$$AF\,(z) \;=\; \sum_{n\;=\;1}^{N} w_n z^{n\;-\;1} \tag{2.11}$$

$$AF\,(z) \;=\; w_n\,(z \;-\; z_1)\,(z \;-\; z_2)\cdots(z \;-\; z_N) \tag{2.12}$$

$$z_n \;=\; e^{j\Psi_n} \tag{2.13}$$

If $w_n \;=\; 1$, (2.12) represents a unit-circle in the z-domain, and (2.13) represents zeros on that unit-circle. Haupt [33] notes that for a ULA with half-wavelength spacing between elements, equation (2.13) can be expressed as

$$\Psi_n \;=\; 2\arccos\left\{\frac{\cos\left(\frac{(n\;-\;0.5)\pi}{N\;-\;1}\right)}{\cosh\left(\frac{\pi A_{\mathrm{cheb}}}{N\;-\;1}\right)}\right\} \tag{2.14}$$

$$A_{\mathrm{cheb}} \;=\; \frac{1}{\pi}\arccosh\,(\mathrm{R_o}) \tag{2.15}$$

Substituting (2.14) into (2.12), letting $w_n \;=\; 1$, and simplifying the result yields a polynomial in z whose coefficients are the array's Chebyshev amplitude coefficients. The Dolph-Chebyshev array can be steered in azimuth by using the canonical form $\phi_{\mathrm{n}}\,(\phi_{\mathrm{o}}) \;=\; -\,\beta\cos\,(\phi_{\mathrm{o}})\,(n \;-\; 1)$ [3].

Balanis and Haupt [3,33] discuss the Uniform Circular Array (UCA). The concept

behind a UCA is similar to the ULA except that the antenna elements are equally spaced on a circle with radius r. They both note that the array factor for the UCA can be expressed as a sum of Bessel functions where Haupt expressed it in (2.16) as

$$AF \;=\; N \sum_{n=-\infty}^{\infty} J_{\mathrm{nN}}\left(\beta r\right) e^{j\mathrm{nN}\left(\frac{\pi}{2} \,-\, \xi\right)} \tag{2.16}$$

where

$$\xi \;=\; \arctan\left(\frac{v - v_o}{u - u_o}\right) \tag{2.17a}$$

$$u \;=\; \sin\left(\theta\right)\cos\left(\phi\right) \tag{2.17b}$$

$$v \;=\; \sin\left(\theta\right)\sin\left(\phi\right) \tag{2.17c}$$

u_o and v_o are equations (2.17b) – (2.17c) applied in the mainbeam direction, and $J_{\mathrm{nN}}\left(\beta r\right)$ are Bessel functions of the First Kind of order $n \times N$. Haupt notes that if N is large, then (2.16) can be simplified to $N\, J_{\mathrm{o}}\left(\beta r\right)$. The mainbeam of the UCA can be steered to (θ_o, ϕ_o) by applying a phase at each element per [33] as:

$$\zeta \;=\; e^{-j\beta r \sin(\theta_0)\cos(\phi_o \,-\, \phi_n)} \tag{2.18}$$

where ϕ_n is the angular location of element n on a radius r in the XY-plane. Reduced SLLs of UCAs using Dolph-Chebyshev (and Dolph-Chebyshev equivalent) amplitude weights are investigated in [38–41].

A concentric circular array (CCA) can be formed by arranging several circular arrays with radius $r_{\mathrm{n}},\ n \in [1, N_{\mathrm{r}}]$ such that $r_1 < r_2 < \cdots < r_{\mathrm{n}}$, and a single element is placed in the CCA's origin. Haupt [33] stated that the array factor of the CCA

can then be expressed as

$$AF \;=\; 1 \;+\; \sum_{n=1}^{N_{\mathrm{r}}} w_{\mathrm{n}} N_{\mathrm{n}} \sum_{-\infty}^{\infty} J_{mN_n} \left(\beta r_{\mathrm{n}} \sin\left(\theta\right) e^{jmN_{\mathrm{n}}\left(\frac{\pi}{2} \,-\, \phi\right)} \right) \tag{2.19}$$

If the radii of the concentric rings are large, Haupt notes that the CCA's array factor is independent of ϕ, and it can be expressed in terms of each ring's principle component J_{o} in (2.20).

$$AF \;=\; 1 \;+\; \sum_{n=1}^{N_{\mathrm{r}}} w_{\mathrm{n}} N_{\mathrm{n}} J_{\mathrm{o}} \left(\beta r_{\mathrm{n}} \sin\left(\theta\right) \right) \tag{2.20}$$

It should be noted that w_{n} represents element weights for ring n. If the rings are equally spaced with radius $r_{\mathrm{n}} \;=\; n\lambda/2$, and the spacing between elements is $d_{\mathrm{n}} \;=\; \lambda/2$, then the maximum number of equally spaced elements in ring n is $N_{\mathrm{n}} \;=\; \lfloor 2\pi n \rfloor$ [33]. To the best of the author's knowledge, research in SLL reduction in CCAs using array weighting methods such as Dolph-Chebyshev is limited to [42].

An extension of array weighting is beampattern synthesis. The power in beampattern synthesis is in creating arbitrarily shaped radiation patterns that meet different mainlobe and sidelobe requirements as noted above as well as custom SLLs in a specific set of directions. Zhang and Ser [43] note that the Dolph-Chebyshev method cannot specify null locations, mainlobe beamwidth, and mainlobe ripple. They develop a pattern synthesis method that accounts for mutual coupling between antenna elements and null multiple angular regions to thwart interference. The beampattern synthesis is treated as an optimization problem where a cost function is minimized with respect to the array weights. They evaluate two cost functions (Dolph-Chebyshev and quadratic) and made the problem convex by imposing upper and lower limits on

the mainlobe.

In [43], the authors simulate their synthesis methods on 32-element ULAs and UCAs both with 0.5λ and 0.3λ spacing. Their synthesis method with the Dolph-Chebyshev cost function outperform both the classical Dolph-Chebyshev weighting method in forming a beampattern that steered a null at 90° with a 20° beamwidth while simultaneously nulling 20° and 120° with 1° and 6° null widths. Their method performs better with a ULA having 0.5λ interelement spacing although the 0.3λ spacing ULA achieves acceptable beampatterns, and they have similar results for the UCA arrays.

2.4 Beamforming with Gradient Search Based Adaptive Algorithms

There has been recent research in anti-jamming beamforming arrays. Traditional gradient search beamforming methods include Least Mean Squares (LMS), Minimum Variance Distortionless Response (MVDR), Sample Matrix Inversion (SMI), and Conjugate Gradient Method (CGM). Gross summarizes these method in [9], and Van Trees discusses these methods in great detail in [2].

Zhang *et al.* [8] notes that GPS systems need to be protected from accidental and intentional interference because GPS systems are used for navigation and guidance. GPS systems initialize with no knowledge of the channel state and signal directions, so blind anti-jamming techniques such as minimum variance (MV) null-steering are used. However, current blind anti-jamming techniques create carrier phase errors that

prevent GPS receivers from accurately predicting the estimated range or cause them to lose carrier phase lock. They develop and verify through simulations an algorithm that removes an estimated phase error from a MV beamformer. They implement their algorithm using SMI to estimate the signal covariance matrix $\mathbf{R}$ and its inverse. Although their simulations produce positive results, Gross [9] notes that SMI can produce erroneous results if $\mathbf{R}$ is poorly conditioned.

A sidelobe canceller as described by Applebaum [44] uses a blocking matrix to remove the SOI from the MVDR training data prior to shifting the sidelobes to cancel interferers. However, this algorithm cancels the SOI if a mismatch in its assumed DOA exists. To alleviate these problems, Lei *et al.* [45] propose a generalized sidelobe canceller in which the SOI blocking matrix is widened to account for SOI direction mismatch. Although they prove that this method prevents SOI self-cancellation with SOI DOA mismatch, their beamformer needs to know the SOIs DOA with mismatch limits *a priori*.

Xu and Lui [46] comment that current blind adaptive algorithms exploit known properties (such as their modulation types) of the SOI and interferers to determine the SOIs DOA. They develop an adaptive blind beamforming algorithm called Non-circularity Rate Maximization (NORM) that blindly recovers noncircularly polarized signals such as BPSK, ASK, and AM signals in the presence of circular (i.e. purely phase modulated) interference and noise. They compare their NORM beamforming algorithms results to MVDR and covariance-cumulant (C^2) beamformers with a non-

circularly polarized SOI that have a 5° DOA mismatch. They compare these methods both with and without the presence of circularly polarized interfering signals.

Without circularly polarized interference present, they note that MVDR steers a null in the SOI's direction when an SOI direction mismatch exists, and the C^2 and NORM methods pointed the main beamlobe at 5°. With the presence of a QPSK interferer at 30°, MVDR adds an additional null at 30°, and the C^2 beamformer places a null close to the desired signal direction while attempting to null the QPSK interferer. On the other hand, their NORM beamformer does not move the SOI's mainlobe while steering a null at 30°. It should be noted that [46] assumes that the SOI always uses noncircular modulation. If the SOI changes its modulation to circular modulation (i.e., QPSK), the NORM algorithm would no longer distinguish the SOI from circularly modulated interference.

Chen *et al.* [47] develop three modified CGM based beamforming algorithms that minimizes the BER for a Quadrature Phase Shift Keyed (QPSK) wireless system and compared them to MMSE. Their first method is called minimum BER (MBER), and it finds optimal array weights by calculating the gradient of the beamforming array's marginal error PDFs. These PDFs are based on the real and imaginary parts of the receiver's decision regarding the transmitted symbol given that its decision is made in error. Their second method uses a Parzen window estimate of the array's output PDF using a K length block of data and a kernel width similar in form to the array's noise standard deviation. They called their third method Least BER (LBER). It

estimated the array output PDF using a normalized K length block of data that gave rise to a stochastic gradient of the marginal error PDFs.

They simulate their three algorithms and MMSE for a three element linear array with one SOI and three interference signals. The MBER algorithm performs better than MMSE when the SIRs of all the three jammers are 0 dB, as MBER requires a lower Signal-to-Noise-Ratio (SNR) than MMSE for those SIRs. When all three jammers have -2 dB SIR, MMSE no longer functions while MBER attains a BER of 10^{-10} for SNR of $\approx$24 dB. Neither MMSE or MBER perform well when the SIRs for the first two jammers are -2 dB and -6 dB for the third jammer. They also compare magnitudes and phases of the array radiation pattern for the array weights calculated by MMSE and MBER given the cases of SNR = 15 dB, SIR = 0 dB for all jammers, and SNR = 20 dB and SIR = -2 dB for all jammers. Both pattern magnitudes are similar. Even though only one jammer is completely nulled in the second case, the phases of the other two jammers are 90° out of phase with respect to the SOI for MBER while the other jammers are 90° and 30° out of phase with respect to the SOI for MMSE.

Lee [48] notes that interference problems between femtocell basestations occur because they are installed in confined areas. Their simulations show that interference can be mitigated through a combination of coordinated user scheduling (CUS) and beamforming techniques. Because WiFi basestations are typically installed indoors, they can be considered femtocells. CUS is infeasible because jammers may not be

WiFi basestations operating in a femtocell network.

2.5 Beamforming using Genetic Algorithms

There has been relatively few studies of GA optimization of antenna arrays. Haupt *et al.* [49] built a beamforming array that successfully thwarts a single jammer. It consists of eight active elements with sixteen dipole antennas per element. They assume that the SOI's direction of arrival (DOA) is known, and they tune the array to focus energy towards the SOI before turning a jammer on. If no jammers are present, the algorithm would minimize the SOI's power because their GA minimized the array's output power. They solved this problem by modifying a limited number of attenuator and phase shifter bits. This allows their algorithm to form a null towards a single jammer while not altering the main lobe. Their algorithm's goal is to thwart a jammer, and it does not form the main lobe because the array is already tuned towards the SOI.

Massa *et al.* [50] simulate a modified standard GA with real-time parameterized crossover and mutation probabilities to thwart jammers with randomly varying DOAs. They parameterize the crossover and mutation probabilities on the binary strings' variance in a population of N trial solutions. Probabilities vary linearly with solution variance: Maximized (minimized) when solutions in the current generation have low (high) variance. Their modified GA discards a percentage of $\delta(k)$ solutions at the k^{th} generation and replaces them with randomly generated strings. The percentage of replaced solutions varies linearly with solution variance, and the replacement rate

is maximized when solution variance is minimal. Weile *et al.* [28] develop a dual (i.e., diploid) string GA with triallelic values. Their simulations show that a GA with dominance and diploidy (DDGA) can thwart mobile jammers. Their work is based on GA theory developed by Goldberg [24]. They add two more operators called storage and resurrection. The storage operator copies the most fit string in the current generation into memory, and the resurrection operator replaces the least fit string in that generation with the most fit string in memory.

Boeringer *et al.* [21] discuss a real valued GA with dynamic crossover and mutation probabilities. This GA with a dynamic parameter adjustment strategy is used to optimize the array weights for array pattern synthesis. Their GA uses tournament selection mate pairing, positive non-integer crossover, and mutation with a variable range. Positive non-integer crossover means that a number at random is chosen to define the probability that one-point, two-point, and three-point crossovers are used. Mutation range is defined as how far a mutated element value is modified from its original value. Their supervisory algorithm prevents premature convergence by dynamically adjusting parameters (mutation range, mutation rate, and number of crossovers) in directions that give the most cost improvement. In an amplitude-only array synthesis problem, their GA with dynamic parameter adjustment strategy converges in 100 iterations compared to over 1000 iterations with a static parameter GA with two crossovers, small mutation rate, and small mutation range. A static parameter GA with full mutation rate, full mutation range, and two crossovers does

not converge in over 10,000 iterations.

Lee *et al.* [51] simulate a phase-only null steering beamformer optimized by a GA. The beamformer is a linear array with 6-bit phase shifters and 100 isotropic antenna elements. The GA maximizes the output SIR as its fitness function, and it uses the first three least significant bits of phase shifters with odd phase shifts to create nulls. They successfully nulled two jammers with the mainbeam at 0deg, the first jammer 2.8deg away from the mainbeam, and the second jammer 4deg away from the mainbeam. They compare their initial simulations with a 20 element patch antenna simulated in a 3-dimentional electromagnetic simulator and saw comparable results.

Ares *et al.* [52,53] compare the performance of genetic algorithms and simulated annealing in optimizing the array weights that synthesized sum and difference patterns of an $N+1$ element array with Doplh-Chebyshev. Sum and difference patterns fill in the nulls creating a pattern that has null depths 5 dB less than the SLLs. Because there are 2^{N} possible weight combinations, an exhaustive search is unfeasible, and they find that genetic algorithms and simulated annealing are applicable in solving this problem. Their simulations show that both algorithms solve this problem with the GA having faster convergence times than simulated annealing. They also show in [53] that convergence times increases exponentially with number of array elements N for exhaustive search while GA convergence times stay relatively flat. This occurs because the GA does not need to search the entire parameter space to find optimal

solutions.

2.6 Beamforming using Simulated Annealing

Other researchers have approached the beamforming problem through the use of simulated annealing in terms of optimizing array layouts as well as optimizing array weights. For example, Evans *et al.* [30] optimize the feed of a 2 x 2 patch antenna using simulated annealing, and Sadler [54] implements a version of simulated annealing with a dual temperature schedule to optimize array layout of a direction finding (DF) antenna array. The DF array is optimized to reduce the number of ambiguous angles (i.e., sidelobes or grating lobes) that prevent the array from finding a target signal. Sadler defines an ambiguity function that accounts for all possible steering directions of the array within its upper hemisphere. The simulated annealing algorithm minimizes a cost function that includes both the ambiguity function and a weighted element distribution function that accounts for imperfect distribution of elements over the array space.

Sadler's two stage temperature schedule in [54] is selected such that once that initial convergence is detected, the temperature schedule is changed to one with a lower temperature. This allows small improvements to the array layout because higher cost solutions are discarded with low temperatures. The array elements positions are also perturbed by picking a random number from a uniform distribution. Prior to convergence, the element positions are perturbed by a maximum of $\pm 0.1\lambda$, and after convergence the maximum disturbance is limited to $\pm 0.01\lambda$.

In this manner, Sadler [54] shows that a seven element concentric circular array arranged in two rings (with three elements on an inner ring of radius 1.5λ and four elements on an outer ring of radius 4.5λ) has 36 total ambiguous peaks with values greater than 4.0 dB with the largest ambiguous peak being 4.5 dB high. The target signal has a peak value of 10 dB. This is a significant improvement to a reference circular array with one element in the center and seven elements on a radius of 4.5λ. The reference array has a total of 70 ambiguous peaks with amplitude greater than 4.0 dB, and the largest ambiguous peak has an amplitude of 5.0 dB.

Dong *et al.* [31] considers the case of optimizing MIMO antenna element positions to create the largest possible virtual array given a set of M transmitting and N receiving elements. The M transmitted waveforms are orthogonal, and not all element position combinations maximized the virtual array's aperture. They combine simulated annealing with cyclic difference sets (CDS) to reduce the search space and increase the algorithm's convergence time. They constrain the locations of one array's (receiving or transmitting) as a CDS. For $M = 3$, $N = 5$, and the largest contiguous aperture of the virtual array $L = 63$ calculated by an exhaustive search algorithm, they reduce the search space from over 1 billion to roughly 2000 combinations by using a CDS. They showed that the SA alone converges in over 30,000 iterations while the CDS constrained SA converges in less than 250 iterations. With the receive elements constrained by a CDS, their hybrid algorithm creates transmit arrays with positions that generate $64 \leq L \leq 72$.

The simulated annealing algorithm in [31] minimizes a cost function that includes a term to minimize the total user transmit power and two penalty terms. The first penalty term accounts for individual BERs that are greater than the maximum permitted BERs, and the second penalty term accounts for individual transmit powers that are greater than permitted powers. Their simulated annealing algorithm creates solutions that meets all of the individual BER constraints with a total transmit power of 73.4 W. However, their reference gradient search algorithm does not meet any of the BER constraints despite obtaining a total transmit power of 64.1 W. Two version of their optimizer meet the three BER constraints, but the total transmit powers are greater than simulated annealing meaning that their optimizer found suboptimal solutions.

Pascual-Iserte *et al.* [55] optimizes the weights of multiple beamformers using simulated annealing and compare their results to a Lagrange gradient search optimizer and an alternate and maximize (AM) optimizer. Their simulated beamformers operate in a multiuser MIMO-OFDM system. They optimize the beamformers to minimize total user transmit power. They constraine individual user Bit Error Rates (BERs) to be no greater than 10^{-3}, 10^{-3}, and 10^{-2} and add an optional constraint on individual transmit power (i.e., $P_T^i \leq P_{max}^i$). The problem is non-convex due to the constraint set.

2.7 Antenna Array Hardware Fault Detection, Recovery, and Localization

Detection of hardware faults in an antenna array is necessary for the correction of such faults. Migliore [56] develops a technique for detecting array faults by evaluating the near field of a N element array under test (AUT) and placing M probes with effective heights $h(\theta, \phi)$ at distances $\bar{r}_{\mathrm{m}}$, $m \in [1, M]$ from the AUT's origin. Each antenna element has complex excitation $\bar{x}_{\mathrm{n}}$, $n \in [1, N]$ with complex electric-field radiation patterns $\bar{f}_{\mathrm{n}}(\theta, \phi)$. The voltage at the probe outputs can be expressed as a linear system of equations

$$\mathbf{A}\hat{\bar{x}} \;=\; \hat{\bar{y}} \tag{2.21}$$

where

$$\hat{\bar{x}} \;=\; (\hat{x}_1, \hat{x}_2, \cdots, \hat{x}_{\mathrm{N}})^{\mathrm{T}} \tag{2.22}$$

$$\hat{\bar{y}} \;=\; (\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_{\mathrm{M}})^{\mathrm{T}} \tag{2.23}$$

$$\mathbf{A} \in \mathcal{C}^{\mathrm{M}\times\mathrm{N}} \; s.t. \; A_{\mathrm{mn}} \;=\; \frac{e^{-j\beta r_{\mathrm{m,n}}}}{4\pi r_{\mathrm{m,n}}} \hat{f}(\theta_{\mathrm{m,n}}, \phi_{\mathrm{m,n}})\, h(\theta_{\mathrm{m,n}}, \phi_{\mathrm{m,n}}) \tag{2.24}$$

$$r_{\mathrm{m,n}} \;=\; |r_{\mathrm{m}} \;-\; r_{\mathrm{n}}| \tag{2.25}$$

and $\theta_{\mathrm{m,n}}, \phi_{\mathrm{m,n}}$ are relative angles between the n^{th} antenna element and the m^{th} probe. The matrix elements of (2.24) is similar to an array factor calculation except that the system formed by (2.21) is formed in the near-field and $M \neq N$ whereas the array factor calculation is an approximation that assumes the measuring points are in the far-field with $M = N$ probes having zero effective height. In [56] it is shown that inverting the system of (2.21) required that $M \geq N$.

Migliore also notes that the number of faults $K \ll N$, and their goal is to identify the faults with $M \ll N$. Their technique assumes a reference array to

compare with the AUT, as the technique needed knowledge of the fault free array's element excitations. Migliore labels the reference array's excitation and probe vectors $\hat{\bar{x}}^{\mathrm{r}}$ and $\hat{\bar{y}}^{\mathrm{r}}$ and of the AUT $\hat{\bar{x}}^{\mathrm{d}}$ and $\hat{\bar{y}}^{\mathrm{d}}$. With these definitions, they re-form the system described by equation (2.21) by replacing equations (2.22) – (2.23) with

$$\hat{\bar{x}} \;=\; \hat{\bar{x}}^{\mathrm{r}} \;-\; \hat{\bar{x}}^{\mathrm{d}} \tag{2.26}$$

$$\hat{\bar{y}} \;=\; \hat{\bar{y}}^{\mathrm{r}} \;-\; \hat{\bar{y}}^{\mathrm{d}} \tag{2.27}$$

As such, Migliore notes that the fault detection and localization problem in the presence of noise becomes a constrained minimization problem described by

$$\min_{\mathrm{x}} ||\hat{\bar{x}}||_1 \; \textit{subject to} \; ||\mathbf{A}\hat{\bar{x}} \;-\; \hat{\bar{y}}||_2 \;\leq\; \epsilon \tag{2.28}$$

where $\epsilon \ll 1$. Migliore notes that (2.28) is convex with a unique minimum. The problem is in determining the minimum number of measurement probes necessary to create a stable ℓ_1 norm in the calculation of (2.28). This problem is generally solved by properly choosing the measurement probe locations such that the probes are far enough away from each other to prevent spatial correlation between their electromagnetic fields. They also comment that the number of probes $M \;=\; \mathcal{O}\left(K \log\left(N\right)\right)$.

Migliore [56] evaluate their method on a simulated $N \;=\; 33 \times 33$ planar array with $\lambda/2$ regular lattice and Chebyshev element excitation. They setup $M = 36$ measurement probes and -35 dB noise levels. Two test cases are considered: no faults and five elements faulted with a π radians phase shift. The reference arrays have the same element excitations as the AUTs. In both cases, their method recover the excitation values with 36 measurement probes showing that it can detect and localize

faults in the array. Because the faulty reference array contains the same faults as the AUT, this method works best when the fault locations are correctly hypothesized.

The detection of faults in an antenna array can be performed with the AUT in an environmental test chamber. The idea is to detect occurrences of faults after an AUT is subjected to a series of environmental tests. Chamberlain [57] develops a technique that involved exciting individual antennas of a corporate-fed patch antenna array with impulses. A narrow width pulse is fed into the corner of patch antenna, and the time-varying voltage at the antenna's output is measured on an oscilloscope. Chamberlain's method is useful for testing open-circuits within the array including missing resistors that caused opens within Wilkinson power dividers. They note that open-circuits within Wilkinson power dividers are difficult to detect via return loss measurements because reflections are dissipated by internal isolation resistors.

Chamberlain develops time-domain circuit models of 600 MHz and 1250 MHz resonant patch antennas as well as the 5×5 antenna array. They compare the impulse responses of individual patch antennas with and without faults, and the faults cause the impulse response amplitudes to drop more than 50%. This serves as the criteria for detecting faults in the AUT. They also note that the time-domain impulse response contained all voltage vs. frequency information whereas S-parameter measurements of individual antennas with a VNA contain a low-frequency cutoff. They note that S-parameter measurements produce poor results at low frequencies because antennas and feed-networks are poorly matched at those frequencies causing S_{11} to be large

and S_{21} to be small. This results in high sensitivity of S_{21} to small changes in the antenna attachment configuration, and it increases the risk of false alarms [57].

Once a fault is detected, the array weights needed to be retuned to regain the desired array performance. Liu [58] develops a deterministic approach to array hardware fault recovery that is used in the beamforming array described by [36]. Liu's method minimizes a performance metric P such that

$$\nabla_{\bar{w}^*} P \;=\; 2\,\mathbf{I}\,(\bar{w} \;-\; \bar{w}_{\mathrm{o}}) + 2\mu\,\mathbf{C}\bar{w} \;=\; 0 \tag{2.29}$$

$$\bar{w} \;=\; (\mathbf{I} \;+\; \mu\,\mathbf{C})^{-1}\,\bar{w}_{\mathrm{o}} \tag{2.30}$$

where

$$P \;=\; (\bar{w} \;-\; \bar{w}_{\mathrm{o}})^{\mathrm{H}}\,(\bar{w} \;-\; \bar{w}_{\mathrm{o}}) \;\;+\;\; \mu\,\bar{w}^{\mathrm{H}}\mathbf{C}\bar{w} \tag{2.31a}$$

$$\mathbf{C} \;\in\; \mathcal{C}^{\mathrm{NxN}}\; s.t.\; c_{\mathrm{mn}} \;=\; e^{j\beta(d_n - d_m)\sin(\theta_o)}\mathrm{sinc}\left[\beta\,(\mathrm{d_n} - \mathrm{d_m})\,\mathrm{e}\right],\;\; \mathrm{m,n} \in [1,\mathrm{N}] \tag{2.31b}$$

N is the number of elements in a linear array, and the term e defines a range of angles outside the mainbeam angle θ_o whose SLLs are to be minimized. The weight vector $\bar{w}_{\mathrm{o}}$ represents the quiescent array weights including element faults, and the vector $\bar{w}$ represents the corrected weight vector that needs to be calculated. [58] comment that the first term in (2.31a) minimizes perturbations in the mainbeam while the second term minimizes average power in the side lobes. (2.30) can be calculated using a computer to find $\bar{w}$. The drawback to this method is that the antenna faults need to be localized (i.e., in the definition of $\bar{w}_{\mathrm{o}}$) before $\bar{w}$ can be calculated.

The genetic algorithm has been used to resynthesize an array pattern after element failures. Yeo *et al.* introduce new crossover operators and resynthesize a 32 element Dolph-Chebyshev array with -35 dB SLL with two and three element failures

[22]. Their simulations show that their GA recovered the array pattern with two failures with less than 2 dB error in convergence after 100 generations. Their GA recovers the array pattern with three failures with roughly 6.5 dB in 200 generations. They note that both amplitude and phase weights are required to compensate for element failures. Han *et al.* uses an adaptive weighted beam pattern mask to resynthesize the beam pattern with a GA after a single element failed [23].

Joler [59] develops a self-recovery algorithm (SRA) based on a GA. The radiation pattern of 4 x 4 array without faults is computed, and an external detector flagged a flawed array element. The SRA computed the average error (e) between healthy and flawed radiation patterns. If e is greater than a given tolerance level (tol), the SRA recalculates element amplitude and phase weights until a solution with $e < tol$ is found. They implemente the SRA on a computer and on a FPGA to create an autonomous system that can be used to monitor an antenna array's health.

Mitilineos and Capsalis [60] develop a GA optimization technique that incorporates the probability of element failure in the design of an eight element switched-beam UCA. Their technique redistributes the array weights for balanced performance after element failure during the array's design by anticipating which element failure is most likely to cause worst case post-failure performance. Although their procedure slightly degrades the normally operating array, the worst case post-failure radiation pattern compensated for the element failure.

Oliveri *et al.* [14] develop a technique that uses Bayesian compressive sensing

to localize faults in linear arrays. Their compressive sensing technique is an ℓ_1-norm minimization tool used to retrieve the sparse failure vector $\underline{f}$. They defined this failure vector as the failure-free reference array and the array under test. They calculate the posterior probability $P\left(\underline{f}|\underline{F}\right)$ where $\underline{F}$ is the difference field measurements that is defined as the difference between an error-free N-element linear array and a measured array under test with failed elements. The vector that contains the detected faulted elements is defined as

$$\underline{\hat{f}} \;=\; \arg\max_{\underline{f}} \left\{ P\left(\underline{f}|\underline{F}\right) \right\} \tag{2.32}$$

They use Bayes' theorem to solve $P\left(\underline{f}|\underline{F}\right)$ for the purposes of calculating the solution to (2.32) given the likelihood of a difference field given a failure vector $P\left(\underline{F}|\underline{f}\right)$ and the priors $P\left(\underline{F}\right)$ and $P\left(\underline{f}\right)$. [14] also defines two metrics for the error in detecting and localizing array faults: SNR and the normalized diagnosis error ξ. They compute their approach's sensitivity on the solver's initialization state η and its angular sampling ratio ν (i.e., ratio of number of angular points K to the number of elements N) vs. ξ and various SNR values.

Their results show that there is no optimum for ξ vs. η because each curve for given SNR value have its own minimum in ξ. However, they calculate and successfully use a compromise value η_{opt} based on the average value of $\eta_{\text{best}\rfloor\text{SNR}} = \arg\min_{\eta}\left\{\xi\left(\eta, SNR\right)\right\}$ over $20\,dB \leq$ SNR $\leq 60\,\text{dB}$. They also find that diagnosis error (based on ξ vs. ν for several SNR values) depend non-negligibly on ν without any minima or maxima over $0.85 \leq \nu \leq 1.15$, thus they calculate a compromise

value ν_{opt} in a manner similar to η_{opt}.

[14] also defines a confidence level vector $\underline{\gamma}$ based on (2.32) whose elements γ_{n}, $n \in [1, N]$ measure the reliability of the fault diagnosis $\hat{f}_{\text{n}}$ for each of the N elements. They define a total confidence level Γ where

$$\Gamma \;=\; \frac{1}{N}\sum_{n=1}^{N}|\gamma_{\text{n}}|^2 \tag{2.33}$$

and smaller values of Γ indicated more reliable solutions in terms of $\underline{\hat{f}}$. Their simulations show that low values of total confidence level Γ corresponded to low diagnosis errors ξ. However, their BCS method requires SNR = 60 dB to correctly diagnose array faults with $\Gamma \;=\; 6.78 \times 10^{-4}$ and $\xi \;=\; 8.93 \times 10^{-5}$ for a Dolph 100 element array with -30dB PSLLs, but their method incorrectly diagnoses array faults when SNR = 20 dB with resulting $\Gamma \;=\; 1.23 \times 10^{-2}$ and $\xi \;=\; 4.79 \times 10^{-1}$ for that array.

Rivera *et al.* [61] implement a reconfigurable antenna array with vertical and horizontal microstrip sensing lines to localize antenna faults. The antenna array consists of four patch antennas arranged 2 x 2. Each patch antenna is divided into two sections with two diagonal slots that are disconnected in the patch's center, and they place switches at each end to bridge the upper and lower corners of the patches. The ends of the four sensing lines are connected to a VNA. The insertion loss of all possible switch configurations is collected at initial design and stored in a lookup table. A switch fault is detected and localized by turning all switches on and remeasuring the insertion losses of the sensing lines. The switch fault would be indicated by the resulting sensing line pair's insertion loss that differed from the lookup table values.

Iglesias *et al.* [62] combine case based reasoning (CBR) and GAs to find faulty elements in moderate and large linear arrays. CBR involves storing previously found solutions (i.e., cases) to find solutions for new radiation patterns. Their method collects two groups of cases. The first group contains radiation patterns for the i^{th} antenna that is active, and the second group contains patterns with a faulty i^{th} antenna. For every new radiation pattern $\mathbf{x}_{\text{new}}$, their implementation of CBR calculates the average distances between $\mathbf{x}_{\text{new}}$ and old radiation patterns $\mathbf{x}_{\text{k}}$ for the sets containing element i that work properly and the sets where element i becomes faulty.

Using the difference of these two average distances, they calculate a failure vector that contains the possibility of element $i \in [1, N]$ being faulty. They notee that these calculations reduce the number of element that can be faulty, the use of a GA can reduce the computational cost of verifying which elements failed, and the failure vector can reduce the number of antenna combinations to be checked because the current cases can be stored for future comparisons.

[62] also notes that there are two types of CBRs. The first type uses all available cases to propose solutions for new radiation patterns. The second type uses a self-organizing map (SOM) neural network that stores a small subset of cases most similar to the current radiation pattern, and it uses that small subset to propose solutions for new radiation patterns. Their CBA-GA uses a SOM neural network that requires training with a set of reference vectors to adapt the SOM neural network to the prob-

lem. They name the neuron with the smallest Euclidean distance ($||\mathbf{x}_\mathrm{k} - \mathbf{w}_\mathrm{k}||$) to the current radiation pattern x_k the best matching unit (BMU). The BMU reference vectors (and neurons topographically close to them) approach the neural weight learning function

$$\mathbf{w}_i\,(t+1) \;=\; \mathbf{w}_i\,(t) \;+\; \alpha\,(t)\,[\mathbf{x}_\mathrm{k}\,(t) \;-\; \mathbf{w}_\mathrm{k}\,(t)] \tag{2.34}$$

where i represents the BMU of $\mathbf{x}_\mathrm{k}$ or one of its neighbors, $\alpha\,(t)$ is a learning coefficient, and t represents the current time step.

The result of training the neural network as noted by [62] is to produce a set of reference vectors $\mathbf{w}_\mathrm{k}$ that approaches the PDF of the current radiation patterns x_k. After training, every new radiation pattern $\mathbf{x}_\mathrm{new}$ is compared to set of reference vectors to determine the new BMU. They then reduce the number of cases to be compared by considering only BMUs or their neighboring neurons. This reduced set of cases is called a proposed solution. The proposed solution is fed into a GA that finds a final solution that contains a vector of faulty elements given the new radiation pattern.

They test both types of CBR plus GA on a simulated 100 element linear array tuned to a flattop radiation pattern that have asymmetric SLLs of -25 dB and -20 dB. CBR type 1 has over a 91% success ratio with 24.18 average faulty candidates and 13,120 cases to find the solution when the noise level is 2 dB. For a 30 element linear array with the same radiation pattern, their method's success rate increases to 98% with 25.62 average faulty elements and 12,900 test cases at a 2dB noise level. The

CBR type 2 (SOM neural network) has an 86.6% success ratio with 49 neurons for the 30 element array. It has 30.8 average faulty candidates and uses roughly 1487 cases to find the solution. A 100 neuron SOM has a 83.7% success ratio with 26.2 average faulty candidates and 857 cases needed to find the solution. Increasing the number of neurons decreases the success rate while simultaneously reduces the number of cases to find the solution [62].

Yeo *et al.* develop a technique for detecting and localizing failed antenna array elements using support vector machines [63]. Considering only full functionality and total element failures, they creat a table of all possible 256 failure combinations for an eight element linear array with Chebyshev weights. They consider non-homogeneous polynomials with order d, Gaussian radial basis functions (RBFs), and heavy-tailed RBFs as the kernels for training and testing of their algorithm. They show that linear, second-order, and third-order polynomial RBFs outperform the Gaussian RBF in terms of classification accuracy vs. SNR with all three polynomial RBFs having similar accuracy for SNR $\geq$ 10 dB. They perform simulations of their classifier and compare received patterns with failed elements for $0\,\mathrm{dB} \leq \mathrm{SNR} \leq 25\,\mathrm{dB}$. They note that although the noisy received patterns no longer resembles the original failed pattern for $\mathrm{SNR} \leq 10\,\mathrm{dB}$, their classifier with a linear polynomial RBF still recognizes the failure with an accuracy greater than 90% at 0 dB SNR.

Rodríquez-González *et al.* [64] create a modified GA algorithm for locating multiple faults in an N element planar array. They consider only complete element

faults with zero excitation when an element faults. Without mutual coupling effects considered, they derive the fully-functional array's radiation pattern using the product of array factor with element factor. They call this N element sum $F(\theta, \phi)$ and note that the defective array's pattern can be calculated as

$$F_{\mathrm{C}}(\theta, \phi) \;=\; F(\theta, \phi) - \sum_{n \in \mathrm{C}} \left[F(\theta, \phi) \;-\; F_{\mathrm{C}}(\theta, \phi) \right] \tag{2.35}$$

where $\{F_{\mathrm{C}}(\theta, \phi)\}_{n \in \mathrm{C}}$ represents radiated patterns with only the n^{th} element faulted. They measure the pattern of the defective array $F_{\mathrm{D}}(\theta, \phi)$ by arranging M points in directions $(\theta_{\mathrm{m}}, \phi_{\mathrm{m}}), m \in [1, M]$. Their GA reconfigures the array weights to minimize the squared distance between the measured pattern and the configured pattern as

$$d_{\mathrm{C}} \;=\; \sum_{m=1}^{M} \left[F_{\mathrm{D}}(\theta_{\mathrm{m}}, \phi_{\mathrm{m}}) \;-\; F_{\mathrm{C}}(\theta_{\mathrm{m}}, \phi_{\mathrm{m}}) \right]^2 \tag{2.36}$$

Their GA objective function[2] as described by equation (2.36) makes their modified GA more effective than other researchers' GAs at solving this problem because (2.36) tabulates both the fault-free pattern as well as the N single fault patterns through equation (2.35) and the definition of $F(\theta, \phi)$. They note that GAs implemented by other researchers require array factor patterns to be calculated for every array configuration whether fault-free, single-fault, or multiple-fault. Their GA differs from other GA implementations through a more efficient means of calculating radiation patterns for faulty arrays. They simulate their GA fault finding algorithm on a 100-element linear array consisting of $\lambda/2$ dipoles with $\lambda/2$ spacing that is positioned

[2] Although Rodríquez-González *et al.* does not specify their GA's fitness function in [64], it most likely takes the form of $F_{\mathrm{GA}} \;=\; {}^{1}/_{(d_{\mathrm{C}} \,+\, \epsilon)}$ where $\epsilon \ll 1$.

$\lambda/4$ wavelengths in front of a ground plane. For the fault-free array, they synthesize a pattern with 1.2° -3 dB beamwidth using a linear Taylor distribution with order $\bar{n} = 12$ and -25 dB SLLs.

They compare the results of their GA with exhaustive search both without and with mutual coupling between elements for 50 amplitude-only test patterns. Without mutual coupling, exhaustive search successfully reconfigures the array 100% of the time with over 4-million evaluations in 14.3 seconds. The GA developed by [64] is successful 99.9% of the time with 322,336 evaluations in 4.3 seconds. They note that a conventional GA solves the problem 99.9% of the time with 313,279 evaluations in 175.7 seconds. With mutual coupling included, exhaustive search solves the problem 98.8% of the time with the same number of evaluations and run-time. However, their GA solves the problem 97.9% of the time with 389,368 evaluations in 5.2 seconds. A conventional GA solves the problem with mutual coupling 99.8% of the time with 379,297 evaluations in 965.8 seconds.

2.8 Summary

The assumptions in solving the anti-jamming beamforming with hardware fault recovery problem are discussed in this chapter, and previous research in optimizing beamforming arrays and hardware fault recovery is reviewed. Because the research focus is on arrays for wireless communications links, it is assumed that the signal (both desired and jammer) directions are not known *a priori*. The wireless channel

is also assumed to be time varying, and signal DOAs can be either time varying or static thus requiring an adaptive solution. The problem is combinatorial because the complex array weights are digitized to create a finite search space. This rendered classical optimization algorithms such as LMS impractical.

Much of the theory behind canonical beamsteering and weighting is discussed, including methods for ULAs, UCAs, and CCAs. Investigations of Dolph-Chebyshev amplitude weighting to reduce SLLs in UCAs and CCAs are performed by [38–42]. Beampattern synthesis is an extension of the classical array weighting methods (i.e., Uniform, Binomial, Dolph-Chebyshev), and research in synthesizing array patterns is summarized including research performed by [43].

Gross summarizes traditional gradient search beamforming methods including LMS, MVDR, SMI, and CGM in [9], and Van Trees discusses these methods further in [2]. Improvements to these algorithms (assuming signal known DOAs) are discussed in [8,45–48]. Research in beamforming using genetic algorithms is discussed in [21,49–53] and using simulated annealing in [30, 31, 54, 55].

Previous research in hardware fault detection and recovery in antenna arrays is also discussed in this chapter. Hardware faults prevent the array from anti-jamming, and it is necessary to recovery anti-jamming functionality through software algorithms. The goal of fault detection and recovery is to extend an array's MTBF beyond that of its individual components. It is also important to localize the fault, as doing so can not only allow the recovery algorithm to isolate it from the system,

fault localization can serve as a means of reducing the system's MTBR. The fault localization algorithm can record fault locations in a look up table, and field engineers can use it to aid in system repair and thus reduce system downtime. The location of the faulted element is also important in the sense that physical locations of elements played a role in their effects on array performance [65, 66].

Research in hardware fault detection focuses on injecting signals into the system and measuring the system response [36, 56, 57]. There has been research in fault localization that implies fault detection [14, 61–64], but this research assumes that the array patterns of fully functional systems are known or can be measured with signal directions known *a priori*. Research in fault recovery algorithms either rely on localization of faults in order to recover functionality [58], use pattern masks to re-tune array weights [22,23], use external fault detectors [59], or incorporate element probability of faults in the array design to minimize the effects of faulty elements during system operation [60]. To the best of the author's knowledge, state-of-the-art systems lack the ability to detect, localize, and recovery from array faults while the system operates as parts of wireless communication links.

Chapter 3

New Models for Phased Antenna Array Anti-Jamming

This chapter discusses new models for phased antenna arrays used to anti-jam interference. These models serve as a theoretical basis for evaluating the performance of stochastic algorithms in performing anti-jamming beamforming under various situations such as stationary signals in fading wireless channels. The models apply for N element antenna arrays of arbitrary layouts.

Section 3.1 defines the problem formulation and setup for performing anti-jamming beamforming using an N element array with an arbitrary layout. Section 3.2 discusses the derivation of complex array weight solutions given a known set of SOI and interference DOAs. Although we generally assumed that signal DOAs are unknown prior to implementing the optimization algorithms, it is useful to derive array weight solutions as a means of evaluating algorithm performance. Section 3.3 discusses the models for stationary signal beamforming and anti-jamming in fading wireless channels. Section 3.4 discusses the models for anti-jamming mobile signals, and section 3.5 discusses methods for showing equivalencies between different electromagnetic simulation types used in stochastic optimization.

3.1 Anti-Jamming Beamforming Problem Formulation and Setup

The problem formulation is based on an N element antenna array as shown in Figure 3.1. The element position vectors $\{\bar{d}_i\},\ i \in [1, N]$ are referenced to the coordinate, and each element has a complex weight $\{\hat{a}_i\},\ i \in [1, N]$. In general, two types of beamforming arrays exist: passive and active. Passive arrays assume that the amplitudes have values $0 \leq |\hat{a}_i| \leq 1,\ \forall\, i$, and active arrays assume that $0 \leq |\hat{a}_i| \leq A_{max}$ where $1 < A_{max} < \infty$. In terms of beamforming array solutions, passive arrays constrain the complex weight search space to an area defined by the unit circle in complex space, and active arrays require the use of amplifiers to extend the solution search beyond the unit circle.

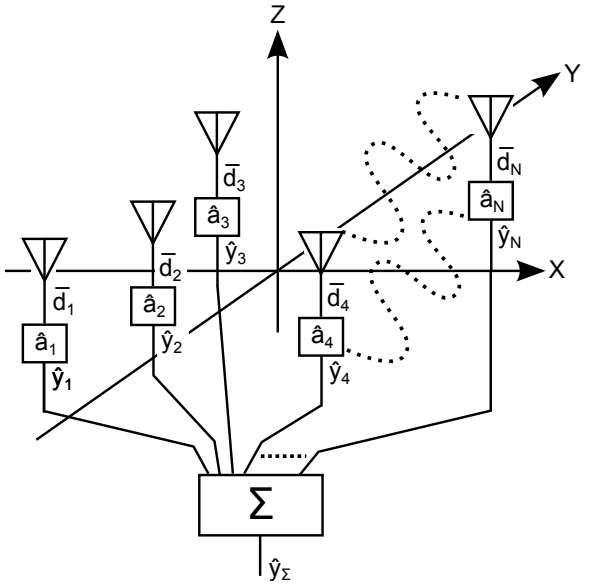


Figure 3.1: Diagram of a N-Element antenna array with arbitrary layout.

The array factor equation discussed in Section 2.2 is repeated below for reference

$$\hat{A}\left(\theta,\phi\right) \;=\; \sum_{i=1}^{N} \hat{a}_{\mathrm{i}} e^{j\beta \bar{a}_{\mathrm{R}}(\theta,\phi)\bullet \bar{d}_{\mathrm{i}}} \tag{3.1}$$

where

$$\bar{a}_{\mathrm{R}}\left(\theta,\phi\right) \;=\; \bar{a}_{\mathrm{x}} \sin\left(\theta\right)\cos\left(\phi\right) \;+\; \bar{a}_{\mathrm{y}} \sin\left(\theta\right)\sin\left(\phi\right) \;+\; \bar{a}_{\mathrm{x}} \cos\left(\theta\right) \tag{3.2}$$

$\beta \;=\; 2\pi/\lambda_o$ is the free-space wavenumber, and λ_o is the free-space wavelength. Although the array factor does not include mutual coupling effects, it is a good first order approximation to an N-element antenna array's pattern. The array factor is a sum of individual antenna element positional contributions to the total electric field in far-field multiplied by the individual element's complex weight. Derivation of the array-factor is discussed in [3,37,67] The array factor (3.1) can be expressed in vector notation as

$$\hat{A}\left(\theta,\phi\right) \;=\; \hat{\bar{a}}^{\mathrm{T}} \cdot e^{j\beta\, \bar{A}_{\mathrm{R}}(\theta,\phi)} \tag{3.3}$$

where

$$\hat{\bar{a}} \;=\; \left[\hat{a}_1, \hat{a}_2, \ldots, \hat{a}_{\mathrm{N}}\right]^{\mathrm{T}} \tag{3.4a}$$

$$\bar{A}_{\mathrm{R}}\left(\theta,\phi\right) \;=\; \left\{\bar{a}_{\mathrm{R}}^{\mathrm{T}} \cdot \mathbf{D}^{\mathrm{T}}\right\}^{\mathrm{T}} \;=\; \mathbf{D} \cdot \bar{a}_{\mathrm{R}}\left(\theta,\phi\right) \tag{3.4b}$$

$$\bar{a}_{\mathrm{R}}\left(\theta,\phi\right) \;=\; \left[\sin\left(\theta\right)\cos\left(\phi\right), \sin\left(\theta\right)\sin\left(\phi\right), \cos\left(\theta\right)\right]^{\mathrm{T}} \tag{3.4c}$$

$$\mathbf{D} \;=\; \left[\bar{d}_1, \bar{d}_2, \ldots, \bar{d}_{\mathrm{N}}\right]^{\mathrm{T}} \tag{3.4d}$$

$$\bar{d}_{\mathrm{i}} \;=\; \left[d_{\mathrm{ix}}, d_{\mathrm{iy}}, d_{\mathrm{iz}}\right]^{\mathrm{T}}, i \in [1, N] \tag{3.4e}$$

It should be noted that $\mathbf{D} \in \mathcal{R}^{\mathrm{N}\times 3}$ contains all of the N antenna element position vectors, the vector dot product $\bar{A}_{\mathrm{R}}\left(\theta,\phi\right) \in \mathcal{R}^{\mathrm{N}\times 1}$ such that the exponential term in (3.3) is $\mathcal{R}^{\mathrm{N}\times 1}$, and $\hat{\bar{a}} \in \mathcal{C}^{\mathrm{N}\times 1}$.

3.2 Derivation of Complex Array Weight Solutions for Given SOI and Interference Directions

Although it is assumed that the SOI and interference directions are not known *a priori*, the complex array weights for an N element antenna array with arbitrary layout given a set of SOI and interference directions are derived in this section. The derivation of array weights that steer a main beam towards the SOI while simultaneously minimizing array gain towards interference directions will serve as a means of comparing the algorithms' performance. The goal is to find $\hat{\bar{a}}$ such that

$$\exists\, \hat{\bar{a}}_0 \rightarrow \mathrm{G_I} \;=\; \max\left\{\hat{\mathrm{A}}\left(\theta \,=\, \theta_\mathrm{o}, \phi \,=\, \phi_\mathrm{o}\right)\right\} \tag{3.5}$$

$$\text{given } \hat{\mathrm{A}}\left(\theta \,=\, \theta_\mathrm{j}, \phi \,=\, \phi_\mathrm{j}\right) \;\leq\; \mathrm{G_j} \;=\; \epsilon,\; \forall \mathrm{j} \in [1, \mathrm{J}]$$

$$\text{where } \epsilon << 1,\; \theta_\mathrm{j} \;\neq\; \theta_\mathrm{o},\; \&\; \phi_\mathrm{j} \;\neq\; \phi_\mathrm{o}\; \forall \mathrm{j}$$

It is assumed that $\{\theta_j\}$ and $\{\phi_j\}$ in (3.5) are linear independent (i.e., J separate angles for J jammers). The problem at hand is essentially one of solving N independent equations to find N complex weights. Namely,

$$\hat{A}\left(\theta \,=\, \theta_o, \phi \,=\, \phi_o\right) \;=\; G_I \;=\; \hat{\bar{a}}^{\mathrm{T}} \,\cdot\, e^{j\beta\, \bar{A}_{\mathrm{R}}(\theta_o, \phi_o)} \tag{3.6}$$

$$\left\{\hat{A}\left(\theta \,=\, \theta_j, \phi \,=\, \phi_j\right) \;=\; G_j \;=\; \hat{\bar{a}}^{\mathrm{T}} \,\cdot\, e^{j\beta\, \bar{A}_{\mathrm{R}}(\theta_j, \phi_j)}\right\},\; j \in [1, J] \tag{3.7}$$

In general, the total number of SOI and interference signals is less than the number of antennas, so equations (3.6) – (3.7) represent a system of $1 \,+\, J \;<\; N$ equations.

Because there are more unknowns than equations in this system, the problem cannot be solved as is. However, $M = N - (J+1)$ arbitrary directions can be chosen with minimal gains $G_m \leq \epsilon \; m \in [1, M]$ such that

$$\left\{ \hat{A}\left(\theta = \theta_m, \phi = \phi_m\right) = G_m = \hat{\bar{a}}^{\mathrm{T}} \cdot e^{j\beta \, \bar{A}_{\mathrm{R}}(\theta_m, \phi_m)} \right\}, \; m \in [1, M] \tag{3.8}$$

where $\theta_m \neq \theta_j$, $\theta_m \neq \theta_o$, $\phi_m \neq \phi_j$, and $\phi_m \neq \phi_0 \, \forall m, j$. One can note that the multimodality of the anti-jamming beamforming problem arises from the multiple values available to $\{G_j \leq \epsilon\}$ depending on acceptable values of ϵ and the possible combinations of complex array weights that achieve those levels when solving (3.5).

Although the choice of $\{\theta_m, \phi_m\}, \; m \in [1, M]$ adds to the problem's multimodality, the problem is still multimodal if $M = 0$ for the reasons noted above. These directions can also be thought of as degree of freedoms since they can be chosen arbitrarily. The problem described by (3.5) with the addition of the M equations described by (3.8) can be restated as

$$\exists \, \hat{\bar{a}}_0 \rightarrow \mathrm{G_I} = \max \left\{ \hat{\mathrm{A}}\left(\theta = \theta_{\mathrm{o}}, \phi = \phi_{\mathrm{o}}\right) \right\} \tag{3.9}$$

$$\text{given } \hat{\mathrm{A}}\left(\theta = \theta_{\mathrm{j}}, \phi = \phi_{\mathrm{j}}\right) \leq \mathrm{G_j} = \epsilon, \; \forall \mathrm{j} \in [1, \mathrm{J}]$$

$$\text{while forcing } \hat{\mathrm{A}}\left(\theta = \theta_{\mathrm{m}}, \phi = \phi_{\mathrm{m}}\right) \leq \mathrm{G_m} = \epsilon, \; \forall \mathrm{m} \in [1, \mathrm{M}]$$

$$\text{where } \epsilon << 1, \; \theta_{\mathrm{j}} \neq \theta_{\mathrm{o}}, \theta_{\mathrm{m}} \neq \theta_{\mathrm{j}}, \; \theta_{\mathrm{m}} \neq \theta_{\mathrm{o}},$$

$$\phi_j \neq \phi_o, \; \phi_m \neq \phi_j, \; \phi_m \neq \phi_0 \; \forall j, \; m$$

$$\text{and } \mathrm{M} = \mathrm{N} - (\mathrm{J} + 1)$$

Equation (3.9) represents an N equation system, and $M = 0$ only if $J + 1 = N$. It is easy to show that these equations can be represents in matrix form as

$$\mathbf{B}\hat{\bar{a}} = \hat{\bar{y}} \tag{3.10}$$

where $\mathbf{B}$ represents the array manifold for one SOI, J jammers, and M arbitrary nulling directions, and $\hat{\bar{y}}$ represents the array outputs in the N specified directions. Both $\mathbf{B}$ and $\hat{\bar{y}}$ can be expressed as

$$\mathbf{B} = \begin{bmatrix} e^{j\beta A_{R,1}(\theta_o,\phi_o)} & e^{j\beta A_{R,2}(\theta_o,\phi_o)} & \dots & e^{j\beta A_{R,N}(\theta_o,\phi_o)} \\ e^{j\beta A_{R,1}(\theta_{j=1},\phi_{j=1})} & e^{j\beta A_{R,2}(\theta_{j=1},\phi_{j=1})} & \dots & e^{j\beta A_{R,N}(\theta_{j=1},\phi_{j=1})} \\ \vdots & \vdots & \ddots & \vdots \\ e^{j\beta A_{R,1}\left(\theta_{j=J},\phi_{j=J}\right)} & e^{j\beta A_{R,2}\left(\theta_{j=J},\phi_{j=J}\right)} & \dots & e^{j\beta A_{R,N}\left(\theta_{j=J},\phi_{J=1}\right)} \\ e^{j\beta A_{R,1}(\theta_{m=1},\phi_{m=1})} & e^{j\beta A_{R,2}(\theta_{m=1},\phi_{m=1})} & \dots & e^{j\beta A_{R,N}(\theta_{m=1},\phi_{m=1})} \\ \vdots & \vdots & \ddots & \vdots \\ e^{j\beta A_{R,1}(\theta_{m=M},\phi_{m=M})} & e^{j\beta A_{R,2}(\theta_{m=M},\phi_{m=M})} & \dots & e^{j\beta A_{R,N}(\theta_{m=M},\phi_{m=M})} \end{bmatrix} \tag{3.11}$$

$$\hat{\bar{y}} = [G_I, G_J, \dots, G_J, G_M, \dots, G_M]^T \tag{3.12}$$

Note that the rows of (3.11) are simply the transpose of (3.4b) applied to specific directions in θ and ϕ. The gain terms in (3.12) are real valued constants, and they are related to the array gain as the square root of array gain. If the matrix $\mathbf{B}$ is invertible (i.e., $\det(\mathbf{B}) \neq 0$), then the complex array weights for a given set of directions (SOI, jammer, arbitrary nulling directions) can be calculated as

$$\hat{\bar{a}}_o \;=\; \mathbf{B}^{-1}\,\hat{\bar{y}} \tag{3.13}$$

The inverse of matrix $\mathbf{B}$ is clearly not invertible for all signals impinging on the array. For that matrix to be invertible, all of the rows in (3.11) must be linearly independent in the phase terms. An obvious example is when an interfering signal coincides with the SOI, as nulling the jammer would also null the SOI, and maximizing the mainbeam gain towards the SOI would also maximize jammer power at the array's output. Other directions are less obvious and are dependent on the array's layout.

3.2.1 Calculating Weights for Uniformly Spaced Linear Arrays

The complex weight solutions for electronically steering ULAs is well known [3, 33], so comparisons of the constant amplitude phase weights $\{\delta_{\mathrm{n}} \;=\; \beta x_{\mathrm{n}} \cos\phi_{\mathrm{o}}\}$ for N-element ULAs with elements placed on the x-axis with the complex weights calculated using (3.13) is a useful performance metric. For example, a 10-element ULA steered towards 90° has the same amplitude and phase weights using uniform amplitude and the canonical $\{\delta_{\mathrm{n}}\}$ form compared to equation (3.13) as shown in Table 3.1 and Figure 3.2. For simplicity, jammer directions are chosen that coincided with the null directions that naturally occur for this phased array. The uniform weighted ULA is indistinguishable from the ULA with complex weights calculated by inverting matrix $\mathbf{B}$, as the amplitude weights differed by $\pm 1\%$, and the phase weights are the same.

The ability of equation (3.13) to find complex array weights is compared with the amplitude and phase weights for a 10-element ULA with uniform amplitude weights steered to 30° as well as Dolph-Chebyshev 10-element ULAs steered to 90° and 30°. The patterns are indistinguishable as shown in Figures 3.3 – 3.5 with comparable weights as listed in Tables 3.2 – 3.4.

Table 3.1: Comparison of complex array weights calculated using (3.13) with uniform amplitude weights and canonical phase steering for $N = 10$ element ULA with mainbeam steered to 90°, $\mathrm{G_I} = \mathrm{N}$, and $\mathrm{G_j} = 0,\ \mathrm{j} \in [1, \mathrm{N}]$.

SOI / Jammer Directions (°)	Uniform Amplitude Reference		Inverse Matrix Calculation	
	Amplitude	Phase (Radians)	Amplitude	Phase (Radians)
90.00 (SOI)	1.00	0.00	0.99	0.00
0.00	1.00	0.00	0.99	0.00
36.88	1.00	0.00	1.00	0.00
53.12	1.00	0.00	1.00	0.00
66.44	1.00	0.00	1.01	0.00
78.44	1.00	0.00	1.01	0.00
101.60	1.00	0.00	1.00	0.00
113.60	1.00	0.00	1.00	0.00
126.90	1.00	0.00	1.00	0.00
143.10	1.00	0.00	0.99	0.00

The graphs shown in Figures 3.2 – 3.5 with parameters listed in Tables 3.1 – 3.4 represent the optimal solutions for those arrays because $\mathrm{G_j} = 0,\ \forall \mathrm{j} \in [1, \mathrm{N} - 1]$. However, it follows that other solutions exist for $\mathrm{G_j} \neq 0,\ \mathrm{j} \in [1, \mathrm{N}, -1\,]$ provided that the signal directions form an invertible array manifold matrix $\mathbf{B}$ as defined by (3.11).

A random number generator in Matlab is seeded to generate a repeatable random

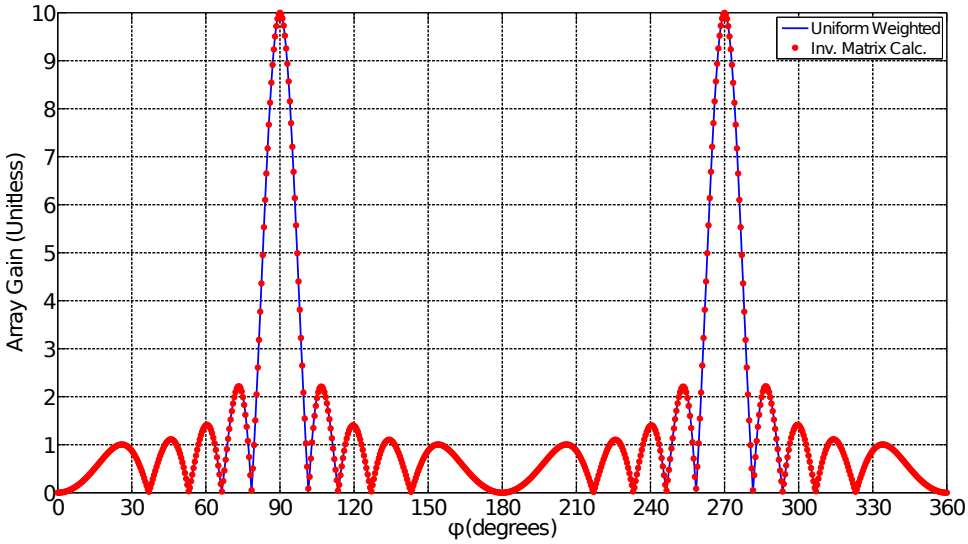


Figure 3.2: Array Factor Patterns for a ULA steered to 90° showing a reference pattern created with uniform weights and pattern with complex weights calculated using equation (3.13).

Table 3.2: Comparison of complex array weights calculated using (3.13) with uniform amplitude weights and canonical phase steering for $N = 10$ element ULA with mainbeam steered to 30°, $\mathrm{G_I} = \mathrm{N}$, and $\mathrm{G_j} = 0,\ \mathrm{j} \in [1, \mathrm{N}]$.

SOI / Jammer Directions (°)	Uniform Amplitude Reference		Inverse Matrix Calculation	
	Amplitude	Phase (Radians)	Amplitude	Phase (Radians)
30.00 (SOI)	1.00	0.00	1.00	0.00
48.25	1.00	$-1 \times \pi\sqrt{3}/2$	1.00	$-1 \times \pi\sqrt{3}/2$
62.19	1.00	$-2 \times \pi\sqrt{3}/2$	1.00	$-2 \times \pi\sqrt{3}/2$
74.56	1.00	$-3 \times \pi\sqrt{3}/2$	1.00	$-3 \times \pi\sqrt{3}/2$
86.19	1.00	$-4 \times \pi\sqrt{3}/2$	1.00	$-4 \times \pi\sqrt{3}/2$
97.69	1.00	$-5 \times \pi\sqrt{3}/2$	1.00	$-5 \times \pi\sqrt{3}/2$
109.50	1.00	$-6 \times \pi\sqrt{3}/2$	1.00	$-6 \times \pi\sqrt{3}/2$
122.20	1.00	$-7 \times \pi\sqrt{3}/2$	1.00	$-7 \times \pi\sqrt{3}/2$
137.20	1.00	$-8 \times \pi\sqrt{3}/2$	1.00	$-8 \times \pi\sqrt{3}/2$
159.10	1.00	$-9 \times \pi\sqrt{3}/2$	1.00	$-9 \times \pi\sqrt{3}/2$

set of $\{\mathrm{G_j}\}$ such that the nulls in reference to the main beam are $-40\ \mathrm{dB_{MB}} < \{\mathrm{G_j}\} < -20\ \mathrm{dB_{MB}},\ \mathrm{j} \in [1, \mathrm{J} = \mathrm{N} - 1]$. For applications where it is acceptable to have nulls at least 20 dB down from the mainbeam, these levels represent acceptable suboptimal

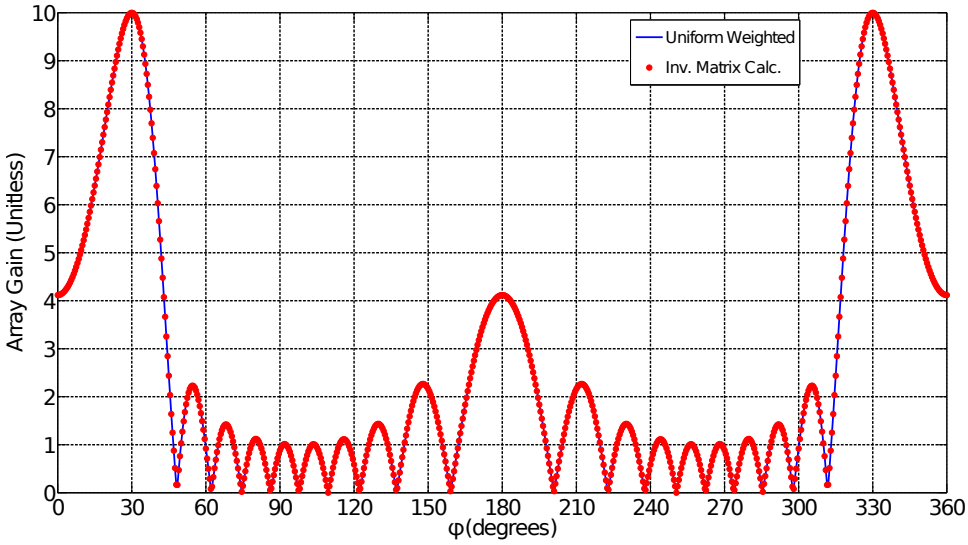


Figure 3.3: Array Factor Patterns for a ULA steered to 30° showing a reference pattern created with uniform weights and pattern with complex weights calculated using (3.13).

Table 3.3: Comparison of complex array weights calculated using (3.13) with Chebyshev amplitude weights for SLL = -20 dB and canonical phase steering for $N = 10$ element ULA with mainbeam steered to 90°, $G_I = 9.54$, and $G_j = 0$, $j \in [1, N]$.

SOI / Jammer Directions (°)	Chebyshev Amplitude Reference		Inverse Matrix Calculation	
	Amplitude	Phase (Radians)	Amplitude	Phase (Radians)
90.00 (SOI)	0.78	0.00	0.78	0.00
0.00	0.72	0.00	0.72	0.00
37.81	0.94	0.00	0.94	0.00
54.31	1.12	0.00	1.12	0.00
67.19	1.21	0.00	1.21	0.00
76.44	1.21	0.00	1.21	0.00
103.60	1.12	0.00	1.12	0.00
112.80	0.94	0.00	0.94	0.00
125.70	0.72	0.00	0.72	0.00
142.20	0.78	0.00	0.78	0.00

solution for the complex array weights.[1]

The suboptimal solution for (3.9) calculated by (3.13) is compared with a uniform

[1]Clearly, this would represent an unacceptable solution for applications that required null depths much greater than -20 dB_{MB}.

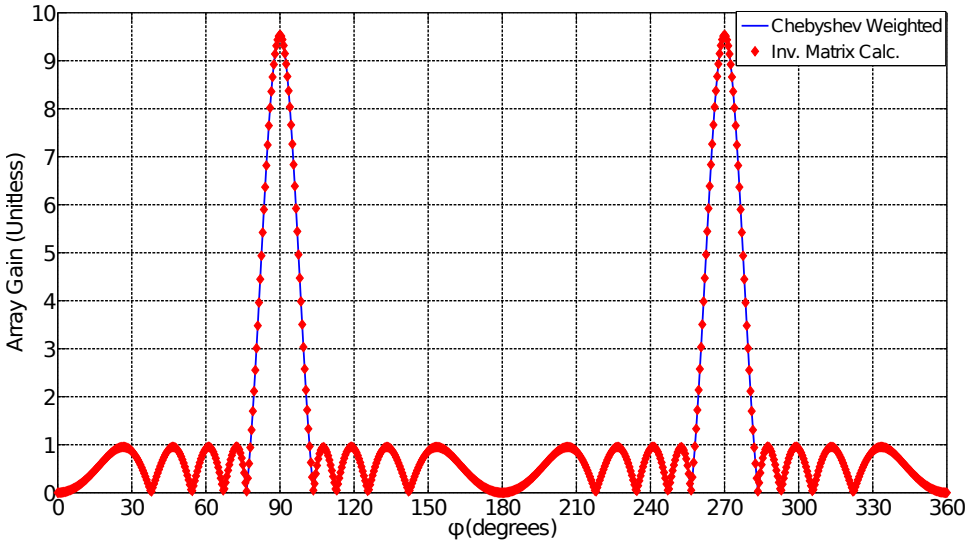


Figure 3.4: Array Factor Patterns for a ULA steered to 90° showing a reference pattern created with Chebyshev weights for SLL = -20 dB and pattern with complex weights calculated using (3.13).

Table 3.4: Comparison of complex array weights calculated using (3.13) with Chebyshev amplitude weights for SLL = -20 dB and canonical phase steering for $N = 10$ element ULA with mainbeam steered to 30°, $G_I = 9.54$, and $G_j = 0,\ j \in [1, N]$.

SOI / Jammer Directions (°)	Chebyshev Amplitude Reference		Inverse Matrix Calculation	
	Amplitude	Phase (Radians)	Amplitude	Phase (Radians)
30.00 (SOI)	0.78	0.00	0.78	0.00
50.88	0.72	$-1 \times \pi\sqrt{3}/2$	0.72	$-1 \times \pi\sqrt{3}/2$
61.44	0.94	$-2 \times \pi\sqrt{3}/2$	0.94	$-2 \times \pi\sqrt{3}/2$
73.56	1.12	$-3 \times \pi\sqrt{3}/2$	1.12	$-3 \times \pi\sqrt{3}/2$
85.62	1.21	$-4 \times \pi\sqrt{3}/2$	1.21	$-4 \times \pi\sqrt{3}/2$
97.69	1.21	$-5 \times \pi\sqrt{3}/2$	1.21	$-5 \times \pi\sqrt{3}/2$
110.10	1.12	$-6 \times \pi\sqrt{3}/2$	1.12	$-6 \times \pi\sqrt{3}/2$
123.40	0.94	$-7 \times \pi\sqrt{3}/2$	0.94	$-7 \times \pi\sqrt{3}/2$
138.20	0.72	$-8 \times \pi\sqrt{3}/2$	0.72	$-8 \times \pi\sqrt{3}/2$
154.10	0.78	$-9 \times \pi\sqrt{3}/2$	0.78	$-9 \times \pi\sqrt{3}/2$

weighted $N = 10$ element ULA steered to 30° with the resulting $\{G_j\}$ levels, reference array weights, and calculated array weights listed in Table 3.5. The array patterns are normalized to the SOI's mainbeam gain to show null depths referenced to the

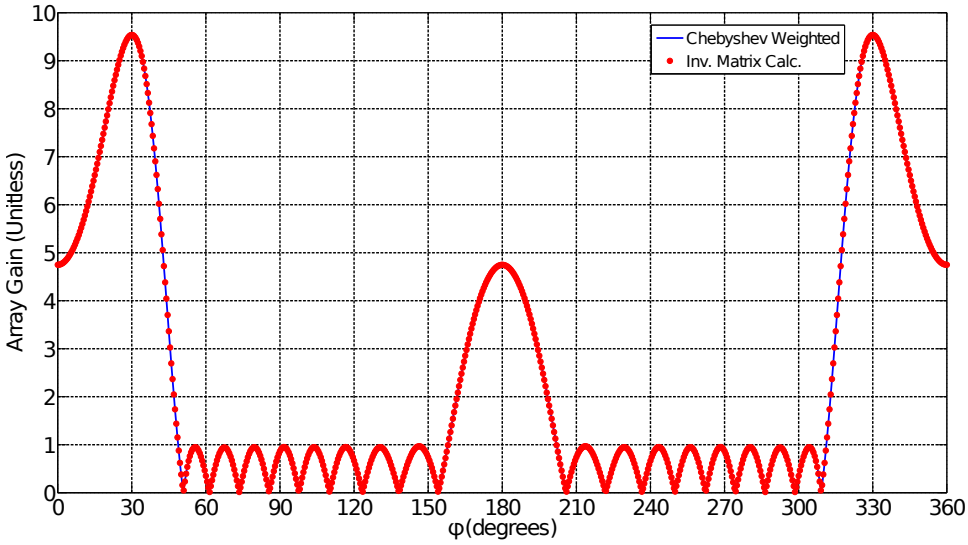


Figure 3.5: Array Factor Patterns for a ULA steered to 30° showing a reference pattern created with Chebyshev weights for SLL = -20 dB and pattern with complex weights calculated using (3.13).

SOI gain in Figure 3.6. The two array patterns are not the same, as the reference weights differ considerably from the weights calculated using (3.13) with the $\{G_j\}$ levels shown. Similar results are also compared for a N = 10 element ULA with interference directions set to those of a Dolph-Chebyshev ULA with SLL = -20 dB as shown in Table 3.6 and Figure 3.7.

3.2.2 Calculating Weights for Circular Arrays

A circular array consists of antenna elements equally spaced on a circle of radius r. Circular arrays can be amplitude and phase steered in manners similar to ULAs. The maximum number of elements on a circular array with radius $r = \lambda_o/2$ is six, as this ensures that the spacing between elements is $d \geq \lambda_o/2$ where λ_o is the free space wavelength. The layout of a six-element circular array for $\lambda_o = 12.5cm$

Table 3.5: Comparison of complex array weights calculated using (3.13) with uniform amplitude weights and canonical phase steering for N = 10 element ULA with mainbeam steered to 30°, $G_I = N$, and $G_j \neq 0,\ j \in [1, N-1]$.

SOI / Jammer DOAs (°)	G_j	G_j (dB_{MB})	Uniform Ampl. Ref.		Inverse Matrix Calc.	
			Ampl.	Phase	Ampl.	Phase
30.00 (SOI)	–	–	1.00	0.00	1.45	0.00
48.25	0.52	-25.64	1.00	$-1 \times \pi\sqrt{3}/2$	1.02	$-1.01 \times \pi\sqrt{3}/2$
62.19	0.47	-26.63	1.00	$-2 \times \pi\sqrt{3}/2$	0.95	$-2.02 \times \pi\sqrt{3}/2$
74.56	0.37	-28.62	1.00	$-3 \times \pi\sqrt{3}/2$	0.98	$-3.00 \times \pi\sqrt{3}/2$
86.19	0.69	-23.17	1.00	$-4 \times \pi\sqrt{3}/2$	0.87	$-1.72 \times \pi\sqrt{3}/2$
97.69	0.13	-37.60	1.00	$-5 \times \pi\sqrt{3}/2$	0.94	$-2.69 \times \pi\sqrt{3}/2$
109.50	0.37	-28.63	1.00	$-6 \times \pi\sqrt{3}/2$	0.87	$-3.66 \times \pi\sqrt{3}/2$
122.20	0.59	-24.63	1.00	$-7 \times \pi\sqrt{3}/2$	0.98	$-4.69 \times \pi\sqrt{3}/2$
137.20	0.41	-27.79	1.00	$-8 \times \pi\sqrt{3}/2$	0.95	$-5.67 \times \pi\sqrt{3}/2$
159.10	0.95	-20.48	1.00	$-9 \times \pi\sqrt{3}/2$	1.02	$-6.68 \times \pi\sqrt{3}/2$

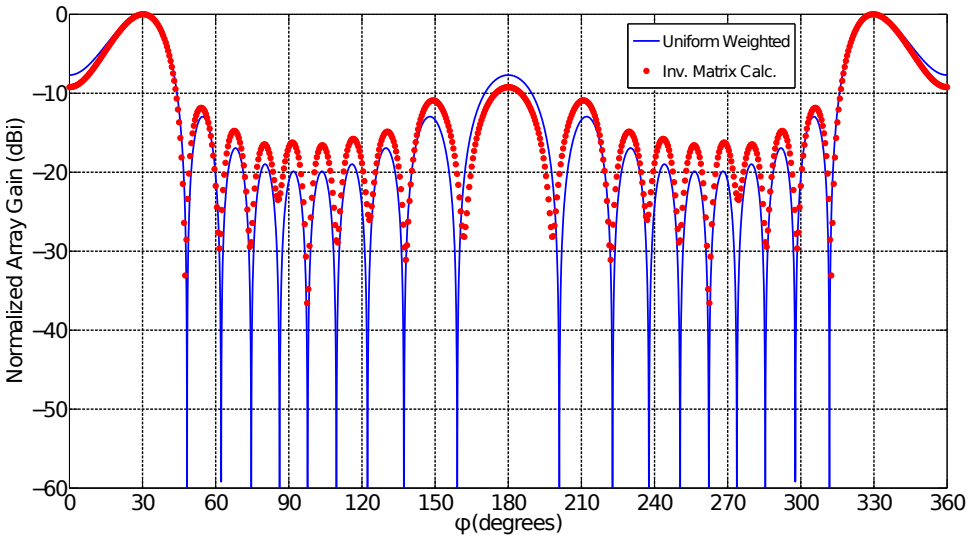


Figure 3.6: Array Factor Patterns for a ULA steered to 30° showing a reference pattern created with uniform weights and pattern with detuned complex weights calculated using (3.13) and variable $G_j \neq 0,\ j \in [1, N-1]$.

($f_o = 2.4\ GHz$) is shown in Figure 3.8.

The amplitude and phase weights needed to steer a six-element circular array are calculated using uniform weights and the canonical phase steering equation (for

Table 3.6: Comparison of complex array weights calculated using (3.13) with Chebyshev amplitude weights and canonical phase steering for N = 10 element ULA with mainbeam steered to 30°, $G_I = 9.54$, and $G_j \neq 0,\ j \in [1, N-1]$.

SOI / Jammer DOAs (°)	G_j	G_j (dB_{MB})	Chebyshev Ampl. Ref.		Inverse Matrix Calc.	
			Ampl.	Phase	Ampl.	Phase
30.00 (SOI)	–	–	0.78	0.00	1.23	$-0.01 \times \pi\sqrt{3}/2$
50.88	0.52	-25.23	0.72	$-1 \times \pi\sqrt{3}/2$	0.72	$-1.01 \times \pi\sqrt{3}/2$
61.44	0.47	-26.23	0.94	$-2 \times \pi\sqrt{3}/2$	0.86	$-2.03 \times \pi\sqrt{3}/2$
73.56	0.37	-28.21	1.12	$-3 \times \pi\sqrt{3}/2$	1.07	$-3.00 \times \pi\sqrt{3}/2$
85.62	0.69	-23.76	1.21	$-4 \times \pi\sqrt{3}/2$	1.05	$-4.02 \times \pi\sqrt{3}/2$
97.69	0.13	-37.18	1.21	$-5 \times \pi\sqrt{3}/2$	1.15	$-5.00 \times \pi\sqrt{3}/2$
110.10	0.37	-28.22	1.12	$-6 \times \pi\sqrt{3}/2$	1.00	$-6.99 \times \pi\sqrt{3}/2$
123.40	0.59	-24.37	0.94	$-7 \times \pi\sqrt{3}/2$	0.95	$-7.98 \times \pi\sqrt{3}/2$
138.20	0.41	-27.37	0.72	$-8 \times \pi\sqrt{3}/2$	0.70	$-9.00 \times \pi\sqrt{3}/2$
154.10	0.95	-20.07	0.78	$-9 \times \pi\sqrt{3}/2$	0.81	$-6.68 \times \pi\sqrt{3}/2$

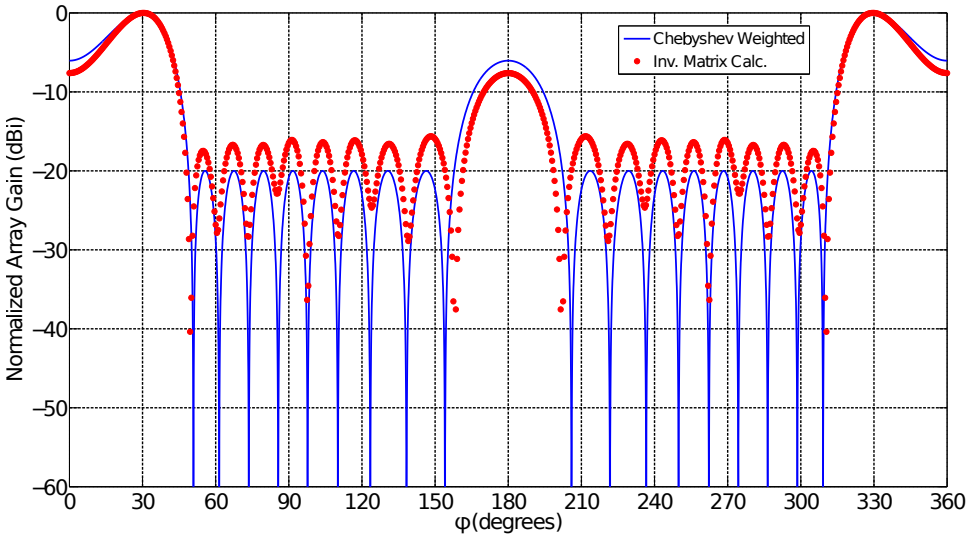


Figure 3.7: Array Factor Patterns for a ULA steered to 30° showing a reference pattern created with Chebyshev weights for SLL = -20 dB and pattern with detuned complex weights calculated using (3.13) and variable $G_j \neq 0,\ j \in [1, N]$.

phasing a circular array) as described by Haupt [33]. Our calculations are compared with equation (3.13) for the six-element circular array steered to 0°, 60°, and 30°. The matrix inversion method agrees well with the reference calculations for the 0° and

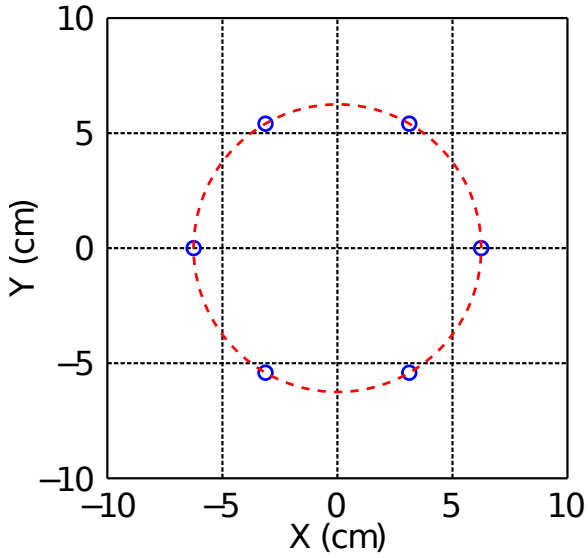


Figure 3.8: Layout of N = 6 element circular array with equally spaced elements on a radius $r = \lambda/2$.

60° SOI cases as can be seen in Tables 3.7 – 3.8 and Figures 3.9 – 3.10.

Table 3.7: Comparison of complex array weights calculated using (3.13) with uniform amplitude weights and canonical phase steering for $N = 6$ element circular array with mainbeam steered to 0°, $G_I = N$, and $G_j = 0$, $j \in [1, N - 1]$.

SOI / Jammer Directions (°)	Uniform Amplitude Reference		Inverse Matrix Calculation	
	Amplitude	Phase (Radians)	Amplitude	Phase (Radians)
0 (SOI)	1.00	$-\pi$	1.00	$+1.00\pi$
44.84	1.00	-0.5π	1.00	$+1.50\pi$
105.00	1.00	$+\pi$	1.00	$+2.50\pi$
157.10	1.00	$+0.5\pi$	1.00	$+3.00\pi$
202.90	1.00	$+0.5\pi$	1.00	$+2.50\pi$
255.00	1.00	-0.5π	1.00	$+1.50\pi$

When the SOI's DOA is changed to 30° with new reference phase weights, one of the six nulls seen in the previous patterns vanishes as shown in Figure 3.11. This makes the problem intractable, as one of the J equations of (3.7) no longer exists, and $N - 1$ equations are present for solving N unknowns. It should be noted that

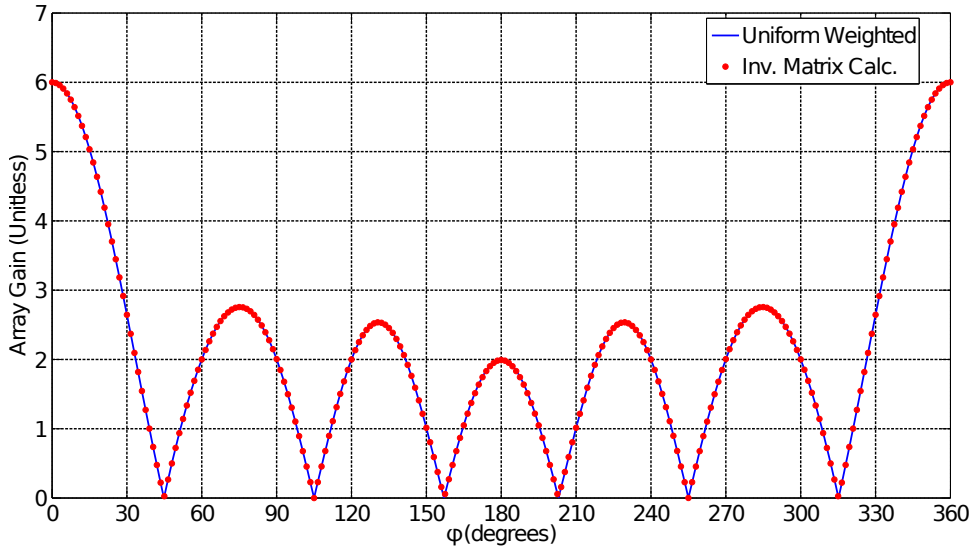


Figure 3.9: Array Factor Patterns for a circular array steered to 0° showing a reference pattern created with Uniform weights and pattern with complex weights calculated using (3.13) and $\mathrm{G_j} = 0,\ \mathrm{j} \in [1, \mathrm{N} - 1]$.

Table 3.8: Comparison of complex array weights calculated using (3.13) with uniform amplitude weights and canonical phase steering for $N = 6$ element circular array with mainbeam steered to 60°, $\mathrm{G_I} = \mathrm{N}$, and $\mathrm{G_j} = 0,\ \mathrm{j} \in [1, \mathrm{N} - 1]$.

SOI / Jammer Directions (°)	Uniform Amplitude Reference		Inverse Matrix Calculation	
	Amplitude	Phase (Radians)	Amplitude	Phase (Radians)
60 (SOI)	1.00	-0.5π	1.00	-0.50π
15.16	1.00	$-\pi$	1.00	-1.00π
104.80	1.00	-0.5π	1.00	-0.50π
165.00	1.00	$+0.5\pi$	1.00	$+0.50\pi$
217.10	1.00	$+\pi$	1.00	$+1.00\pi$
262.90	1.00	$+0.5\pi$	1.00	$+0.50\pi$

the array factor for a circular array can be expressed as an infinite sum of n Bessel functions of the First Kind of order $n \times N$, $J_{\mathrm{nN}}(\beta\, r)$, $\mathrm{n} \in (-\infty, +\infty)$, per [3, 33] (see equation (2.16)). Although the 0^{th} order Bessel function is the principle component, the other Bessel functions in this sum cannot be ignored since the small array size means that $J_{\mathrm{nN}}(\beta\, r) \neq 0$ for $n \neq 0$. Since Bessel functions of order $|\nu| > 0$ are

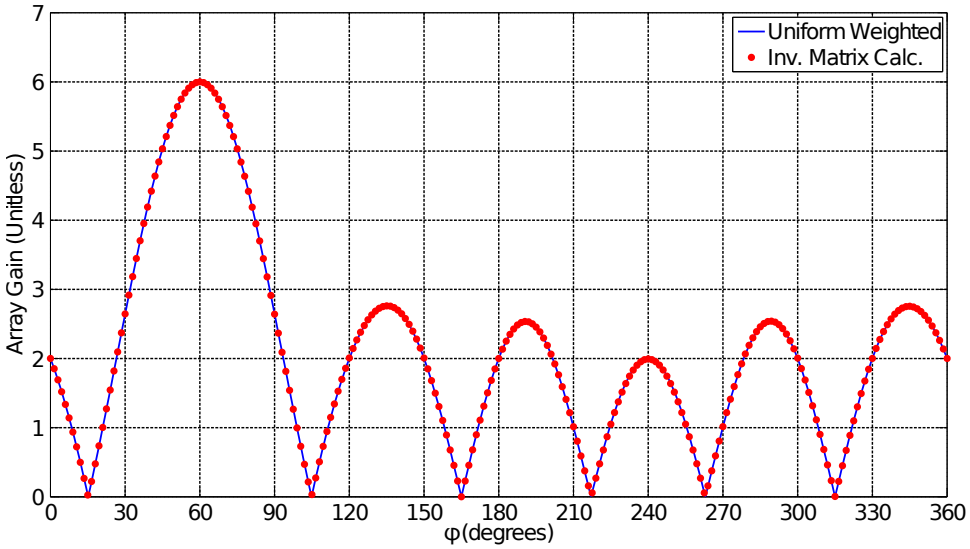


Figure 3.10: Array Factor Patterns for a circular array steered to 60° showing a reference pattern created with Uniform weights and pattern with complex weights calculated using (3.13) and $G_j = 0,\ j \in [1, N]$.

out of phase with $J_0(\beta r)$ and each other [68], it is possible for their zeros to change location as the array's mainbeam is rotated.

In an attempt to alleviate this problem, a new direction is chosen that can be inserted into the system of equations via equation (3.8). The array factor had local minima at 120° and 300° with equal values, and 300° with $G_{m=1} = G_{j=4} = 0$ is chosen. It is clear from Table 3.9 and Figure 3.11 that this does not completely solve the problem. Although the inverse matrix calculation gave a pattern whose mainbeam and $\{G_j\}$ nulls matched the reference pattern, the sidelobes between 75.19° to 174.20° and 245.80° to 344.80° do not match. The calculated weights require an active array because their amplitudes are greater than unity.

The setting $G_{m=1} = 2$ used to match the value of the sidelobe at 300° is also

investigated. The resulting calculated weights listed in Table 3.10 have amplitudes greater than unity, and they do not match the reference weights. The two patterns shown in Figure 3.12 do not completely match although the mainbeam and null locations are correct. This example shows that this method described by equation (3.13) produces limited results when the array layout is circular.

Table 3.9: Comparison of complex array weights calculated using (3.13) with uniform amplitude weights and canonical phase steering for $N = 6$ element circular array with mainbeam steered to 30°, $\mathrm{G_I} = \mathrm{N}$, and $\mathrm{G_j} = 0,\ \mathrm{j} \in [1, \mathrm{N} - 1]$.

SOI / Jammer Directions (°)	Uniform Amplitude Reference		Inverse Matrix Calculation	
	Amplitude	Phase (Radians)	Amplitude	Phase (Radians)
30 (SOI)	1.00	$-\frac{\sqrt{3}}{2}\pi$	1.27	-0.75π
75.19	1.00	$-\frac{\sqrt{3}}{2}\pi$	0.95	-1.03π
174.20	1.00	0	1.09	-1.87π
245.80	1.00	$+\frac{\sqrt{3}}{2}\pi$	1.27	-1.25π
300.0 (non-null)	1.00	$+\frac{\sqrt{3}}{2}\pi$	0.95	-0.97π
344.80	1.00	0	1.09	-0.13π

A concentric circular array (CCA) is a planar array that is arranged in concentric circles about the array's origin with a single element placed at the origin [33]. A discussion of the theory behind calculating complex phase weights for CCAs is relevant since the proof-of-concept beamforming array to be discussed later in this thesis is a type of CCA. The simplest form of a CCA is a one-ring circular array with a single element located at the origin as shown in Figure 3.13.

The method described by (3.13) with the layout shown in Figure 3.13 is compared

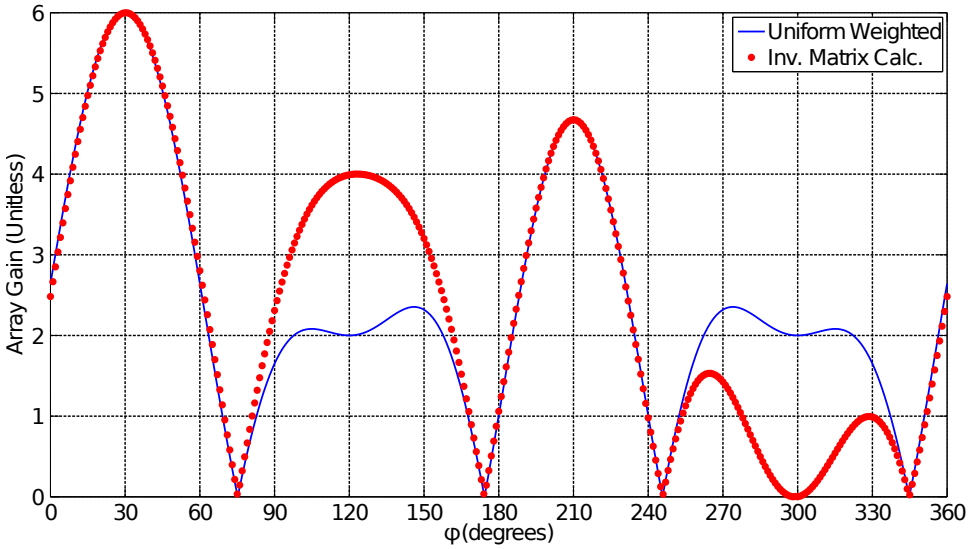


Figure 3.11: Array Factor Patterns for a circular array steered to 30° showing a reference pattern created with Uniform weights and pattern with complex weights calculated using (3.13) and $G_j = 0,\ j \in [1, N]$.

Table 3.10: Comparison of complex array weights calculated using (3.13) with uniform amplitude weights and canonical phase steering for $N = 6$ element circular array with mainbeam steered to 30°, $G_I = N$, $G_j = 0,\ j \in [1, N-3] \cup (N-1)$, and $G_{j=N-2} = 2$.

SOI / Jammer Directions (°)	Uniform Amplitude Reference		Inverse Matrix Calculation	
	Amplitude	Phase (Radians)	Amplitude	Phase (Radians)
30 (SOI)	1.00	$-\frac{\sqrt{3}}{2}\pi$	1.64	-0.67π
75.19	1.00	$-\frac{\sqrt{3}}{2}\pi$	1.14	-1.17π
174.20	1.00	0	1.34	-1.77π
245.80	1.00	$+\frac{\sqrt{3}}{2}\pi$	1.64	-1.33π
300.0 (non-null)	1.00	$+\frac{\sqrt{3}}{2}\pi$	1.14	-0.83π
344.80	1.00	0	1.34	-0.23π

with a reference array pattern calculated using uniform weights and canonical steering using incremental phase shifts that are calculated per [33]. The SOI is steered to 0°,

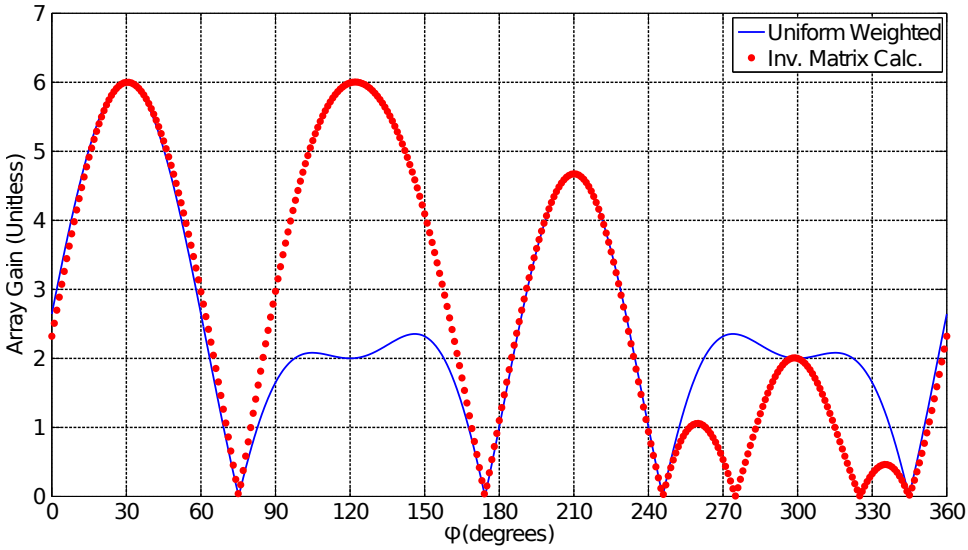


Figure 3.12: Array Factor Patterns for a circular array steered to 30° showing a reference pattern created with Uniform weights and pattern with complex weights calculated using (3.13) and $G_j = 0$, $j \in [1, N]$.

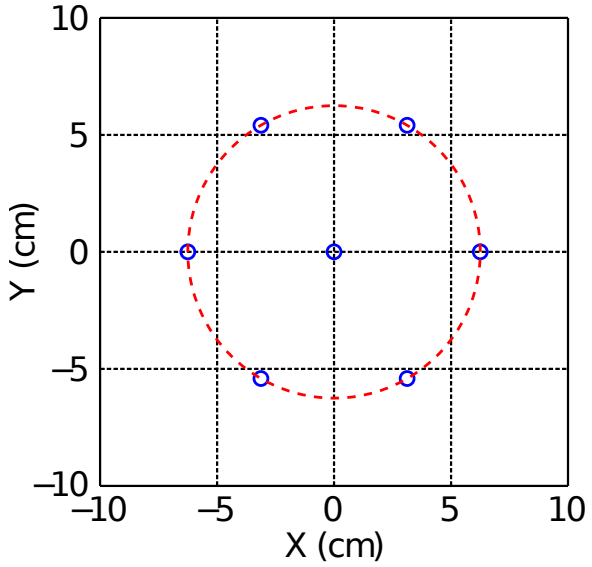


Figure 3.13: Layout of $N = 7$ element concentric circular array with six equally spaced elements on a radius $r = \lambda/2$ and one element in the center.

and the jammer directions are chosen to be the six nulls that appeared in the reference pattern. The SOI/jammer directions as well as the reference and calculated weights

are shown in Table 3.11. The resulting Azimuth patterns are shown in Figure 3.14.

Table 3.11: Comparison of complex array weights calculated using (3.13) with uniform amplitude weights and canonical phase steering for a $N = 7$ element single-ring concentric circular array with an element located at the origin and its mainbeam steered to 0°, $G_I = N$, and $G_j = 0,\ j \in [1, N]$.

SOI / Jammer Directions (°)	Uniform Amplitude Reference		Inverse Matrix Calculation	
	Amplitude	Phase (Radians)	Amplitude	Phase (Radians)
0 (SOI)	1.00	0	5.99	$+1.00\pi$
51.41	1.00	$-\pi$	4.68	$+0.98\pi$
98.22	1.00	-0.5π	3.94	$+1.07\pi$
164.50	1.00	$+0.5\pi$	3.94	$+0.93\pi$
195.40	1.00	$+\pi$	4.68	$+1.02\pi$
261.80	1.00	$+0.5\pi$	3.94	$+0.93\pi$
308.60	1.00	-0.5π	3.94	$+1.07\pi$

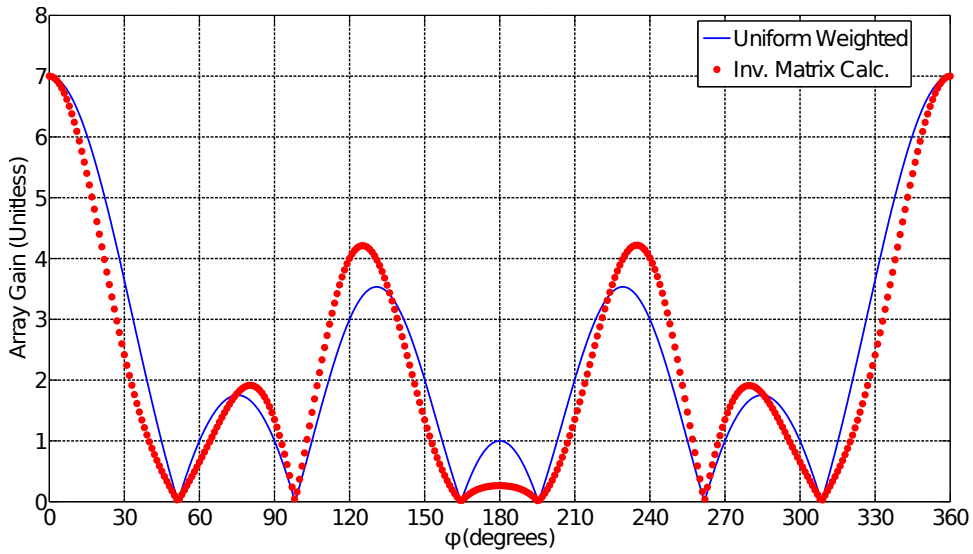


Figure 3.14: Array Factor Patterns for a $N = 7$ element concentric circular array steered to 0° showing a reference pattern created with Uniform weights and pattern with complex weights calculated using (3.13) and $G_j = 0,\ j \in [1, N]$.

It is clear that this method found a different solution for steering a $N = 7$

element CCA towards 0° compared to the canonical reference, as this method created a different set of complex weights as listed in Table 3.11. The array operates inefficiently in the sense that the SOI gain is seven (unitless) whereas the amplitude weights varied from approximately four to approximately six. The azimuth patterns have the same SOI gain and null directions, but their sidelobe gains and shapes clearly differ (see Figure 3.14).

Multiple-ring CCAs with uniform amplitude weights and canonical phase steering have a total number of nulls that is less than $N-1$, so the method discussed above cannot be readily evaluated in that case. Although it is possible to reduce the size of the matrix $\mathbf{B}$ by utilizing symmetry in the complex array weights, such an approach is too restrictive and will not be discussed here.

3.3 Stationary Signal Beamforming and Anti-Jamming in Fading Wireless Channels with N-Element Arrays

A model for optimization of N-element beamforming arrays through fading wireless channels is developed as discussed in [34]. A high level diagram of a wireless system with an N-element beamforming array is shown in Figure 3.15. The array is connected to a receiver whose output is fed to a computer. The computer acts like a controller and optimizes the array weights such that only SOI and noise appears at the output. The diagram shows two jammers and omits multipath reflections for

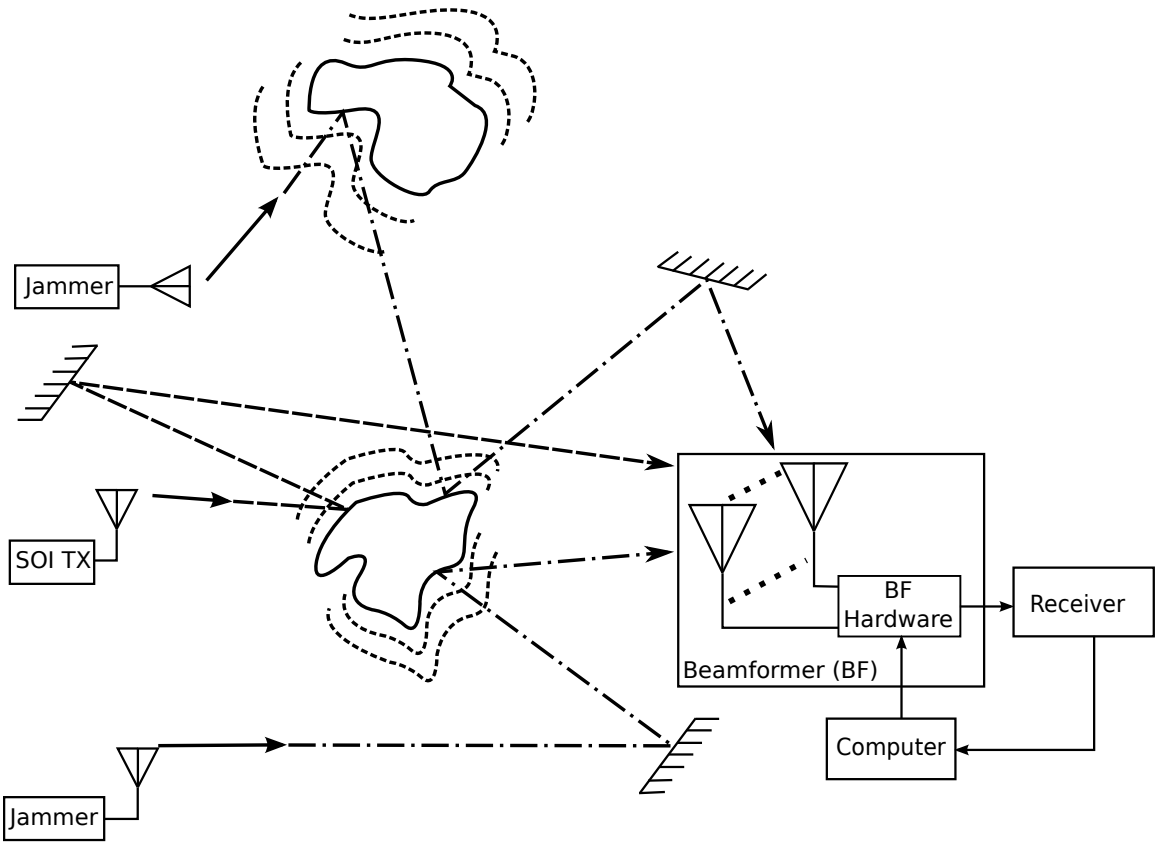


Figure 3.15: High level diagram of a wireless system with interference and non-LOS paths with stationary signals. Obstacles can be mobile or stationary to cause time-varying fades in received signals. Multipath reflections not shown for clarity.

clarity. Note that both stationary and mobile obstructions are present.

The mobile objects shown in Figure 3.15 create fades in the signals that impinge on the array. These objects are assumed to move slowly (e.g., no faster than 2 m/sec). An analytic block diagram for this model is shown in Figure 3.16. The channel model is $\hat{\mathbf{H}} \in \mathcal{C}^{(1+J)\times(1+J+Q)}$ where J represents the number of interference signals, and Q represents the number of multipath reflections impinging on the array. The unweighted array elements (i.e., antenna output terminals) are represents by

$\boldsymbol{\Sigma} \in \mathcal{C}^{(1+J+Q)\times(N)}$. Because antenna elements behave like spacial filters, the array transforms $1+J+Q$ directional inputs into N voltages that are individually weighted and summed to create an output $\hat{y}(t) = \hat{\bar{a}}^{\mathrm{T}} \cdot \hat{\bar{x}}(t) \in \mathcal{C}^{1\times 1}$. For anti-jamming beamforming to be successful, $1 + J + Q \leq N$ because it is assumed that multipath reflections are non-coherent with respect to the original signals, and the array cannot isolate more than N independent signals.

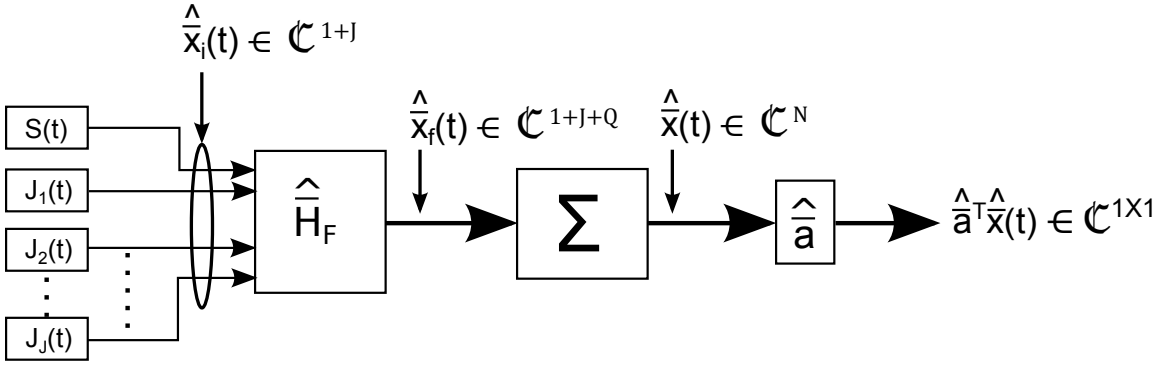


Figure 3.16: Analytic block diagram of wireless communication link from SOI and interference inputs to a fading wireless channel to output of beamformer.

In [34], it is noted that this channel is a fast-fading Rayleigh channel in part because it is NLOS. The mobile objects move fast enough such that the signals impinging on the array are at significantly different fade levels by the time the computer sets and begins evaluation of the next array settings. Assuming that the receiver can distinguish between signals and interference (including multipath reflections), the output SINR at time t is expressed as

$$\mathrm{SINR_{Out}(t)} = \frac{\mathrm{G_A}(\theta_0, \phi_\mathrm{o})\,\mathrm{P_{S,Avg}(t)}}{\sum_{\mathrm{j}=1}^{\mathrm{J}} \mathrm{G_A}(\theta_\mathrm{j}, \phi_\mathrm{j})\,\mathrm{P_{j,Avg}(t)} + \mathrm{N_o}} \tag{3.14}$$

where N_o is the Johnson (thermal) noise defined as

$$N_o \;=\; k_B T_A B \tag{3.15}$$

k_B is Boltzmann's constant, T_A is the temperature (°K) at the array's summed output, and B is the system bandwidth. The antenna array gains (in SOI and interference directions) are noted as $G_A(\theta_0, \phi_o)$ and $G_A(\theta_j, \phi_j), j \in [1, J]$ in equation (3.14). Output signal powers are averaged in (3.14) to minimize effects of noise in simulations and in measurements. $P_{\mathrm{S,Avg}}(t)$ represents the average SOI power at the array's output, and $P_{\mathrm{j,Avg}}(t), j \in [1, J]$ represents averaged output interference powers.

For optimization with a simple genetic algorithm (SGA), the array's output SINR is used as the SGA's fitness in Figure 1.5. It should be noted that the SGA operates on binary strings, and many wireless protocols use digital communications [69], so time t in (3.14) needs to be discretized such that $t = t_{\mathrm{n,p}} = (2P \cdot n + p)\, T_{\mathrm{samp}}$, and (3.14) becomes

$$\mathrm{SINR_{Out}}\left[\mathrm{t_{n,p}}\right] \;=\; \frac{\mathrm{G_A}\left(\theta_0, \phi_\mathrm{o}\right) \mathrm{P_{S,Avg}}\left[\mathrm{t_{n,p}}\right]}{\sum_{\mathrm{j}=1}^{\mathrm{J}} \mathrm{G_A}\left(\theta_\mathrm{j}, \phi_\mathrm{j}\right) \mathrm{P_{j,Avg}}\left[\mathrm{t_{n,p}}\right] \;+\; \mathrm{N_o}} \tag{3.16}$$

where $n \in [0, N_{\max}]$ represents the generation number, $p \in [1, 2P]$ represents the current SGA string, and P is the number of mate pairs in the SGA population. (Population size is $2P$.) T_{samp} represents the amount of time it took for the SGA to evaluate one string. This number includes time needed to setup the complex array weights and to calculate average signal powers.

The average signal powers in equations (3.14) and (3.16) include losses due to

random time-varying fading. Without loss of generality, the loss factors can be factored out of the averaged received powers such that (3.14) and (3.16) become

$$\mathrm{SINR_{Out}\,(t)} \;=\; \frac{\mathrm{G_A\,(\theta_0,\phi_o)\,L_{Fade,S}\,(t)\,P^{NF}_{S,Avg}\,(t)}}{\sum_{j=1}^{J}\mathrm{G_A\,(\theta_j,\phi_j)\,L_{Fade,\,j}\,(t)\,P^{NF}_{j,Avg}\,(t)} \;+\; \mathrm{N_o}} \tag{3.17}$$

$$\mathrm{SINR_{Out}\,[t_{n,p}]} \;=\; \frac{\mathrm{G_A\,(\theta_0,\phi_o)\,L_{Fade,S}\,[t_{n,p}]\,P^{NF}_{S,Avg}\,[t_{n,p}]}}{\sum_{j=1}^{J}\mathrm{G_A\,(\theta_j,\phi_j)\,L_{Fade,\,j}\,[t_{n,p}]\,P^{NF}_{j,Avg}\,[t_{n,p}]} \;+\; \mathrm{N_o}} \tag{3.18}$$

The loss factors $L_{\mathrm{Fade,S}}$ and $\{L_{\mathrm{Fade,\,j}}\}$, $j \in [1, J]$ are averaged over the same time windows as the received powers. These loss factors are non-negative with values greater than one when constructive interference due to multipath occurs. This channel is considered fast-fading, as it is assumed that $|L_{\mathrm{Fade,\,\{S\,or\,j\}}}\,[t_{\mathrm{n,p}}] \;-\; L_{\mathrm{Fade,\,\{S\,or\,j\}}}\,[t_{\mathrm{n,p-1}}]| >$ 3 dB, $\forall\,\mathrm{n} \in [1, \mathrm{N}]$, $\forall\,\mathrm{p} \in [1, 2\mathrm{P}]$, and $\forall\,\mathrm{j} \in [1, \mathrm{J}]$.

It is useful to understand how channel fading affected optimization algorithms in performing anti-jamming beamforming. Suppose that fading effects are removed from the calculation of SINR. A modified fade-free version of equation (3.18) can be written

$$\mathrm{SINR^{NF}_{Out}\,[t_{n,p}]} \;=\; \frac{\mathrm{G_A\,(\theta_0,\phi_o)\,P^{NF}_{S,Avg}\,[t_{n,p}]}}{\sum_{j=1}^{J}\mathrm{G_A\,(\theta_j,\phi_j)\,P^{NF}_{j,Avg}\,[t_{n,p}]} \;+\; \mathrm{N_o}} \tag{3.19}$$

It can easily be shown that

$$\mathrm{SINR_{Out}\,[t_{n,p}] > SINR^{NF}_{Out}\,[t_{n,p}]} \quad \text{if} \quad \sum_{j=1}^{J}\mathrm{L_{Fade,\,j}\,[t_{n,p}] > L_{Fade,S}\,[t_{n,p}]} \tag{3.20}$$

In other words, fading can artificially improve SINR when interfering signals are in deeper fades compared to the SOI.

3.4 Mobile Signal Beamforming and Anti-Jamming with N-Element Arrays

We now consider the case of mobile jammer where the wireless channel is not necessarily Rayleigh. The signal sources can change positions in a way such that some signals are not blocked, and LOS paths between the sources and the beamformer exist (see Figure 3.17). It is assumed that the beamformer in Figure 3.17 does not move. Thus, the wireless channel with mobile and stationary signals is better modeled with Rician fading. The analytic block diagram of this channel is the same as shown in Figure 3.16.

Weile *et al.* [28] shows that a GA with dominance and diploidy optimized an array when signals are mobile. Because the wireless channel and optimization algorithms used are stochastic in nature, the four events that can occur during a stochastic algorithm run are described in Table 3.12.

Table 3.12: Four events that can occur during a stochastic algorithm run.

Event	Event Description
A	Signal DOAs changed.
B	SINR fitness dropped.
C	The algorithm chose a worse solution.
D	The wireless channel faded.

Using these event definitions and Bayes Theorem [70], we can write the posterior

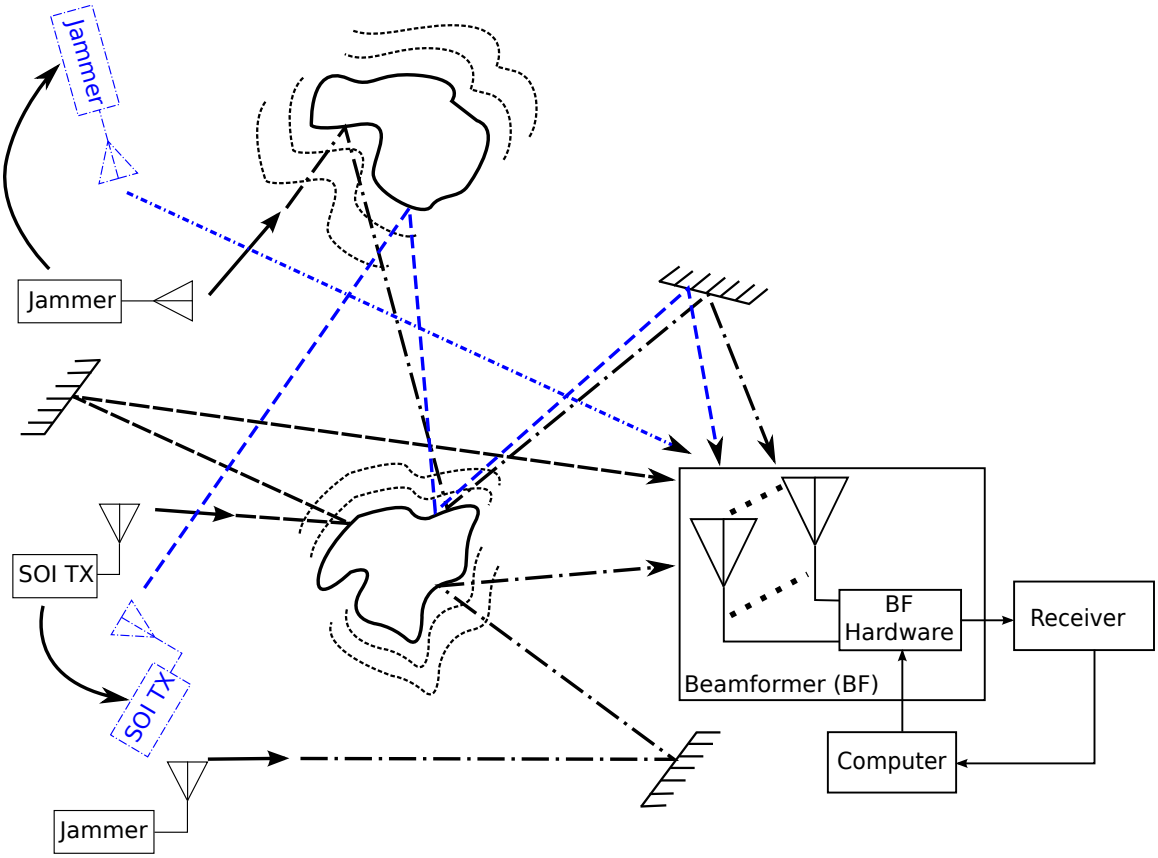


Figure 3.17: Upper level diagram of a wireless system with mobile and stationary signals. Obstacles can be mobile or stationary to cause time-varying fades in received signals. Multipath reflections not shown for clarity.

probability that the signal DOAs changed given that a SINR fitness drop is observed:

$$P(A|B) = \frac{P(B|A)\ P(A)}{P(B)} \tag{3.21}$$

Time indexes (i.e., evaluation / generation numbers) are omitted in equation (3.21) for clarity. A Venn diagram for this event system is shown in Figure 3.18. Event B is confined within events A, C, and D, as the only three events that can cause SINR drops are variable DOAs, channel fades, and algorithm mistakes[2]. From Figure 3.18,

[2]Hardware faults (Event E) can cause SINR fitness to drop. We consider hardware faults in the next chapter.

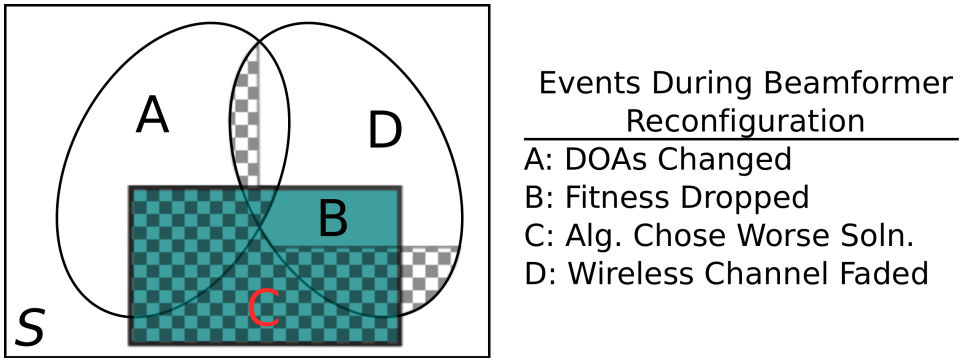


Figure 3.18: Venn Diagram showing the relationship between events A, B, C, and D in the parameter search space S. Event B subspace is represents by a solid color, and Event C subspace is represents by a checkerboard pattern.

it is observed that

$$B \;=\; B \cap C \;+\; B \cap D \;-\; B \cap C \cap D \tag{3.22}$$

The subtraction of $B \cap C \cap D$ in equation (3.22) is necessary because event C overlapped events A and D. Event C therefore overlapped subset $B \cap D$ and events contained in that subsets would have been doubled counted otherwise. Before proceeding, it should be explained why $B \neq C$ in Figure 3.18. By definition, a solution S_n is worse than its predecessor if its distance (in the parameter search space) from the global optimum $||S_n - G_o||$ is greater than its predecessor. In other words,

$$\Delta||S|| \;=\; ||S_n - G_o|| \;-\; ||S_{n-1} - G_o|| \begin{cases} S_n \text{ Worse} & \text{if } \Delta||S|| < 0 \\ S_n \text{ Same} & \text{if } \Delta||S|| = 0 \\ S_n \text{ Better} & \text{if } \Delta||S|| > 0 \end{cases} \tag{3.23}$$

A fitness function converts the current solution S_n into a positive number that can be compared and ranked against other solutions. In general, a fitness function can

be defined as $\mathcal{F}\left(S_n; \bar{\theta}, \bar{\phi}; \mathcal{H}\left(t\right)\right)$ whose output $\mathcal{F}$ is a non-negative number. $\bar{\theta}, \bar{\phi}$ are $(1 + J \times 1)$ vectors that represents the SOI and interference DOAs, and $\mathcal{H}\left(t\right)$ represents the time-varying wireless channel response at time t. Assuming that the DOAs are constant (as is the case for event A in Figure 3.18), the time-varying nature of the channel potentially causes fades that can artificially decrease the current solution's fitness even if $\Delta||S|| \geq 0$. Channel fading can force the algorithm to generate worse solutions with better fitness. For example, if a fade sufficiently attenuated the k^{th} jammer at evaluation n, the algorithm would likely choose a solution S_n without a null directed at the k^{th} jammer. Although this solution likely has a higher fitness that solution S_{n-1}, $\Delta||S|| < 0$ indicating that S_n is a worse solution. Thus, events B and C cannot be equal.

Observe that if the DOAs are time-varying, the global optimum (as well as local optima) become time-varying. For solutions that exist in the subspace $C \cap D$, $G_{o,n} \neq G_{o,n-1}$, current solutions with fitness less than their predecessors are also worse solutions. The subspace defined by $A \cap D$ is also problematic. Not only can channel fades exist as signals changed their DOAs, a moving signal can cause a fade in the channel (i.e., a jammer moves into a fade). Therefore, solutions existing in the subspace of $A \cap D$ outside B can be better or worse solutions. Solutions existing in the subspace defined by $A \cap D \cap B$ are assumed to be worse solutions because they represents solutions that are farther away from the updated global optimum. An example is when the SOI moves into a fade, the resulting SINR fitness would

decrease by definition and represents a worse solution.

The subsets in equation (3.22) are converted into equivalent probabilities with the application of Bayes Rule in equation (3.24) to create equations (3.25a) – (3.25c)

$$P(B) = P(BC) + P(BD) - P(DBC) \tag{3.24}$$

$$P(B) = P(B|C)\,P(C) + P(B|D)\,P(D) - P(D\,B|C)\,P(C) \tag{3.25a}$$
$$P(B) = P(B|C)\,P(C) + P(B|D)\,P(D) - P(D|B\,C)\,P(B|C)\,P(C) \tag{3.25b}$$
$$P(B) = P(B|C)\,P(C)\,[1 - P(D|B\,C)] + P(B|D)\,P(D) \tag{3.25c}$$

An upper bound on equation (3.25c) can be created by dropping the term $P(D|B\,C)$:

$$P(B) \leq P(B|C)\,P(C) + P(B|D)\,P(D) \tag{3.26}$$

with equality obtained when $P(D|B\,C) = 0$. The difficulty arose in calculating the prior probabilities, as they are generally problem dependent. The priors relied on the state of the wireless channel, the beamforming array layout used, and the optimization algorithm used. For example, $P(B|D) \leq 1$ because an interfering signal can move from a sidelobe in the array's radiation pattern into a null, and this would cause SINR to increase. We observe situations in SGA simulations with stationary signals (see Section 3.3) where fading caused artificial SINR increases, and we expect that this would also occur when the signals are mobile.

This thesis proposes a Bayesian assisted temperature schedule (see Figure 3.19) to aid simulated annealing and HCA in reconfiguring a beamforming array to handle mobile signals. The idea behind a Bayesian assisted temperature schedule is to add an additional temperature offset Eval_{Off} in the next algorithm iteration such that the

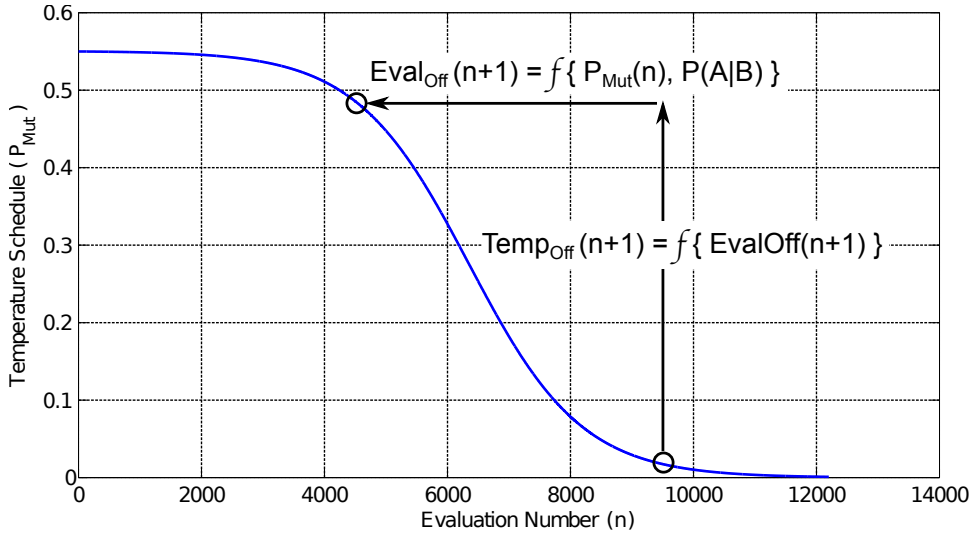


Figure 3.19: Example Bayesian assisted temperature schedule with $9.5 \times 10^{-4} \leq P_{\text{Mut}} \leq 0.55$ shown over 12,200 evaluations.

sigmoidal temperature cooling schedule is redefined as

$$T_{\text{Mod}}(n) \;=\; P_{\text{Mut, Mod}}(n) \;=\; 1 - \frac{1}{1 + e^{-\tau(n - \text{Eval}_{\text{Off}}(\text{n})) + T_{\text{off}}}} \tag{3.27}$$

It follows that $\text{Eval}_{\text{Off}}(\text{n}+1) = f\{\text{P}_{\text{mut}}(\text{n}), \text{P}(\text{A}|\text{B})\}$. The size of the Bayesian offset depends on the likelihood that the signals are mobile and on the algorithm's stage in its search. An offset is unnecessary if the algorithm encounters stationary signals such that $P(A|B) \to 0$. Its conditioning on $P_{\text{mut}}(n)$ prevents generation of unnecessary offsets that force the algorithm into an endless exploration loop, so an offset is also unnecessary if $T_{\text{Mod}}(n) \geq (P_{\text{Mut,i}} + P_{\text{Mut,f}})/2$. Assuming that the offset is originally zero, a simple offset implementation would set $\text{Eval}_{\text{Off}}(\text{n}+1)$ such that $T_{\text{Mod}}(n+1) \;=\; 0.9P_{\text{Mut,i}}$ if $P(A|B)(n) \geq 0.5$ and $T_{\text{Mod}}(n) < (P_{\text{Mut,i}} + P_{\text{Mut,f}})/2$.

3.5 Equivalence of Stochastic Optimization of Anti-Jamming Beamforming Arrays Using Electromagnetic Models of Varying Complexity

In performing stochastic optimization of anti-jamming beamforming arrays, there are multiple EM simulation tools that can be used in calculating solution fitness functions. The fidelity of these simulations (as compared to *in-situ* measurements) depends strongly on the EM model that is used. For example, the array factor calculation (as shown by the setup shown in Figure 3.20) treats antenna elements as infinitesimal dipoles, and it assumes that the measurement point is in the far-field with respect to the antenna array. The calculation is a superposition (i.e., weighted sum) from the the individual elements, and it ignores coupling between antenna elements and reflections off array hardware such as metallic step-attenuators and phase shifters.

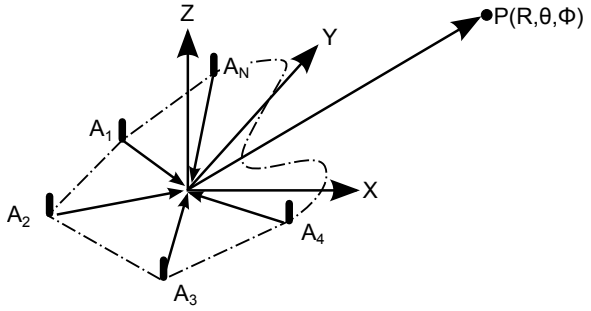


Figure 3.20: Model of an N-element antenna array consisting of infinitesimal dipole elements with arbitrary layout used to perform the array factor calculation.

To increase the model complexity and fidelity, we can treat the individual elements as dipoles with physical lengths as shown in Figure 3.21. This is the standard Method of Moments (MOM) approach (using a program such as WIPL-D) used to model antenna arrays consisting of dipole elements. Clearly, it incorporates mutual

coupling between antenna elements since reflections between elements affect the mutual impedance between the elements. This model is an improvement over the array factor method, but it still neglects reflections off nearby components.

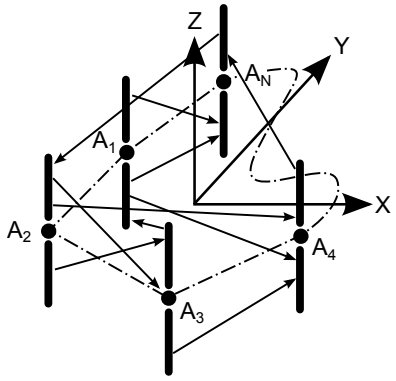


Figure 3.21: Model of an N-element antenna array that consisted of wire dipole elements with arbitrary layout used in a MOM radiation pattern calculation.

To further increase fidelity, we can add metallic components to the array model as shown in Figure 3.22. The model is closer to reality because the model incorporates both mutual coupling and external reflections that affect *in-situ* radiation pattern measurements. This model can be built and simulated with either MOM or FEM. The drawback is that the third model has increased computational costs although it is possible to reduce these computational costs as will be discussed in Section 5.3

A method of incorporating mutual coupling into the optimal array weights is discussed in the literature by Bevelacqua [6], Zhang *et al.* [43], and Joler *et al.* [59]. This method is summarized in Figure 3.23.

The optimal weights are first found using a stochastic algorithm that calculated the antenna array's radiation patterns using array factor calculation. The inverse of the coupling matrix ($\mathbf{M}_C^{-1}$) is calculated using MOM, and the optimized weights with

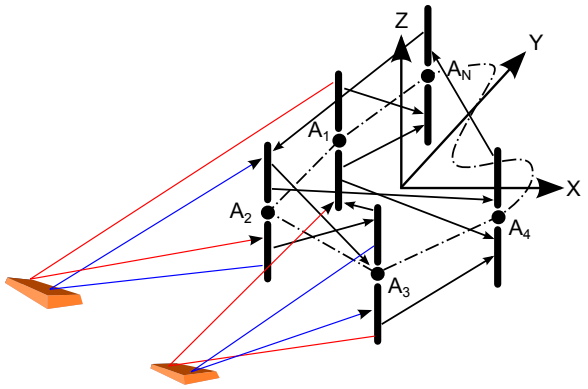


Figure 3.22: Model of an N-element antenna array that consisted of wire dipole elements plus metallic hardware components with arbitrary layout used in a MOM radiation pattern calculation.

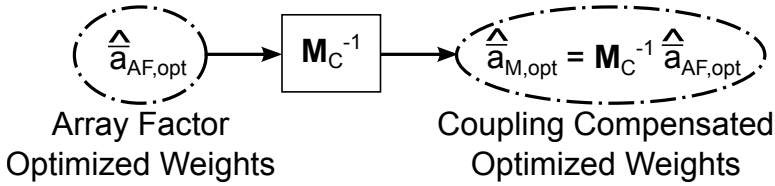


Figure 3.23: Method of calculating optimal array weights with compensation for mutual coupling between antenna elements.

compensation for mutual coupling are calculated as

$$\hat{\bar{a}}_{\mathrm{M,opt}} \;=\; \mathbf{M}_C^{-1} \cdot \hat{\bar{a}}_{\mathrm{AF,opt}} \tag{3.28}$$

The contribution here is the realization that the it is possible to incorporate the effects of reflections off hardware near the antennas into the optimal weights. This can be done either directly by calculating the array factor, or it can be done by finding the optimal weights with a stochastic optimization algorithm integrated with a MOM program as shown in Figure 3.24.

The matrix $\mathbf{M}_{\mathrm{CR}}^{-1}$ represents the inverse coupling and hardware reflections matrix, and it can be calculated by simulating the antenna array model of Figure 3.22 in

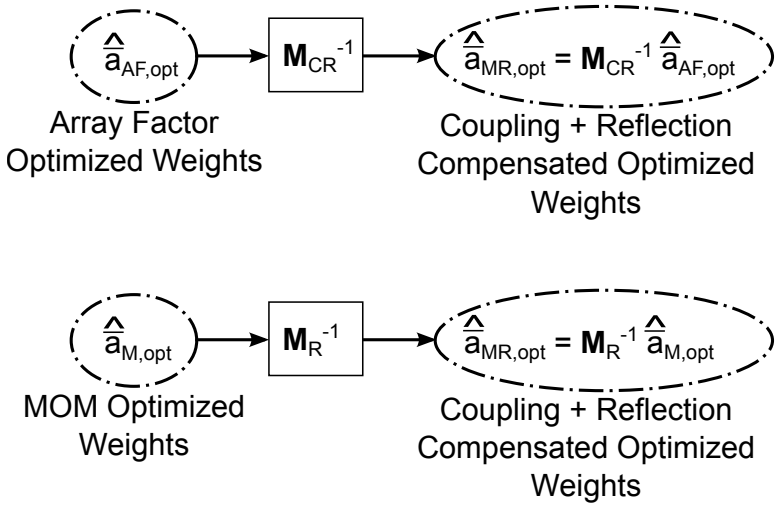


Figure 3.24: Method of calculating optimal array weights with compensation for mutual coupling between antenna elements and reflection off hardware near antenna elements.

a MOM or FEM program. By multiplying this inverse matrix by $\hat{\hat{a}}_{\mathrm{AF,opt}}$, the equivalent optimized array weights with compensation for mutual coupling and external reflections can be calculated. Alternatively, the optimal array weights with mutual coupling only considered (i.e., $\hat{\hat{a}}_{\mathrm{M,opt}}$) can be calculated by using a MOM or FEM program by running a stochastic algorithm with the model described in Figure 3.21. The inverse matrix $\mathbf{M}_{\mathrm{R}}^{-1}$ represents the effects of external reflections, and its multiplication with $\hat{\hat{a}}_{\mathrm{M,opt}}$ would also create an equivalent set of optimized weights with compensation for mutual coupling and external reflections.

From the discussion we can conclude that stochastic algorithms that use any of the three array models discussed above are theoretically equivalent (Figure 3.25). This conclusion stems from the discussions of coupling compensation [6, 43, 59] to determine equivalent optimal array weights that include compensation for mutual compensation between elements. We further note that this compensation can be

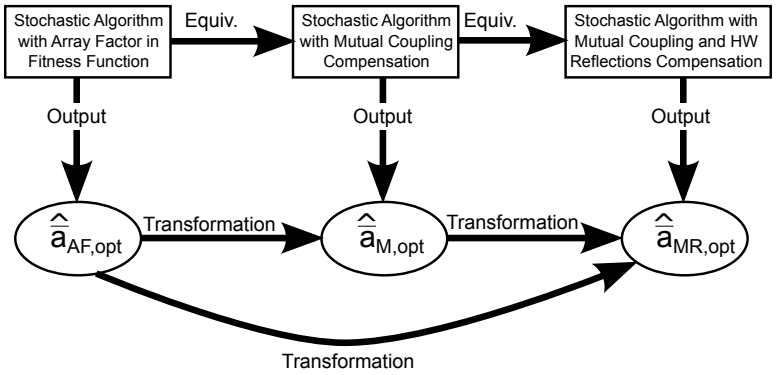


Figure 3.25: Diagram showing equivalence of stochastic algorithms using array factor calculations, mutual coupling only compensation, and mutual coupling plus external reflections compensation in calculating optimum array weights.

extended to include external reflections provided that array hardware is included in the overall array model. This is not to say that the three algorithms produce the same optimized array weights, as the three optimal array weight vectors shown in Figure 3.25 are related to each other by the appropriate transformation described by Figures 3.23 and 3.24.

3.6 Summary

We discuss models for phased antenna array anti-jamming in this chapter. The problem is formulated for a N-element antenna array with arbitrary layout assuming infinitesimal dipole elements. Although this formulation ignores secondary mutual coupling effects, it is applicable for a wide range of antenna arrays, and it serves as a basis for the remainder of this chapter.

A method of deriving the complex array weight solutions for a given set of sig-

nal directions is derived following the problem formulation. This method involves calculating the inverse of the array manifold matrix, so it requires that the matrix is invertable. This means that an N-element array can handle at most N significantly different signal DOAs. This method serves as a baseline for comparing array weight solutions found by stochastic optimization algorithms. It is shown that the method of calculating the complex array weights is analytically sound by comparing the array weights calculated using equation (3.13) with linear arrays with Uniform and Dolph-Chebyshev weights.

It is proven indirectly that the anti-jamming beamforming problem is multimodal by showing that complex array weight solutions existed for $N = 10$ element array patterns (with both Uniform and Chebyshev reference weights) having randomly chosen null depths $-40\,dB \leq G_{\mathrm{j}} \leq -20\,dB, j \in [1, N-1]$. The multimodality of the anti-jamming beamforming problem arises in part from optimization algorithms' shortcoming of getting stuck on non-optimal solutions (i.e., local optima) having non-zero null-depths. This method clearly produces different weights and radiation patterns as shown in Tables 3.5 – 3.6 and Figures 3.6 – 3.7. This method is compared on both UCAs and CCAs with Uniform weighting, and the method produces different solutions having the same SOI beam gains and null depths but differing SLLs as shown in Table 3.11 and Figure 3.14.

Chapter 4

Models for Phased Antenna Array Hardware Fault Recovery

This chapter discusses new models for hardware fault recovery in phased antenna arrays. These models build upon the anti-jamming phased array models discussed in Chapter 3. The goal of this chapter is to demonstrate a means of detecting hardware faults in an antenna array as well as localizing and recovering from those faults. The goal of hardware fault recovery is to re-optimize the antenna array to perform anti-jamming and SOI beamforming despite the presence of hardware faults. Fault localization serves to provide reliability data on the array's components.

In our model, we assume that faults are limited to antenna array hardware. Software is assumed to operate without faults, and computing hardware is also assumed to be fault-free. Although it is possible for software and computing hardware faults to exist in real systems, discussion of these fault types are beyond the scope of this thesis.

This chapter continues as follows. Section 4.1 discusses the problem formulation and setup for implementing hardware fault recovery. Section 4.2 develops new models for hardware fault detection, Section 4.3 describes models for hardware fault recovery, and Section 4.4 details the models for fault localization. Section 4.5 presents theoret-

ical limits on how much of the SOI array gain can be recovered after hardware faults occurred.

4.1 Hardware Fault Recovery Problem Formulation

The antenna array setup for hardware fault recovery is similar to the array described in Section 3.1, with the exception that at lease one element is faulted as shown in Figure 4.1. The complex weights consist of attenuators and phase shifters. The fault at element k can be a damaged antenna, attenuator, phase shifter, or a combination of the three components. Possible faults include open/short circuits, detuned antennas, stuck-at-previous settings complex weights, and stuck-at-random settings complex weights. Only one faulty element is shown in Figure 4.1 for simplicity.

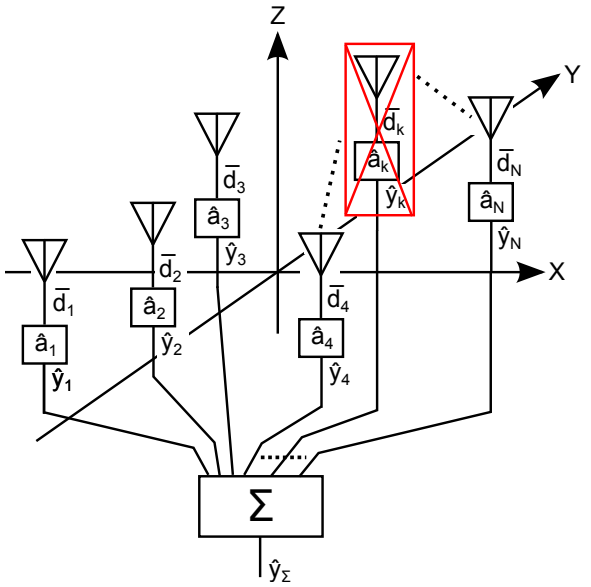


Figure 4.1: Diagram of a N-Element antenna array with arbitrary layout showing a faulty element k.

The goal of hardware fault detection and optimization is to ensure that an an-

tenna array continues to focus energy on the SOI while simultaneously thwarting interference despite the presence of hardware faults. To obtain this goal, fault detection and recovery are integrated into the optimization algorithm as shown in Figure 4.2. The model includes fault localization to maintain a database of suspected faults for later repair (Figure 1.4), and the model assumes that fault localization is not necessary to perform recovery. The optimization algorithm reacts to a SINR fitness function, and a failure to anti-jam is detected by a significant drop in SINR assuming ideal conditions. The fault recovery algorithm re-optimizes all of the array weights with the goal of regaining anti-jamming functionality despite the presence of hardware faults.

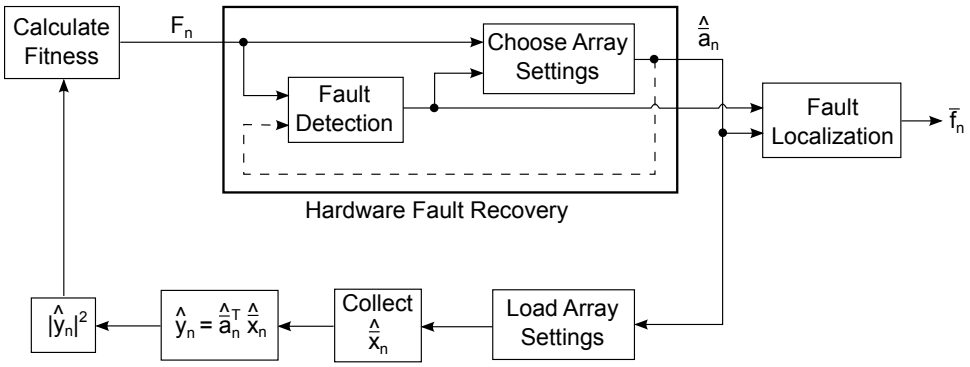


Figure 4.2: High level diagram showing hardware fault detection, recovery, and localization integrated with an optimization algorithm that performs anti-jamming beamforming.

It is shown in Figure 4.2 that the current array weights $\hat{\bar{a}}_n$ can be used in performing fault detection. If the current array weights are fed back as inputs into the algorithm, hardware fault recovery can be performed in a closed-loop fashion. Otherwise, hardware fault recovery can be performed open-loop if fault detection is

implemented using only the current fitness value F_n. Although the model shows only current values of F_n and $\hat{\bar{a}}_n$, it is possible to include memory in the fault detection function such that previous values can be used to detect faults.

4.2 Diagnostic Models for Hardware Fault Detection

This section presents the models for performing hardware fault detection. The basic fault detector shown in Figure 4.3 can be thought of as a decision device whose inputs are the current fitness function value F_n and chosen solution $S_n = \hat{\bar{a}}_n$. A closed-loop fault detector (as shown in relation to the optimization algorithm in Figure 4.2) uses both F_n and $\hat{\bar{a}}_n$ in making a decision $D_n = 0, 1$ whether or not a hardware fault exists in the array. An open-loop fault detector uses only the current fitness function value F_n in making its decision. Although fault detection is performed in software, hardware fault detection is possible because the fitness function is based on array output power measurements collected by the control system and stored in memory.

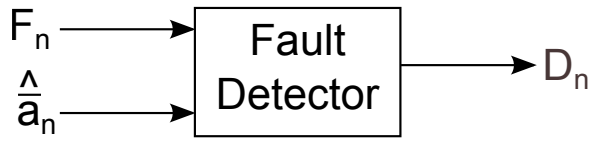


Figure 4.3: High level diagram of a hardware fault detector.

In this section, it is assumed that hardware faults are present in the array and prevent anti-jamming, and we relist the five events in Table 4.1 for reference.[1] Event

[1] A Venn Diagram is described in Figure 3.18 that represented four possible events pertaining to anti-jamming beamforming with mobile signals. This approach cannot be used to detect hardware faults because it is missing a fifth event E representing the possibility of hardware faults.

E clearly complicates the problem because its subspace overlaps with other event subspaces in the solution space S as can be seen in Figure 4.4. Event E is represented by dotted filled shapes. The subspace areas including their intersections are not drawn to scale.

Table 4.1: Five events that can occur while a stochastic algorithm adapts an anti-jamming beamforming array subjected to hardware faults.

Event	Event Description
A	Signal DOAs changed.
B	SINR fitness dropped.
C	The algorithm chose a worse solution.
D	The wireless channel faded.
E	Hardware faults existed in the array.

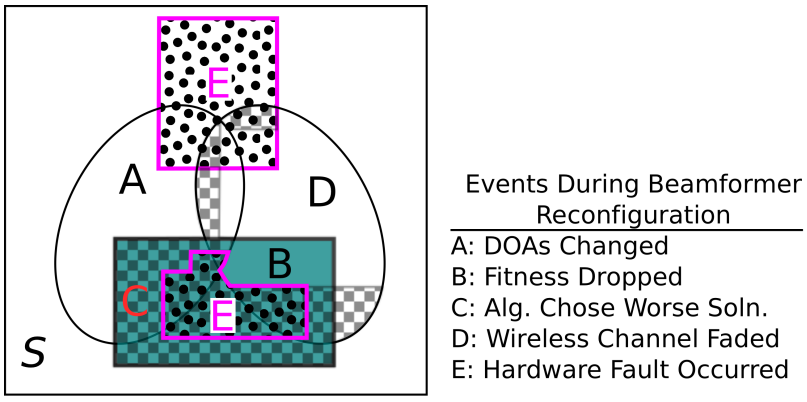


Figure 4.4: Venn Diagram showing the relationship between events A, B, C, D, and E in the parameter search space S.

To explain the significance of the subspace intersections, we consider several cases given that a fault (event E) occurs:

1. Stationary signals, no fading (event E): If SINR increases, the algorithm finds

a better solution (event E alone). However, if SINR decreased, the algorithm always finds a worse solution. This corresponds to the subspace $E \cap B \cap C$.

2. Stationary signals in a fading channel (subspace $E \cap D$): Fading creates situations where increases SINR can lead to worse solutions (i.e., faded jammer, subspace $E \cap D \cap C$) or better solutions (subspace $E \cap D$). However, solutions with decreased SINR are always worse than their predecessors (subspace $E \cap D \cap B \cap C$). The reasoning is that only hardware faults or faded SOI degraded SINR.
3. Mobile signals, no fading (subspace $E \cap A$): If SINR increases, this means that the algorithm found a better solution despite presence of hardware faults and mobile signals. This is represented by the subspace $E \cap A$. However, if a new solution's SINR is worse than its predecessor, this means that a new solution is always worse than its predecessor because the new solution moved farther away from the new global optimum. This is represented by the subspace $E \cap A \cap B \cap C$.
4. Mobile signals in a fading channel (Subspace $E \cap A \cap D$): If SINR increases, the current solution may or may not be better than its predecessor. If a jammer fades, this results in a higher SINR even though the solution is farther away from the current global minimum (subspace $E \cap A \cap D \cap C$). Solutions with worse SINR always represent worse solutions (subspace $E \cap A \cap D \cap B \cap C$) because either a hardware fault cause a worse solution to be chosen, or the signals move away from the new global optimum.

A portion of the event E subspace is completely submerged in the subspace $C \cap B$. This means that the probability that SINR decreased $P(B)$ as described by equations (3.24) – (3.26) is unaffected by the presence of hardware faults. This is an indication that faults are indistinguishable from time-varying DOAs (TVDOAs) in a Bayesian probabilistic sense. Therefore, tracking the fitness function alone is insufficient to diagnose what caused the algorithm to fail to configure the array weights for anti-jamming, and additional information is necessary to understand why and how the algorithm failed. It is possible, however, to distinguish hardware faults from TVDOAs by tracking the complex array weights $\hat{\bar{a}}_n$ along with the fitness function values F_n. Assuming that the fault detector has internal memory, let

$$\bar{F}_{ns} = \{F_i,\, F_{i+1},\, \cdots,\, F_n\}^T \tag{4.1}$$

$$\hat{\bar{\bar{a}}}_{ns} = \left\{\hat{\bar{a}}_i,\, \hat{\bar{a}}_{i+1},\, \cdots,\, \hat{\bar{a}}_n\right\}^T \tag{4.2}$$

where $i = [n_s, n]$, and $1 < n_s < n$ is chosen such that $N_{\text{samp}} = n - n_s + 1$. The simple (i.e., trivial) detector assumes that the algorithm converged such that $\sigma_{an}^2 = \text{var}\left[\hat{\bar{\bar{a}}}_{ns}\right] \approx 0$. It assumes that the wireless channel is AWGN with no fades, and the signals did not change their locations and directions. The simple detector therefore is equivalent to an open-loop detector in the sense that it monitors only equation (4.1) after the algorithm converged, and it operated on a decision rule

$$D_n = \begin{cases} 1,\, \text{var}\left[\bar{F}_{ns}\right] \gg 0 \\ \\ 0, \text{Otherwise} \end{cases} \tag{4.3}$$

Thus the simple detector is unable to distinguish between hardware faults and TV-

DOAs, as both cause variance in the fitness function values given that the algorithm converged. This detector is trivial because it does not have sufficient information to separate hardware faults from TVDOAs, and it is unable to detect either hardware faults or TVDOAs if the algorithm has not converged. Channel fades can also mislead this detector into falsely detecting hardware faults.[2]

Accounting for the algorithm's state of convergence, a more sophisticated fault detector is based on a four-way hypothesis test:

- $\mathcal{H}_0$: Algorithm converged with no faults and no TVDOAs.
- $\mathcal{H}_1$: Algorithm not converged with no faults and no TVDOAs.
- $\mathcal{H}_2$: Algorithm converged with faults and/or TVDOAs.
- $\mathcal{H}_3$: Algorithm not converged with faults and/or TVDOAs.

This detector assumes that the wireless channel is AWGN with no fades for simplicity. Hardware faults and TVDOAs are grouped together because it would require an eight way hypothesis test when faults and TVDOAs are separated. We discuss a method of discriminating hardware faults from TVDOAs later in this section. Let

$$\mathbf{F}_{an} \;=\; \bar{F}_{ns} \;+\; \hat{\bar{a}}_{ns} \tag{4.4}$$

$$\text{var}\left[\mathbf{F}_{an}\right] \;=\; \text{var}\left[\bar{F}_{ns} \;+\; \hat{\bar{a}}_{ns}\right] \tag{4.5}$$

Because $\mathbf{F}_{an}$ and $\hat{\bar{a}}_{ns}$ are clearly dependent, it follows that equation (4.5) can be

[2]It assumes that the optimization algorithm follows a convergence rule such that $\hat{\bar{a}}_n \to W$ as $n \to \infty$ where W is a complex valued constant. Thus, channel fades do not affect $\hat{\bar{a}}_n$ after the algorithm converges.

rewritten as

$$\text{var}\left[\mathbf{F}_{an}\right] \;=\; \text{var}\left[\bar{F}_{ns}\right] \;+\; \text{var}\left[\hat{\bar{\bar{a}}}_{ns}\right] \;+\; 2\,\text{cov}\left[\bar{F}_{ns},\, \hat{\bar{\bar{a}}}_{ns}\right] \tag{4.6}$$

$$\text{cov}\left[\bar{F}_{ns},\, \hat{\bar{\bar{a}}}_{ns}\right] \;=\; \mathbf{E}\left[\left(\bar{F}_{ns} \;-\; \mu_F\right)\left(\hat{\bar{\bar{a}}}_{ns} \;-\; \mu_a\right)\right] \tag{4.7}$$

$$\sigma_F^2 \;=\; \text{var}\left[\bar{F}_{ns}\right] \;=\; \mathbf{E}\left[\left(\bar{F}_{ns} \;-\; \mu_F\right)^2\right] \tag{4.8}$$

$$\sigma_a^2 \;=\; \text{var}\left[\hat{\bar{\bar{a}}}_{ns}\right] \;=\; \mathbf{E}\left[\left(\hat{\bar{\bar{a}}}_{ns} \;-\; \mu_a\right)^2\right] \tag{4.9}$$

$$\mu_F \;=\; \mathbf{E}\left[\bar{F}_{ns}\right] \tag{4.10}$$

$$\mu_a \;=\; \mathbf{E}\left[\hat{\bar{\bar{a}}}_{ns}\right] \tag{4.11}$$

Using equation (4.7), the correlation coefficient for $\bar{F}_{ns}$ and $\hat{\bar{\bar{a}}}_{ns}$ can be written as

$$\rho_{Fa} \;=\; \frac{\text{cov}\left[\bar{F}_{ns},\, \hat{\bar{\bar{a}}}_{ns}\right]}{\sqrt{\text{var}\left[\bar{F}_{ns}\right] \cdot \text{var}\left[\hat{\bar{\bar{a}}}_{ns}\right]}} \tag{4.12}$$

The correlation coefficient is limited to $-1 \leq \rho_{Fa} \leq +1$, it is zero when the fitness functions and complex array weights are uncorrelated (i.e., equation (4.7) is zero), and is undefined when either variances is zero. Based on equations (4.8), (4.9), and (4.12), the hypothesis is chosen based on

$$\text{Choose} \begin{cases} \mathcal{H}_0 \text{ if } \text{var}\left[\hat{\bar{\bar{a}}}_{ns}\right] \to 0 \;\&\; \text{var}\left[\bar{F}_{ns}\right] \to 0 \;\; (\rho_{Fa} \text{ undefined}) \\ \mathcal{H}_1 \text{ if } \text{var}\left[\hat{\bar{\bar{a}}}_{ns}\right] \gg 0 \;\&\; \text{var}\left[\bar{F}_{ns}\right] \gg 0 \;\&\; \rho_{Fa} > 0 \\ \mathcal{H}_2 \text{ if } \text{var}\left[\hat{\bar{\bar{a}}}_{ns}\right] \to 0 \;\&\; \text{var}\left[\bar{F}_{ns}\right] \gg 0 \;\; (\rho_{Fa} \text{ undefined}) \\ \mathcal{H}_3 \text{ if } \text{var}\left[\hat{\bar{\bar{a}}}_{ns}\right] \gg 0 \;\&\; \text{var}\left[\bar{F}_{ns}\right] \gg 0 \;\&\; \rho_{Fa} \leq 0 \end{cases} \tag{4.13}$$

The conditions behind selecting the null hypothesis are evident. The algorithm con-

verges if it selects roughly the same complex array weights with approximately equal fitness over a time-sampling window. For selecting hypothesis $\mathcal{H}_1$, var $\left[\hat{\bar{\bar{a}}}_{ns}\right] \gg 0$ indicates that the algorithm did not converge by evaluation n. The combination of var $\left[\bar{F}_{ns}\right] \gg 0$ with a covariance coefficient $\rho_{Fa} > 0$ indicates that the algorithm is searching the parameter space and likely choses the current complex array weights $\hat{a}_n$ based on the previous solution's fitness F_{n-1}. A positive covariance coefficient indicates a positive correlation between the most recent complex weights $\hat{\bar{\bar{a}}}_{ns}$ and their associated fitness $\bar{F}_{ns}$. This is expected for normal operation of an optimization algorithm without external stimuli such as hardware faults and TVDOAs.

The conditions used to select hypothesis $\mathcal{H}_2$ indicates that faults and/or TV-DOAs exist. The condition that var $\left[\hat{\bar{\bar{a}}}_{ns}\right] \to 0$ indicates that the algorithm converges, but var $\left[\bar{F}_{ns}\right] \gg 0$ means that an external stimuli causes the converged solution's fitness to change randomly. This means that either a hardware fault appeared, or the signals changed their DOAs. For hypothesis $\mathcal{H}_3$, the condition that var $\left[\hat{\bar{\bar{a}}}_{ns}\right] \gg 0$ shows that the algorithm did not converge. It is searching the parameter space because var $\left[\bar{F}_{ns}\right] \gg 0$, but a non-positive covariance coefficient indicates that the relationship between the current solution and previous solutions' fitness function values are not clear. The solutions and their fitness are either uncorrelated or negatively correlated. This means that a hardware fault or a TVDOA affected the algorithm's search for an optimal solution.

The decision rule described by equation (4.13) accounts for the algorithm's con-

vergence state, but it still did not separate hardware faults from TVDOAs. It assumes that the wireless channel is AWGN, and channel fades cause variances in fitness that in turn cause variations in the complex array weights. If the number of samples, however, are large enough, then variations due to fading can be smoothed out. To distinguish hardware faults from TVDOAs, a third metric is necessary. Let $\bar{\mu}$ and its derivative be defined by

$$\bar{\mu} = [\mu_a, \mu_F]^T \tag{4.14}$$

$$\frac{\partial \bar{\mu}}{\partial t} = \left[\frac{\partial \mu_a}{\partial t}, \frac{\partial \mu_F}{\partial t}\right]^T \tag{4.15}$$

Because the means of the complex array weights and the fitness function values both vary with time, the process is non-stationary. We hypothesize that it is possible to distinguish between faults and TVDOAs by observing the derivatives of the means as described by equation (4.15) in conjunction with equations (4.8), (4.9), and (4.12).

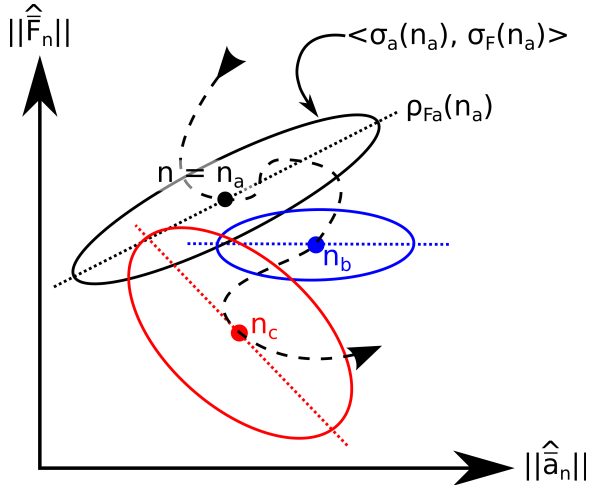


Figure 4.5: Example of time-varying nature of complex array weights and their associated fitness values with time-varying means, variances, and correlation coefficients.

To explain this concept, begin with a depiction of an algorithm's behavior prior to convergence in Figure 4.5. The ellipses represents the standard deviations of the complex array weights and fitness values evaluated over N_{samp} evaluations with $n = n_a,\ n_b,$ and n_c. The ellipses' centers represent the means (see eq. (4.14)) of the array weights and their fitness at $n_a,\ n_b,$ and n_c. The ellipses' slopes represent the correlation coefficient ρ_{Fa} evaluated at $n_a,\ n_b,$ and n_c. The general idea is that the complex array weights and their associated fitness are not stationary due to external factors such as hardware faults and TVDOAs, and the means followed a trajectory defined by those factors.

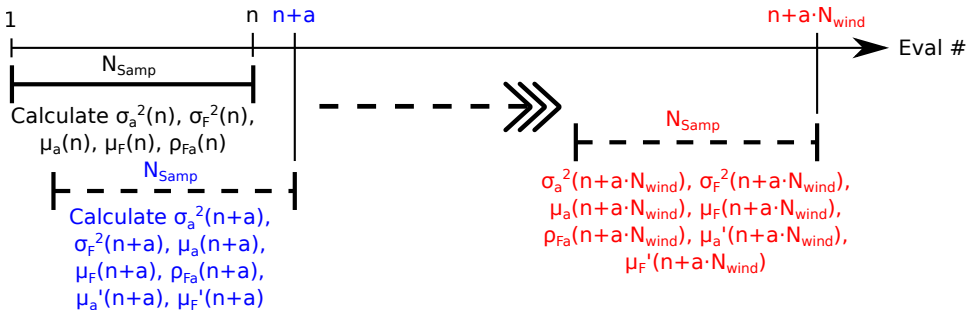


Figure 4.6: Sliding windows used to calculate time-varying means, variances, and correlation coefficients associated with the time-varying array weights and fitness functions.

The time-varying statistical quantities $(\sigma_a^2(n),\ \sigma_F^2(n),\ \mu_a(n),\ \mu_F(n))$ and the derivatives of the means are calculated using a sliding window of sample length N_{Samp} as depicted in Figure 4.6. The offset parameter $a \geq 1$ controls the amount of overlap between windows. Overlapping windows produce responses that track changes in the statistical quantities more rapidly where $a = 1$ gives instantaneous responses. Overlapping windows are more susceptible to noise due to channel fades, and this

causes $\bar{\mu}\left(n + k \cdot N_{Samp}\right),\ k \in [1, N_{Wind}]$ to be noisy. From a probabilistic viewpoint, the statistical quantities for the k^{th} and $(k+1)^{\text{th}}$ windows with $1 \leq a \leq N_{Samp}$ are dependent because the calculations used common samples.

As the algorithm converge, $\frac{\partial \mu_a}{\partial t}(n) \rightarrow 0$ along with $\text{var}\left[\hat{\bar{a}}_{ns}\right] \rightarrow 0$, and the second hypothesis $\mathcal{H}_2$ is chosen if the fitness variance changes per equation (4.13). If $\frac{\partial \mu_F}{\partial t}(n) > 0$, the converged solution's $(\hat{a}_f)$ fitness increases. Because a hardware fault can only decrease the solution's fitness, this means that a signal changed directions in a way that moved the converged solution closer to a new global optimum. However, if $\frac{\partial \mu_F}{\partial t}(n) \leq 0$, the detector cannot discriminate between hardware faults and TVDOAs because both factors can cause decreased fitness function values.

If the algorithm does not converge when faults or TVDOAs are detected, the decision rule of equation (4.13) chooses hypothesis $\mathcal{H}_3$, and the behavior of $\frac{\partial \mu_a}{\partial t}(n)$ and $\frac{\partial \mu_F}{\partial t}(n)$ are split into four quadrants:

1. Q_1: $\frac{\partial \mu_a}{\partial t}(n) > 0$ and $\frac{\partial \mu_F}{\partial t}(n) > 0$
2. Q_2: $\frac{\partial \mu_a}{\partial t}(n) > 0$ and $\frac{\partial \mu_F}{\partial t}(n) \leq 0$
3. Q_3: $\frac{\partial \mu_a}{\partial t}(n) \leq 0$ and $\frac{\partial \mu_F}{\partial t}(n) > 0$
4. Q_4: $\frac{\partial \mu_a}{\partial t}(n) \leq 0$ and $\frac{\partial \mu_F}{\partial t}(n) \leq 0$

We discuss these four cases below. Note that we assume the beamforming array uses passive weights. This means that $||\hat{\bar{a}}_n|| \leq 1\, \forall n$, and the complex array weights occupy an n-dimensional hypersphere with radius of one centered at the origin.

4.2.1 Time Averaged Array Weights and Fitness Both Increasing

If $\frac{\partial \mu_F}{\partial t}(n) > 0$, it clearly follows that the fitness function values generally in-

crease over the last N_{samp} evaluations leading up to evaluation n. The condition that $\frac{\partial \mu_F}{\partial t}(n) > 0$ means that the algorithm chose solutions that moved away from the origin towards the edge of the unit-radius hypersphere as shown in Figure 4.7.

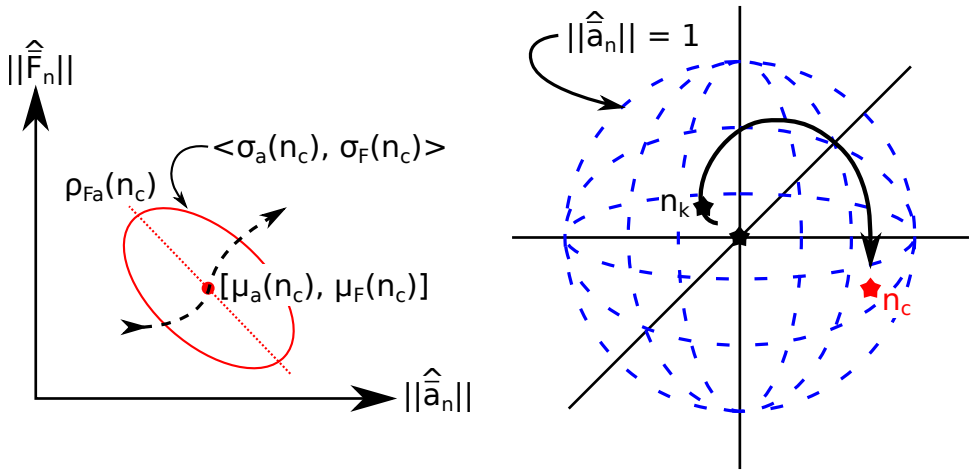


Figure 4.7: An example of an unconverged algorithm in a hypothesis $\mathcal{H}_3$ state with increasing time-averaged array weights and time-averaged fitness (left). The solutions chosen by this algorithm moved from near the origin of the unit-radius hypersphere to its outer edge as $n_k \to n_c$ where $k < c$ (right).

In relationship with var $\left[\hat{\hat{\bar{a}}}_{ns}\right]$ and $\frac{\partial \mu_F}{\partial t}(n) > 0$, this implies that the algorithm compensated for a hardware fault. The fault causes a portion of the complex array weight vector to become ineffective (i.e., dead weight), so the algorithm needs a solution with additional weighting to compensate for that loss. Although TVDOAs inherently change the latest solutions up to $\hat{\bar{a}}_n$, its magnitude is roughly the same as the last N_{samp} solutions because the complex weight vector is tracking the DOAs on a constant radius sphere within the unit-radius hypersphere if the fitness trended towards larger values.

Because fault detection is inherently stochastic, it is useful to set a bound on

determining how well this metric is in discriminating faults from TVDOAs in hypothesis $\mathcal{H}_3$. This bound is difficult to determine when considering that events (see Figure 4.4) occurred concurrently, and the hypothesis $\mathcal{H}_3$ region is divided into four subregions for a total of eight hypotheses[3]. However, it is possible to approximate the performance of this detector by selecting two hypothesis. Namely, either hypothesis $\mathcal{H}_0$ occurs, or hypothesis $\mathcal{H}_3$ with the constraints that $\frac{\partial \mu_a}{\partial t}(n) > 0$ occurs.

Letting $\hat{\bar{x}}_n = \left[\hat{\bar{a}}_n, \bar{F}_n\right]^T$, $A_{31} = \{\mathcal{H}_3 \cap Q_1\}$ be the set of points in $\mathcal{H}_3$ with $\frac{\partial \mu_a}{\partial t}(n) > 0$ and $\frac{\partial \mu_F}{\partial t}(n) > 0$, an approximate bound is the Neyman-Pearson Test described in [71]:

$$\hat{\bar{x}}_n \in \mathcal{H}_0 \text{ if } \mathrm{L}\left(\hat{\bar{\mathrm{x}}}_\mathrm{n}\right) = \frac{\mathrm{P}\left(\hat{\bar{\mathrm{x}}}_\mathrm{n}|\mathcal{H}_0\right)}{\mathrm{P}\left(\hat{\bar{\mathrm{x}}}_\mathrm{n}|\mathcal{H}_3, \mathrm{Q}_1\right)} \geq \gamma; \quad \hat{\bar{\mathrm{x}}}_\mathrm{n} \in \mathrm{A}_{31} \text{ Otherwise} \tag{4.16}$$

$$P_{\mathrm{FA}} = \sum_{L\left(\hat{\bar{x}}_n\right) < \gamma} P\left(\hat{\bar{x}}_n|\mathcal{H}_0\right) \leq \alpha \tag{4.17}$$

4.2.2 Time Averaged Array Weights Increasing, Time Averaged Fitness non-Increasing

Recall from the definition of hypothesis $\mathcal{H}_3$ that either a hardware fault occurred, or a signal's DOA is time-varying. The additional conditions that $\frac{\partial \mu_a}{\partial t}(n) > 0$ and $\frac{\partial \mu_F}{\partial t}(n) \leq 0$ means the algorithm is searching the parameter space and finding solutions with lower fitness. Although it is possible that a hardware fault occurred, it is also possible that μ_a increases due to the algorithm finding worse solutions during its search. A non-increasing $\frac{\partial \mu_F}{\partial t}(n)$ therefore prevented the detector from discriminating between faults and TVDOAs.

[3]Hypothesis $\mathcal{H}_2$ should also be divided into four subregions. However, doing so is unnecessary since faults cannot be distinguished from TVDOAs once the algorithm converged.

4.2.3 Time Averaged Array Weights non-Increasing, Time Averaged Fitness Increasing

Because $\frac{\partial \mu_F}{\partial t}(n)$ is increasing while $\frac{\partial \mu_a}{\partial t}(n)$ either decreased or remained constant, this implies that TVDOAs are present. The solutions move towards the center of the array weight hypersphere or stay on a constant radius. The solution finds better solutions with lower magnitudes. If the channel fades, it assumes that N_{Samp} is large enough to smooth out fading effects both in the means (μ_a, μ_F), means derivatives ($\frac{\partial \mu_a}{\partial t}$, $\frac{\partial \mu_F}{\partial t}$), and variances (var $\left[\hat{\bar{a}}_{ns}\right]$, var $\left[\bar{F}_{ns}\right]$).

4.2.4 Time Averaged Array Weights and Fitness Both non-Increasing

This case represents the algorithm searching the parameter space, but it searches closer to the origin of the complex array weight hypersphere. The fitness function values are on average decreasing (indicated by $\frac{\partial \mu_F}{\partial t}(n) \leq 0$). When combined with the hypothesis $\mathcal{H}_3$ condition that $\rho_{Fa} \leq 0$, this indicates that the solutions that the algorithm found are on average getting worse. This can have been caused by either hardware faults or TVDOAs. Under this set of conditions, the detector can not distinquish between faults and TVDOAs.

The Neyman-Pearson test described by (4.16) to test the effectiveness of this method in (4.13) requires that the critical regions under test be well defined, and it does not test the effectiveness of the other hypothesis. An alternative means of evaluating the system described by (4.13) is to treat the system as a Markov chain with

three binary parameters: state of algorithm convergence (converged or not converged), hardware faults present vs. not present, and TVDOAs present vs. DOAs static. The Markov chain had eight states as shown in Figure 4.8 and Table 4.2.

Table 4.2: State description of generalized Markov chain describing the system's probabilistic behavior when subjected to possible faults and TVDOAs.

State	Binary Value ($h_C\, h_F\, h_T$)	Meaning
S0	000	Unconverged, no faults, static DOAs
S1	001	Unconverged, no faults, TVDOAs
S2	011	Unconverged, faults, TVDOAs
S3	010	Unconverged, faults, static DOAs
S4	110	Converged, faults, static DOAs
S5	100	Converged, no faults, static DOAs
S6	101	Converged, no faults, TVDOAs
S7	111	Converged, faults, TVDOAs

The binary values associated with the states are Grey encoded. The total number of hops in the generalized Markov chain is 64, and the hops between states are color coded: black represented one parameter change between states, blue represented two parameter changes, and red represented three parameter changes. The generalized model allows the possibility of intermittent faults with hops from faulty to non-faulty states shown as dashed lines. The generalized model also allows for the algorithm to change from converged to unconverged. This is shown with double lines. Per [71], the transition probabilities between states are described by

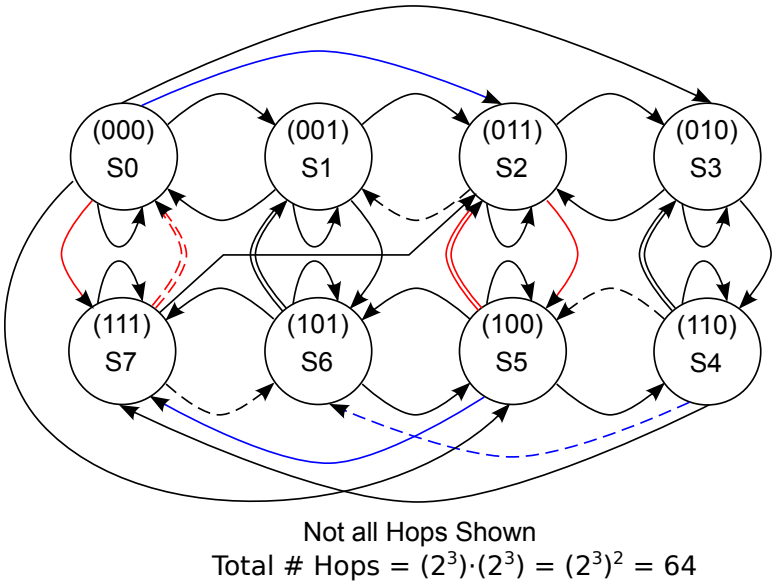


Figure 4.8: A generalized Markov chain showing the system's probabilistic states when subjected to possible faults and TVDOAs.

$$P_{ij} = P\left[S_{n+1} = j | S_n = i\right] \tag{4.18}$$

$$\sum_{j=0}^{St_{\text{tot}}} P_{ij} = 1 \quad \text{where } \mathrm{P_{ij}} \geq 0 \ \forall \mathrm{i,j} \tag{4.19}$$

The Markov model shown in Figure 4.8 is generalized because it assumes that $P_{ij} > 0$ for all possible hops between states and included hops that are unlikely (i.e., hop from state 5 to state 0 where the algorithm changed from converged to unconverged for no reason). The generalized Markov model also allows for intermittent faults. Although it is possible to have intermittent faults in a real system, it is assumed that intermittent faults do not exist to simplify the model. The first step in calculating the transition probability matrix $\mathbf{P}$ is determine unlikely (or disallowed) hops and set their transition probabilities to zero. Noting the number of parameter changes from state i to state j as $\mu_{ij} \in [0, \mu_{\text{Max}}]$ where $\mu_{\text{Max}} = 3$ in this model, and $h_P \in [0, 1]$

where $P = \{C, F, T\}$, the rules for simplifying the Markov chain in Figure 4.8

$$P_{ij} = 0 \text{ iff } \begin{cases} \mu_{ij} = \mu_{\text{Max}}, \text{ or} \\ h_F(i) = \neg h_F(j) = 1, \text{ or} \\ (h_C(i) = \neg h_C(j) = 1) \wedge (h_F(i) = \neg h_F(j) = 0), \text{ or} \\ (h_C(i) = \neg h_C(j) = 1) \wedge (h_T(i) = \neg h_T(j) = 0) \end{cases} \tag{4.20}$$

The first rule indicates that a hop where all of the parameter values changed is so unlikely that is is virtually impossible. The second rule disallows intermittent faults. The third and fourth rules specifies that the model cannot hop from a converged state i to an unconverged state j if either a fault appeared in state j or DOAs started moving in state j. Because a stochastic algorithm tends to stay converged once in a converged state, the detector needs to issue a command that causes the algorithm to entire an unconverged state. The algorithm is either completely reinitialized or partially initialized with memory of the last known best solution. In other words, faults and/or TVDOAs need to be detected before the algorithm can re-enter an unconverged state from a converged state.

4.3 Hardware Fault Recovery Model

In this section, we develop a new model for hardware fault recovery. A simplified model of this process is shown in Figure 4.9.

As is shown in Chapter 6, various stochastic algorithms (such as the GA) op-

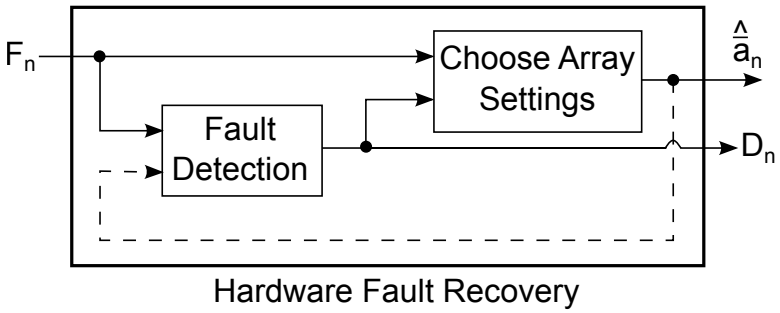


Figure 4.9: A high level model showing hardware fault recovery.

erate in a closed loop and automatically feedback the current array weights $\left(\hat{\bar{a}}_n\right)$ at generation n to create the next set of solutions $\hat{\bar{a}}_{n+1}$. An example of the TDGA automatically detecting and often recovering from a fault is shown in Figure 4.10. In this example, the strings each have a fitness $F \in [0, 100]$ where a fitness of 0 means that the string is selected as a mate with probability 0, and a fitness of 100 means that the string is selected for mating with probability 1.

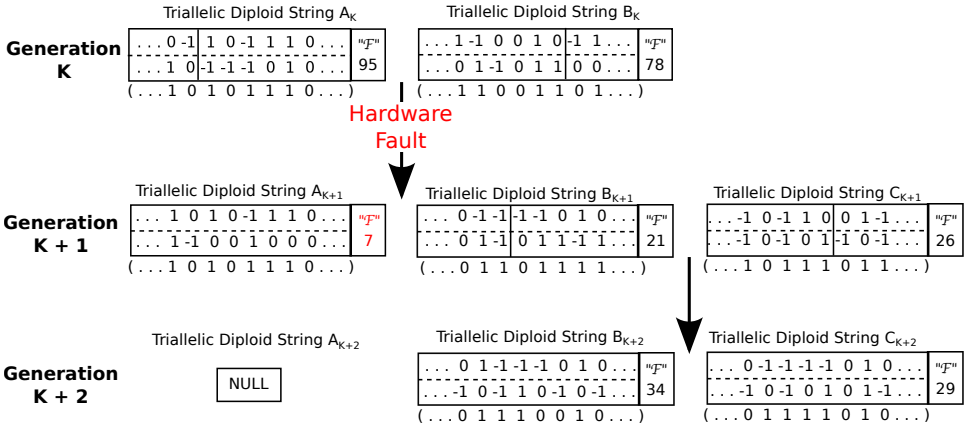


Figure 4.10: Example showing how the TDGA recovers from hardware faults in an antenna array.

Assuming that by generation K the TDGA nears convergence, the strings have high fitness as shown in Figure 4.10. Two example strings (A_K and B_K where the

subscript represented the strings' bitwise values at generation K) are shown to have a fitness of 95 and 78, and are selected for mating. The triallelic string values are shown in the boxes along with the crossover locations, and the encoded haploid values are shown below in parenthesis. Since a hardware fault occurs between generations K and $K+1$, the resulting children strings (A_{K+1} and B_{K+1}) have much lower fitness than their parents.

Suppose that the fitness of string A_{K+1} is too low for mate selection, so a new string C_{K+1} is chosen from the current population to mate with string B_{K+1}. Because string A_{K+1} is not chosen for mating, it effectively dies off. The resulting children strings (B_{K+2} and C_{K+2}) in the second generation following the fault's occurrence have higher fitness than their parents, so the TDGA tends towards a higher fitness averaged over the entire population. It should be noted from the example that prior to the fault, a +1 dominance occurred in the population in the sense that more dominant ones (i.e., +1's) existed than recessive ones (i.e., -1's). However, in the second generation after the fault, it can be seen that the role shifted to a -1 dominance since the recessive ones began to outnumber the dominant ones in the population. A similar effect is noted by Weile and Michielssen [28] in adapting to mobile signals, so the mechanism that the TDGA uses to recover from hardware faults is the same as the one it uses to adapt to mobile signals.

As we will show in Chapter 5, the SGA is able to adapt to mobile signals following convergence through use of its mutation operator. Although the mutation operator

also assists the TDGA in adapting to mobile signals and to faults, it will be shown that the SGA can adapt to hardware faults via the mutation operator. Because the SGA reacts automatically to its environment, the SGA uses the closed-loop fault recovery method as shown in Figure 4.9.

Two methods exist for allowing SA and HCA to perform hardware fault recovery. The first method assumes that recovery can be achieved provided that the fault occurs when the algorithm has not converged. This method is open-loop in the sense that it repeats the temperature schedule multiple times while SA or HCA operated, and it operates without knowledge of the current array weights $\bar{\hat{a}}_n$. There exists a trade-off between the number of times that the temperature schedule is repeated during a given maximum set of evaluations: increases likelihood to convergence to a local maximum instead of global maximum versus the detection probability and response time to a fault. In other words, if the temperature schedule is repeated too many times, SA and HCA will stagnate on suboptimal solutions. If the temperature is allowed to stay cooled for too long, a fault can go undetected, and the converged solution will be invalid (even if SA or HCA reports that that solution maintains a high fitness value).

The second method is closed-loop, and in a manner similar to the detection of mobile signals, the fault-detection model can provide a posterior probability that a fault occurred. An issue behind this approach is that the algorithm cannot in all cases discern faults from mobile signals. This is evident from the complexity of the Venn diagram shown in Figure 4.4 and the diagnosis model discussed in Section 4.2

based on a four-way hypothesis test. Based on these reasons, we chose to implement the open-loop model for hardware fault detection and recovery with SA and HCA, and save implementation of the closed-loop model for future work.

4.4 Hardware Fault Localization

To localize a hardware fault, it is generally necessary to collect complex voltage samples at the output of each antenna branch prior to the summing point shown in Figure 4.1. A problem here is that the complex antenna branch voltages are summed, and only the summed output voltage is available. However, if the antenna array is small in size, it is possible to localize the fault by calculating the cross-correlation of the array's measured radiation pattern with the array's calculated radiation pattern given that element k is removed from an N element array of arbitrary shape. Let this radiation pattern be defined as

$$AR_F\left(\phi,\, \theta \,|\, k\right) \;=\; EF\left(\phi,\, \theta\right) \cdot AF\left(\phi,\, \theta \,|\, k\right) \tag{4.21}$$

$$AF\left(\phi,\, \theta \,|\, k\right) \;=\; \sum_{i=1}^{k-1} \hat{a}_i e^{j\beta \bar{a}_R \cdot \bar{d}_i} \;+\; \sum_{i=k+1}^{N} \hat{a}_i e^{j\beta \bar{a}_R \cdot \bar{d}_i} \tag{4.22}$$

where $EF\left(\phi,\, \theta\right)$ is the element factor common to all of the array elements. Letting the array's measured radiation pattern at a constant θ_o (i.e., the azimuth plane with $\theta_o \;=\; \pi/2$ where $\theta \;\in\; [0, \pi]$) be $AR_M\left(\phi,\, \theta_o\right)$, the probability that a fault in the array occurred given that antenna element k is the element that faulted is defined as

$$P_{\text{Flt}}\left(k \,|\, Fault\right) \;=\; \frac{1}{\xi} \max \left| \int_{-\pi}^{+\pi} AR_F\left(\phi,\, \theta_o \,|\, k\right)^* AR_M\left(\phi \,+\, \tau,\, \theta_o\right) d\tau \right| \tag{4.23}$$

where $\frac{1}{\xi}$ is a normalizing factor to ensure that $P_{\text{Flt}}\left(k \,|\, Fault\right) \in [0, 1]$. Equation (4.23) is the cross-correlation in a single spherical plane and can be extended to the entire sphere by applying a second integral over θ with a dummy variable τ_2. Replacing the integral in equation (4.23) with an operator called xcorr, the analysis can be extended to multiple faults:

$$P_{\text{Flt}}\left([k_a,\, k_b] \,|\, Fault\right) = \frac{1}{\xi} \max \left|\text{xcorr}\left[AR_F\left(\phi,\, \theta_o \,|\, [k_a,\, k_b]\right),\, AR_M\left(\phi,\, \theta_o\right)\right]\right| \tag{4.24}$$

$$P_{\text{Flt}}\left([k_a,\, \cdots,\, k_J] \,|\, Fault\right) = \frac{1}{\xi} \max \left|\text{xcorr}\left[AR_F\left(\phi,\, \theta_o \,|\, [k_a,\, \cdots,\, k_J]\right),\, AR_M\left(\phi,\, \theta_o\right)\right]\right| \tag{4.25}$$

Equation (4.24) represents the probability that two faults occurred in the array with antenna elements k_a, k_b effectively removed from the array, and equation (4.25) represented the probability that up to most $N - 1$ antenna element faults occur in the array. There are three issues with implementing equations (4.23) – (4.25). First, this method is based on array factor and does not account for mutual coupling between elements. Second, it only considers complete antenna faults such that faulty elements are completely removed from the array. This method does not consider partial faults such as an antenna element that is damaged, a step attenuator or phase shifter stuck-at a previous setting, or a step-attenutor or a phase shifter stuck-at a non-controlled setting. Third, the method is computationally intense as both the number of faulty

elements and the array size N increases.

The third issue is addressed by calculating the total number of correlations $C_{\max}$ required given N antennas and at most $N-1$ faults. Because it is more likely that the fault vector containing the element numbers of faulted elements is sparse (see [14]), the number of correlations C_S given sparsity level S such that $\lceil S \cdot N \rceil \geq 1$:

$$C_{\max} = \sum_{J=1}^{N-1} \binom{N}{J} \tag{4.26}$$

$$C_S = \sum_{J=1}^{\lceil S \cdot N \rceil} \binom{N}{J} \tag{4.27}$$

The number of maximum correlations required given the number of elements in the array N as well as the number of correlations required given a sparsity coefficient S are listed in Table 4.3. The number of elements considered varied from 4 to 32, and the sparsity levels considered are 10%, 20%, and 30%.

Table 4.3: Number of correlations needed to localize K faulty elements in a N-element antenna array.

		C_S with Sparsity Coefficient, S		
N	**$C_{\max}$**	**10%**	**20%**	**30%**
4	14	4	4	10
8	254	8	36	92
10	1022	10	55	175
12	4094	78	298	793
16	65534	136	2516	6884
32	4.295E+09	41448	4514872	1.076E+08

It is clear from Table 4.3 that the total number of correlations needed assuming

at most $N - 1$ faults in the array grew exponentially in N, and this method is infeasible for arrays with at least 32 elements because $C_{\mathrm{Max}} > 4$ trillion combinations at that array size. However, assuming a reasonable level of sparsity in the element fault list of 10% to 20%, this method is reasonable for arrays with 12 or 16 elements.

To compensate for mutual coupling between antenna elements and partial element faults, the method Weile and Linden explore in [72] can be extended to calculate the fields given faulty (partially and fully) elements when calculating the array's radiation pattern after a single run of a MOM program such as WIPL-D. In [72], the antenna port voltage vector $\mathbf{V}$ is calculated from a single run of WIPL-D using the resulting diagonal $N \times N$ port impedance matrix $\mathbf{Z}_o$, the $N \times N$ coupled impedance matrix $\mathbf{Z}$, and the $\Psi \in \{\theta, \phi\}$ component patterns $E^{\mathrm{i}}_{\Psi}(\theta, \phi)$ for each antenna element $i \in [1, N]$ as a superposition:

$$\mathbf{V} = \mathbf{Z}\left(\mathbf{Z} + \mathbf{Z}_{\mathrm{o}}\right)^{-1}\mathbf{V}_S \tag{4.28}$$

$$E_{\Psi}(\theta, \phi) = \sum_{i=1}^{N} V_{\mathrm{i}} E^{\mathrm{i}}_{\Psi}(\theta, \phi), \ \Psi \in \{\theta, \phi\} \tag{4.29}$$

Given that the method in [72] calculates the radiation pattern for an array phased with a complex port voltage vector $\mathbf{V}_S$, equation (4.29) can be modified to include $K \in [1, N-1]$ partially or fully faulty elements as a superposition

$$E_{\Psi}(\theta, \phi) = \sum_{\substack{i=1 \\ i \neq k}}^{N} V_{\mathrm{i}} E^{\mathrm{i}}_{\Psi}(\theta, \phi) + \sum_{k=1}^{K} V_{\mathrm{k}} E^{\mathrm{k}}_{\Psi}(\theta, \phi), \ \Psi \in \{\theta, \phi\} \tag{4.30}$$

A fully faulty antenna element can be emulated in equation (4.30) by setting $V_{\mathrm{k}} = 0$ for all faulty elements in the $N \times 1$ source voltage vector $\mathbf{V}_{\mathrm{S}}$. A partial fault is

emulated by setting V_{k} to an expected faulted value not included in the faultless source voltage vector.

4.5 Theoretical Limits on Recoverable SOI Array Gain

When an element is removed from an antenna array, it clearly decreases the array's maximum gain such that the new maximum gain is $(N\ -1) \times G_o$ where G_o is the maximum gain of a single antenna element. This assumes that the array consists of equal elements, and mutual coupling between the elements can be ignored. The question we investigate is whether or not it is possible to recover some (if not all) of the lost gain by retuning the array's complex array weights.

Initially, the research focuses on deriving a bound on the recoverable SOI array gain by using the concept of superdirectivity in which the antenna elements are arranged in a manner such that the SOI gain $G_{\mathrm{S}} > N \times G_o$ (see [3]). Per Newman *et al.* [73], the array's supergain is maximized when the array sensitivity factor K is maximized. Gilbert and Morgan originally define sensitivity factor in [74] as

$$K \ = \ \frac{\sum\limits_{i=1}^{N} |\hat{a}_i|^2}{\sum\limits_{i=1}^{N} \left|\hat{a}_i \, e^{+j\beta \bar{a}_R \cdot \bar{d}_i}\right|^2} \tag{4.31}$$

The definition of the array sensitivity factor in equation (4.31) is essentially the ratio of the sum of the absolute value squared of the array weights to the array factor squared in a given direction. Although it can be seen that K can be maximized

by minimizing the $\bar{a}_R \cdot \bar{d}_i$ term in the denominator sum (as is originally intended by [74] and further developed by [73, 75, 76] in positioning elements to maximize an array's mainbeam gain), it can also be observed the array weights $\hat{\bar{a}}$ also play a role in increasing the array's mainbeam gain beyond its canonically tuned maximum.

There are several issues with using superdirectivity in determining a limit on how much SOI can be recovered after an antenna fault occurred. First, the relationship between the array sensitivity factor and supergain as defined by Newman *et al.* [73] is an empirical one, so it is unclear on what supergain is achieved when equation (4.31) is maximized. Second, the array sensitivity factor in equation (4.31) is defined around an array factor based definition of array gain, so it naturally ignored mutually coupling between antenna elements. Third, the recent developments in [75,76] focuses on applying superdirectivity to specific types of antennas or arrays, and the lack of recent theory made it apparent that this is not a viable approach to deriving limits on the post-fault recoverable SOI gain.

From theoretical observations on CCA radiation patterns in section 3.2.2, it can be seen that the maximum gain for a naturally tuned CCA (i.e., all weights set to $\hat{a}_i = 1,\ \forall i \in [1, N]$) is roughly have that of a CCA tuned to a specific direction using canonical weights. It can be noted that recovered SOI gain is limited by this difference. However, this is an empirical result limited to a specific type of array, so it is not pursued further in this thesis.

4.6 Summary

In this chapter, new models are developed for hardware fault recovery in phased antenna array systems. The problem is formulated assuming a single output port antenna array and discussed new models in hardware fault detection, recovery, and localization. The new models in fault detection and recovery are most useful in aiding stochastic algorithms that did not automatically react to faults as if they are changes to the environment. In fact, it is shown that the TDGA and SGA automatically reacted to and recovered from faults as if the faults are either changes in the environment (i.e., channel fades) or time-varying mobile signals. Hardware fault localization models that can be applied to small antenna arrays discussed. In the attempts to derive theoretical limits on how much SOI gain can be recovered, it is concluded that such a limit does not exist in the general case, and these limits are more applicable to specific array layouts.

Chapter 5

Dynamic Optimization of Beamforming Arrays with Experimental *In-Situ* and Simulation Results

This chapter presents experimental and simulation results of stochastic algorithms in dynamically focusing electromagnetic energy on a SOI while simultaneously minimizing EM energy towards interfering sources. The results in this chapter support our electromagnetic models on dynamic anti-jamming beamforming arrays, and we show the fidelity of these these models as compared against *in-situ* experimental results. We demonstrate the following.

First, we present a first time demonstration of *in-situ* optimization algorithm theory and practice to advance adaptive antenna systems. This includes genetic algorithm and simulated annealing algorithms that adapt a proof-of-concept four antenna beamforming system *in-situ* to both stationary and mobile signals. Second, the performance of stochastic search algorithms are compared in configuring an antenna array for anti-jamming beamforming with both simulations and *in-situ* experiments. The algorithms compared are genetic algorithms, simulated annealing, and hill-climbing. Third, mapping of the fitness landscape via *in-situ* measurements show that this problem is multimodal thereby rendering classical optimization algorithms such as LMS

ineffective in solving the beamforming problem. Fourth, in addition to the stepped mobile case where interfering signals change every M_{St} generations for the GA (every $P \times M_{\mathrm{St}}$ evaluations for the SA and Hill Climbing Algorithm (HCA) where P is the GA population size), it is shown that the GA is capable of real-time tracking and anti-jamming of continuously mobile jammers. Fifth, the fidelity of WIPL-D MOM models of an antenna array is compared with azimuth radiation patterns measured *in-situ* with the hardware system inside an anechoic chamber. Sixth, Hamming distances quantify the amount of genetic material that the SGA, TDGA, SA, and HCA discard during the optimization process. In particular, simulations show that the SGA is capable of tracking mobile jammers if the jammers move after the SGA converges to previous settings because the mutation operator introduced new genetic material into the population.

This chapter proceeds as follows. Section 5.1 discusses the experimental and simulation setups including a list of the anti-jamming beamforming test cases that are investigated. Section 5.3 discusses the FEM and MOM models with a discussion of the AntNet program that allowed the WIPL-D MOM simulation software to be integrated with Matlab. Section 5.4 discusses performance of the stochastic algorithms in anti-jamming with static signals, and section 5.5 quantifies algorithm performance when jammers changed directions in instantaneous steps. Section 5.6 evaluates the abilities of the algorithms to track interfering signals that continually change directions. Finally, section 5.7 summarizes the results discussed this chapter.

5.1 Experimental and Simulation Setup

The antenna array is placed inside an anechoic chamber for experimental measurements as shown in Figure 5.1. A horn antenna capable of wideband operation from 0.3 GHz to 3.0 GHz serves as SOI and Jammers since the computer sends commands to a Velmex stepper motor that rotates a turn-table in the directions of the SOI and Jammers individually. In addition to controlling the motor, the programs containing the stochastic algorithms resides on the computer and set the digital step-attenuators and analog phase shifters via National Instruments controllers.

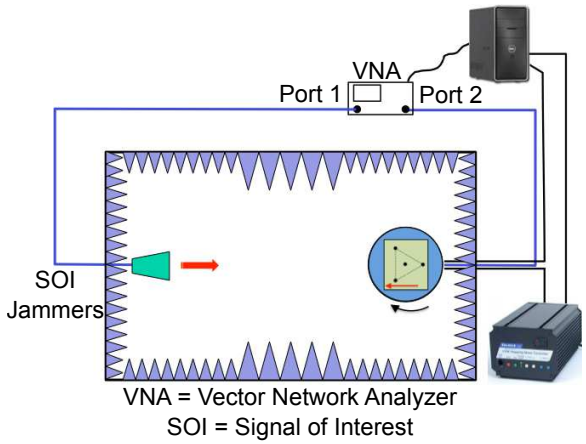


Figure 5.1: Diagram showing experimental setup of a four-antenna array inside an anechoic chamber.

In the current GA generation (SA / HCA evaluation), the program first rotates the stepper motor to the SOI and measured all hardware setting solutions that the algorithm chose in that generation (evaluation). The measurements collected are S_{21}(dB) values as seen by the VNA. The program then repeats this process by rotating the motor to the next signal direction and measuring all hardware settings in that

direction. When all hardware settings in the current generation (evaluation) are measured in the final jammer direction, the program rotates the stepper motor back to 0° and repeats the process for the next generation (evaluation). The procedure described above is repeated until the last GA generation (SA / HCA evaluation) is measured. This measurement setup and procedure is the same one used to evaluate the algorithms' abilities to perform hardware fault recovery as described in Chapter 6.

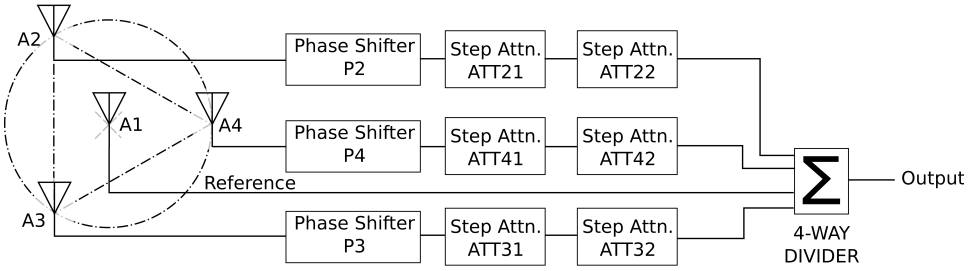


Figure 5.2: Block diagram of a four-antenna anti-jamming beamforming array.

A block diagram of the proof-of-concept system is shown in Figure 5.2. The array consists of four 2.4 GHz WiFi antennas arranged in a equilateral triangle with a reference antenna located in the center of the array. The triangle's vertices are inscribed on a circle with radius $\lambda_0/2$ where λ_0 is the free-space wavelength at 2.4 GHz. Each non-reference path includes a single Miteq 2.4 GHz analog phase shifter with $\pm 5^\circ$ tolerance and two 15.5 dB five-bit digital step-attenuators with serial-clocked latched enabled controls.

For the experiments and simulations described in this chapter, step-attenuators ATT22, ATT32, and ATT42 are set to 0 dB. When the algorithms chose hardware

settings, the phase-shifters are digitized with five bits with 11.6° per bit. With three step-attenuators and three phase shifters, the resulting setting string is 30 bits long as shown in Figure 5.3. The example substring shown in Figure 5.3 encoded a step-attenuator setting of 4.5 dB and a phase-shifter setting of 151°.

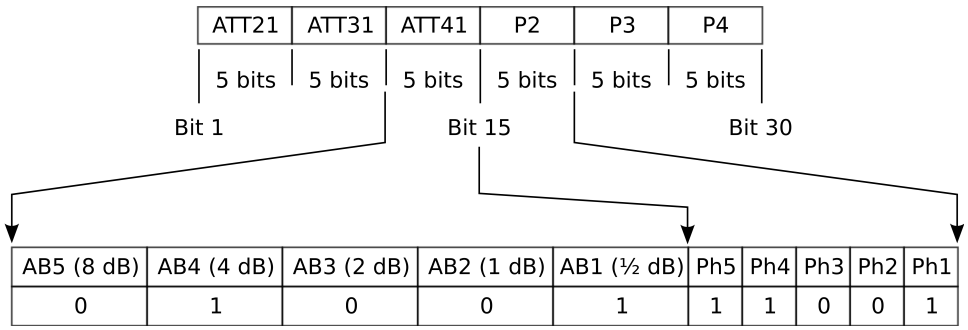
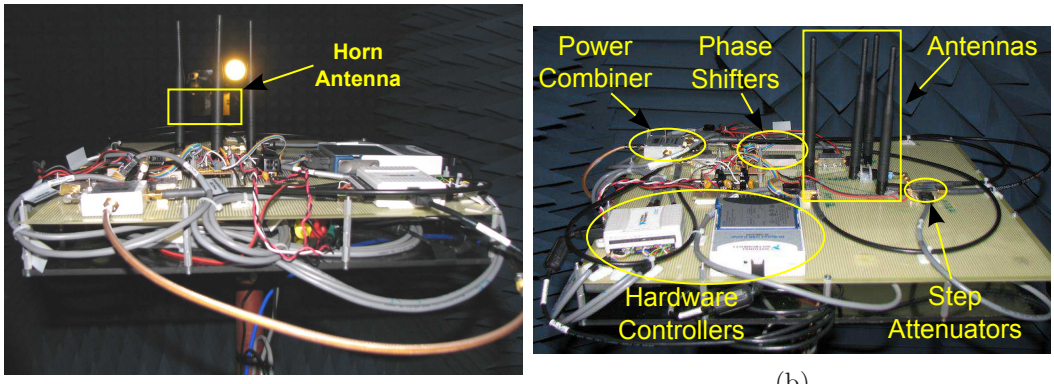


Figure 5.3: Bitwise string encodings of array phase shifters and step-attenuators with a sample encoding.

Our proof-of-concept anti-jamming beamforming system is shown in Figure 5.4 with the major components identified. The array hardware including phase shifters and step-attenuators is mounted on a 16" × 18" FR4 perforated board and mounted the assembly onto a 1/4" thick Plexiglas plate. The Plexiglas plate is mounted into a Delrin pipe that is connected to a Velmex stepper-motor controlled turntable (not shown). The antenna array layout shown in Figure 5.4b is translated into a CAD model and served as the basis of the FEM and MOM models described in Section 5.3.

The test cases used to evaluate the algorithms are listed in Table 5.1. Stepped mobile means that jammer DOAs remains constant for M_{St} generations (every $P \times M_{\mathrm{St}}$ SA / HCA evaluations), switched to another set of DOAs every $M_{\mathrm{St}} + 1$ generation



(a) (b)

Figure 5.4: Photographs of 2.4 GHz antenna array mounted inside CMU's anechoic chamber: (a) Showing chamber horn antenna, (b) Beamforming array with major components identified.

Table 5.1: List of anti-jamming scenarios explored by experiments (Exp) and simulations (Sim).

	Static		**Stepped Mobile**	**Continuous Mobile**
Algorithm	**2 jammer**	**3 Jammer**	**2 Jammer**	**2 Jammer**
SGA	Exp & Sim	Exp & Sim	Sim	Sim
TDGA	Exp & Sim	Sim	Exp & Sim	Sim
SA	Exp & Sim	Exp & Sim	Sim	Sim
HCA	Sim	Sim	Sim	Sim

($P \times M_{\mathrm{St}} + 1$ evaluation) and remains constant for the next M_{St} generations ($P \times M_{\mathrm{St}}$ evaluations). Continuous mobile signals changed their DOAs every generation (evaluation). The SGA and TDGA hardware experiments expand upon the results discussed in [77–80]. Hardware experiments and simulations are conducted with both simulated annealing and the hill-climbing algorithm to compare all four algorithms' performances in terms of converges SINR fitness values and convergence times. The

Table 5.2: Parameters used in SGA and TDGA *in-situ* experiments and simulations.

Parameter	**Value(s)**
Probability of Crossover P_c	0.6
Probability of Mutation, P_m	0.02 gen 0–5; 0.01 after
Number of Elite Strings, n_e	4
Population Size, M	200
Number of Generations (Static Signals)	51
Number of Generations (Stepped Mobile Signals)	61
Stepped Mobile Signal Switching Period in Generations	10
Number of Generations (Continuous Mobile Signals)	51 & 101

SGA and TDGA hardware experiment results are used to evaluate simulation fidelity and to show that simulations are comparable to hardware experiments. We performed simulations only on all HCA test cases, all continuous mobile cases, and SGA and SA stepped mobile cases.

Table 5.3: Parameters used in simulated annealing *in-situ* experiments and simulations.

Parameter	**Value(s)**
Initial Probability of Mutation, P_i	0.55
Final Probability of Mutation, P_f	9.5×10^{-4}
Metropolis Condition Temperature Scaling Factor, T_{Met}	10
Total Evaluations (Stationary Signals), ν_{Tot}	10200
Total Evaluations (Stepped Mobile Signals), ν_{Tot}	12200
Total Evaluations (Continuous Mobile Signals), ν_{Tot}	10200 & 20200

We list our SGA and TDGA parameters (common to both hardware experiments and simulations) in Table 5.2. These are the same settings used in [34, 77–80]. The SA hardware experiment and simulation parameters are listed in Table 5.3. The temperature schedule cooling rate τ and schedule offset T_{off} are calculated using equation (3.27) with $\mathrm{P_i} = \mathrm{P_{Mut,\,Mod}}\,(\nu = 1)$ and $\mathrm{P_f} = \mathrm{P_{Mut,\,Mod}}\,(\nu = \nu_{\text{Tot}})$ as its inputs. The HCA experiments used the same SA settings except the Metropolis Condition temperature scaling factor, T_{Met}, is set to 1×10^{-18} to turn off the Metropolis Condition.

5.2 SINR Fitness Landscape

The anti-jamming beamforming problem is a combinatorial search problem because the array's step-attenuators used binary control settings, and its analog phase shifter settings are digitized to five bits (with 11.6° per bit). With a 30-bit long hardware setting string, the total number of possible combinations is 2^{30}. Given that the time required to set the three step-attenuators and three phase shifters is roughly 100 milliseconds, a brute force search of the entire parameter space would take more than 3.4 years. Because signals can change directions in seconds, this is clearly unfeasible.

To plot the fitness landscape, the 30-bit string shown in Figure 5.3 is divided into two parts: The first 15-bits with encoded settings for the three step-attenuators, and the last 15-bits with encoded settings for the phase shifters. Each binary sub-string is converted into a decimal number with the resulting step-attenuator number plotted on the x-axis, the phase-shifter number plotted on the y-axis, and the SINR fitness

plotted on the z-axis.

An example SINR (dB) fitness landscape for 30 independent trails of an SGA with a population of 200 strings is shown in Figure 5.5. This figure indicates that the landscape is highly multimodal with multiple peaks and values. This would lead classical optimization algorithms such as LMS towards getting stuck in local maxima. Even if LMS is perturbed with noise, it will move from one local maxima due to another. GAs are less likely to get stuck in local maxima [24]. Examples of other fitness landscapes explored by the SGA and TDGA can be found in [34, 79].

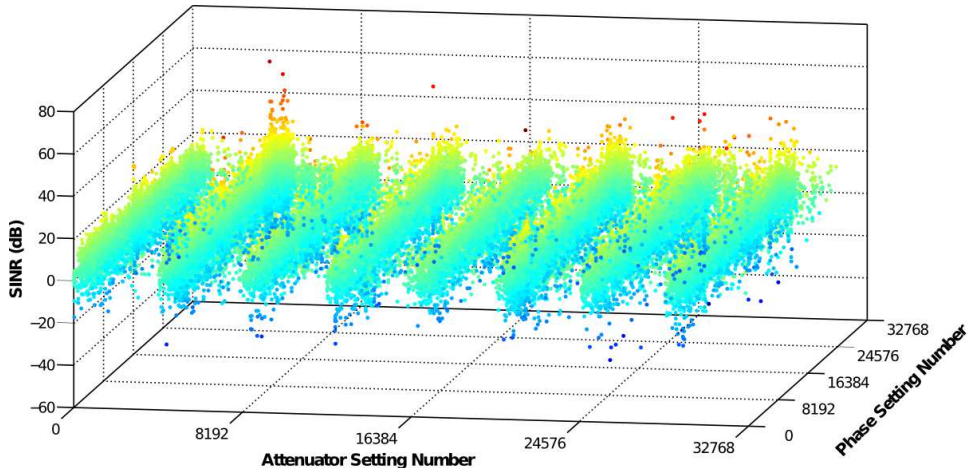


Figure 5.5: Example SINR (dB) fitness landscape for 30 independent trials of an SGA with 200 strings population when SOI at 0° and two jammers at 45° and 200°.

5.3 FEM and MOM Models of Antenna Array

To account for mutual coupling between antennas and reflections off array hardware, electromagnetic models of the proof-of-concept beamforming array are developed in WIPL-D (MOM) and Ansys HFSS v15 (FEM).The models developed are shown in

Figures 5.6 – 5.8. The HFSS model (see Figure 5.6) includes the array hardware components, coax cables, component mounting board, 0.25" Plexiglas turntable mounting plate, and metal standoffs. There are three WIPL-D models developed: antennas only (Figure 5.7), antennas with hardware components (Figure 5.8), and antennas with hardware components and metal standoffs (Figure 5.9).

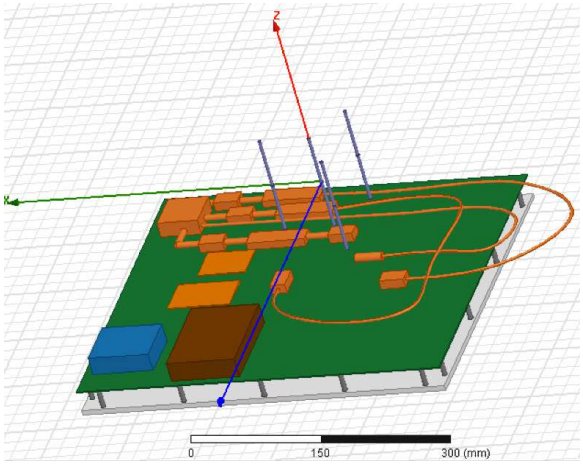


Figure 5.6: HFSS FEM model of the 2.4 GHz beamforming array with antennas, hardware components, mounting boards, and coax cables.

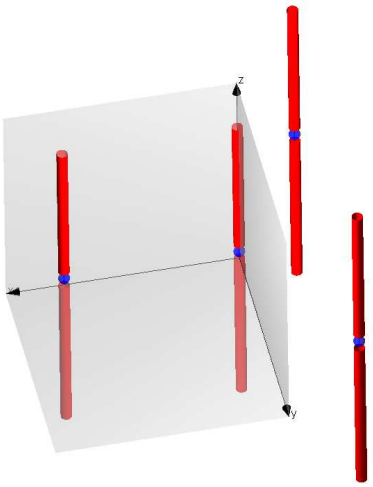


Figure 5.7: WIPL-D MOM model of the 2.4 GHz beamforming array with antennas only.

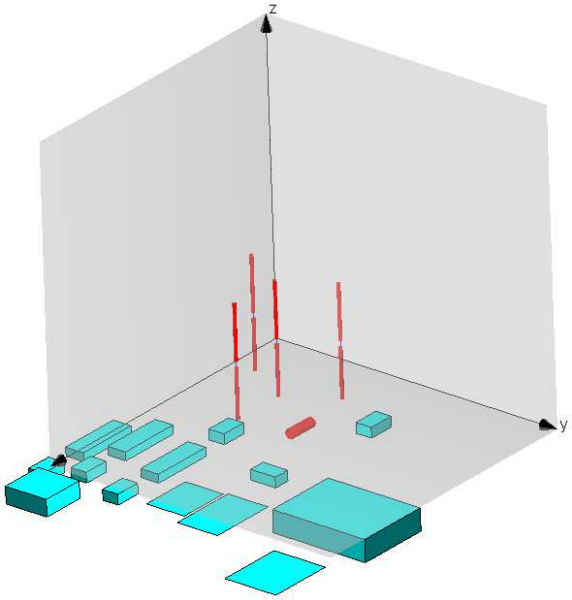


Figure 5.8: WIPL-D MOM model of the 2.4 GHz beamforming array with antennas and hardware components.

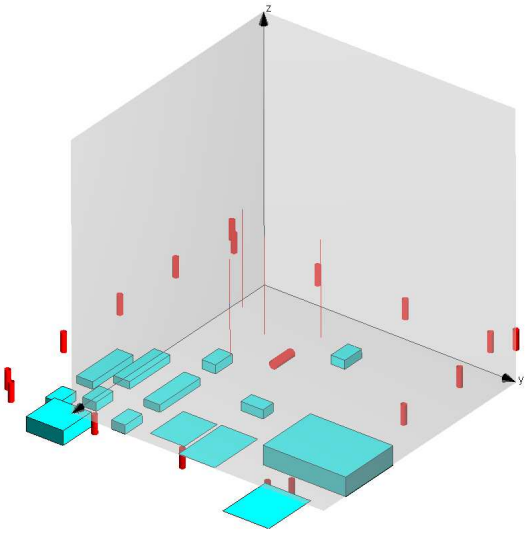


Figure 5.9: WIPL-D MOM model of the 2.4 GHz beamforming array with antennas, hardware components, and standoffs.

The three WIPL-D models have 44 unknowns, 432 unknowns, and 450 unknowns with 0.23 second calculation times. The HFSS model shown in Figure 5.6 has 218785

total tetrahedra and reaches convergence with 18.6 minutes simulation time.

These MOM models of the four-antenna array are integrated with Matlab simulations using the AntNet software provides by Dr. Derek Linden. AntNet is an add-on program to WIPL-D that calculates N-port antenna array far-field patterns by using the $N \times N$ impedance matrix $\mathbf{Z}$, the diagonal $N \times N$ port impedance ($\mathbf{Z}_{\mathrm{o}}$) matrix, and the theta ($\mathrm{E}_{\theta}^{\mathrm{i}}(\theta,\phi)$) and phi ($\mathrm{E}_{\phi}^{\mathrm{i}}(\theta,\phi)$) radiation pattern components for each antenna $i \in [1, N]$ from a single run of WIPL-D. Using a N-vector complex voltage source V_S, AntNet uses network theory and superposition to calculate the beamformed electric fields per [72] as

$$\mathbf{V} \;=\; \mathbf{Z}\left(\mathbf{Z} \;+\; \mathbf{Z}_{\mathrm{o}}\right)^{-1}\mathbf{V}_S \tag{5.1}$$

$$E_{\Psi}(\theta,\phi) \;=\; \sum_{i=1}^{N} V_{\mathrm{i}} E_{\Psi}^{\mathrm{i}}(\theta,\phi)\,,\;\; \Psi \in \{\theta,\phi\} \tag{5.2}$$

The two *in-situ* cases are compared with WIPL-D and HFSS models to evaluate the models' fidelity as shown in Figures 5.10 – 5.11. Both cases stem from solutions found by the stochastic algorithms *in-situ* and located inside an anechoic chamber. The first case is when the SOI is at 0°, and the jammers are at 45° and 200°. The second case is when the SOI remains at 0°, and the jammers are at 120° and 300°. It can be seen that the WIPL-D models with antennas only, antennas plus hardware, and antennas plus hardware and mounting bracket standoffs are very similar. They also agree very well with the *in-situ* measurements shown in Figure 5.10. The HFSS model in Figure 5.10 also agrees well with the *in-situ* azimuth pattern although the

HFSS pattern varies more across the azimuth angles due to the presence of dielectric mounting plates.

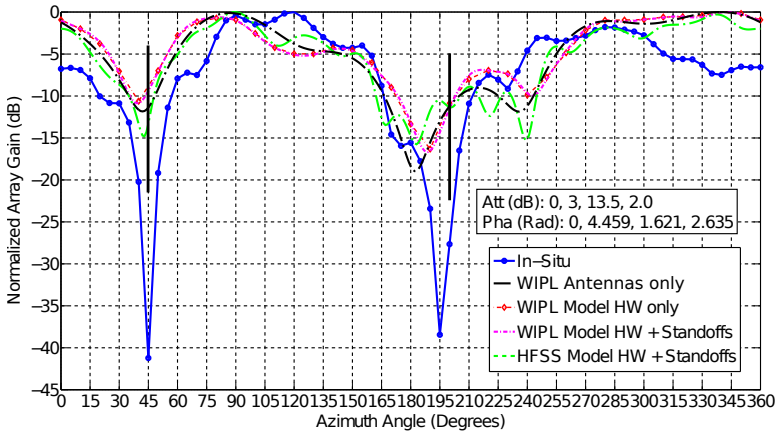


Figure 5.10: Comparison of *in-situ* measurements with WIPL-D and HFSS models of a four-antenna array when the SOI is at 0° and the jammers are at 45° and 200°.

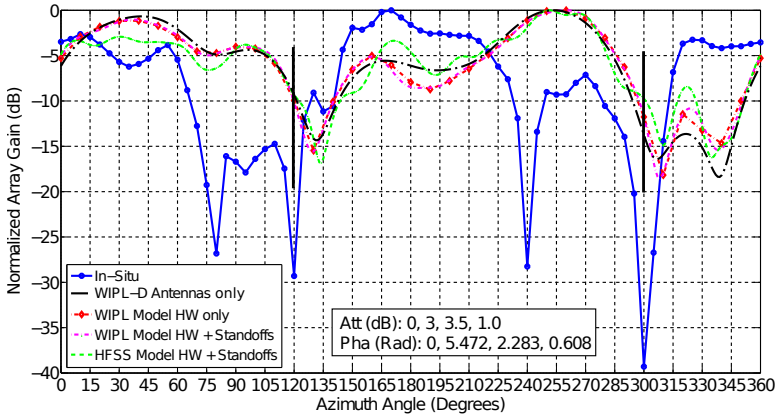


Figure 5.11: Comparison of *in-situ* measurements with WIPL-D and HFSS models of a four-antenna array when the SOI is at 0° and the jammers are at 120° and 300°.

Although the WIPL-D and HFSS models do not capture all of the nulls in Figure 5.11, the null depths at 120° and 300° are conservative compared to *in-situ* measurements, and simulations with the stochastic algorithms predict lower than expect SINR fitness. The WIPL-D and HFSS models track the *in-situ* azimuth pattern in Figure 5.10 with conservative nulls, and the azimuth patterns for three WIPL-D models dif-

fer little compared to the HFSS azimuth pattern. Thus, the WIPL-D simulations have reasonably good fidelity to justify evaluation of results obtained by algorithm simulations.

Even though the HFSS model includes both dielectric mounting plates and coax cables as seen in Figure 5.4, its resulting azimuth pattern differs little from the WIPL-D patterns. Although it is possible that control cables and hardware non-linearities not included in the model created the additional nulls in Figure 5.10, the fidelity of the current models are high enough that further developing the models is deemed unnecessary.

5.4 Algorithm Performance With Static Signals

This section discusses *in-situ* measurements of the SGA, TDGA, and SA in thwarting jammers when all of the signals are static. As shown in Table 5.1, algorithm performance is compared in maximizing SINR with two jammers and with three jammers. Because the three jammer case is invested first during the research, it is discussed in this section first.

5.4.1 One SOI and Three Interfering Signals

Our first set of experiments begins with an investigation into if and how well the SGA can adapt to maximize SINR with one SOI and three jammers via simulations of the array operating in a Rayleigh fading environment [34]. The array is modeled using

an array factor equation that did not account for mutual coupling between antennas, so the next step is to investigate the SGA's performance inside an anechoic chamber. *In-situ* experiments begins with the single SOI and three jammers case as described in [77].

A test case with the SOI at 60° and the three jammers at 105°, 245°, and 320° is discussed. The results with a population size of 200 strings is shown in Figure 5.12 and with a population of 400 strings in Figure 5.13. The shallowest null depth in both cases is 12.1 dB down from the SOI. This corresponds to a SINR fitness of approximately 9 dB. This indicates that increasing the population size from 200 strings to 400 strings has no effect on SGA performance. The number of signals equaled the number of antennas in the array, so this makes the beamforming problem more difficult for the SGA.

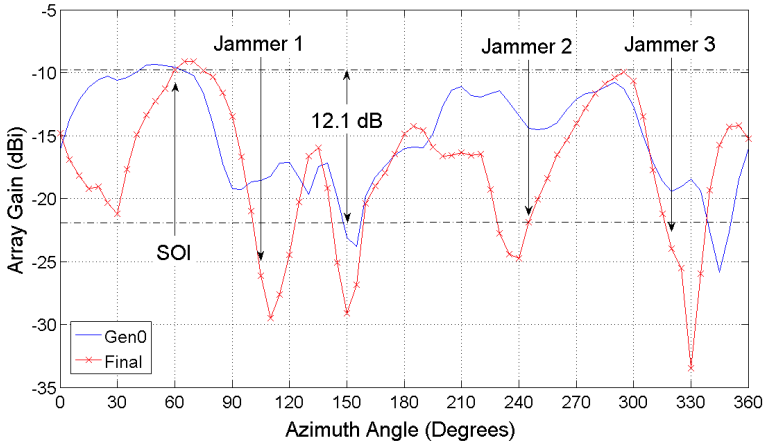


Figure 5.12: *In-situ* measurements of SGA optimized azimuth radiation pattern with a single SOI at 60° and three jammers at 105°, 245°, and 320°. The SGA has a population size of 200 strings

A second test case with the SOI at 0° and three jammers at 45°, 200°, and 300° is

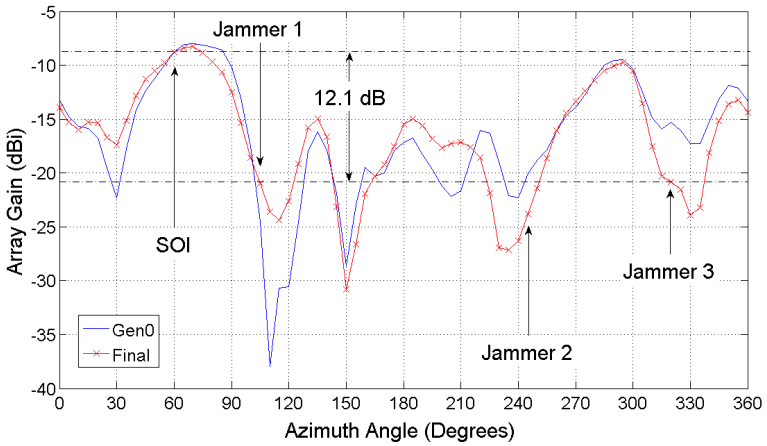


Figure 5.13: *In-situ* measurements of SGA optimized azimuth radiation pattern with a single SOI at 60° and three jammers at 105°, 45°, and 320°. The SGA has a population size of 400 strings

evaluated to verify that the SGA handles other signal direction sets. A best case performance graph (sampled from several independent runs) is shown in Figure 5.14. It exhibits SGA search progress where the SGA performed useful search within the first 15 generations and converges to its final value by generation 20. The maximum SINR at convergence is 15.8 dB which is several decibels higher than what is measured for the one SOI and three jammer case in [77]. The SGA in this experimental run has a population size of 200 strings.

The corresponding initial and converged azimuth radiation plots are shown in Figure 5.15. Although the SOI lost 1 dB of array gain, the SGA improves all three jammers' null depths significantly with the worst case final null depth 17.2 dB down from the SOI. Another means of evaluating the GA's convergence time as is to calculate the population's mean and standard-deviation Hamming distance as noted by [81]. The mean and standard-deviation for the second single SOI and three jammer

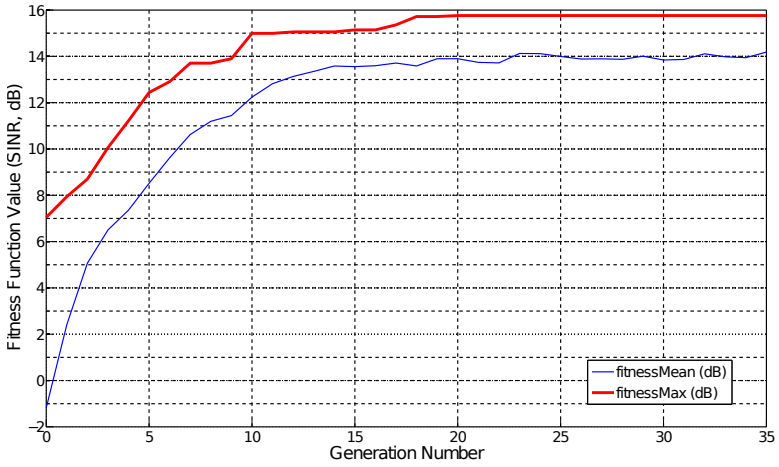


Figure 5.14: *In-situ* best-case learning curve of SGA with a single SOI at 0° and three jammers at 45°, 200°, and 300°. The SGA has a population size of 200 strings.

case described above is shown in Figure 5.15. It can be seen that the SGA converges by generation 15 since the mean Hamming distance dropped to 0.05, and the standard-deviation leveled out to approximately 0.035.

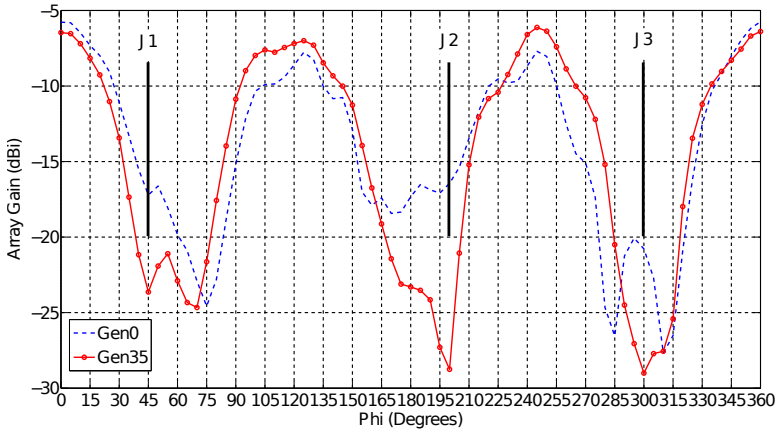


Figure 5.15: *In-situ* best-case SGA optimized azimuth radiation pattern with a single SOI at 0° and three jammers at 45°, 200°, and 300°. The SGA has a population size of 200 strings

Hamming distance can also be used to measure how much genetic information that the GA discarded as it converges to a final solution. If the initial population

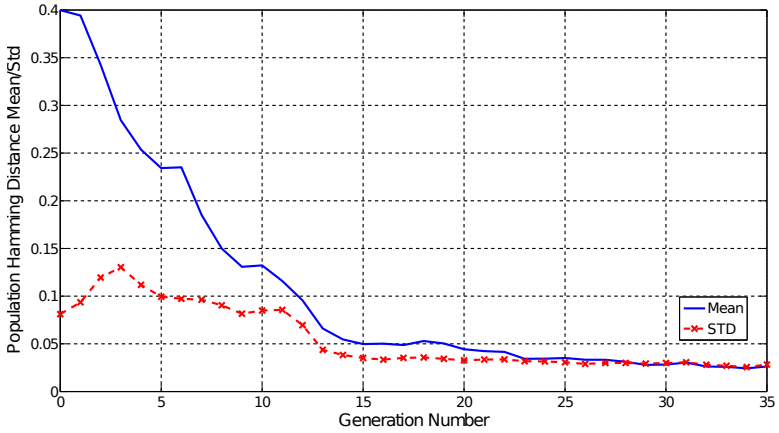


Figure 5.16: *In-situ* best-case Hamming distance curve of SGA with a single SOI at 0° and three jammers at 45°, 200°, and 300°. The SGA has a population size of 200 strings

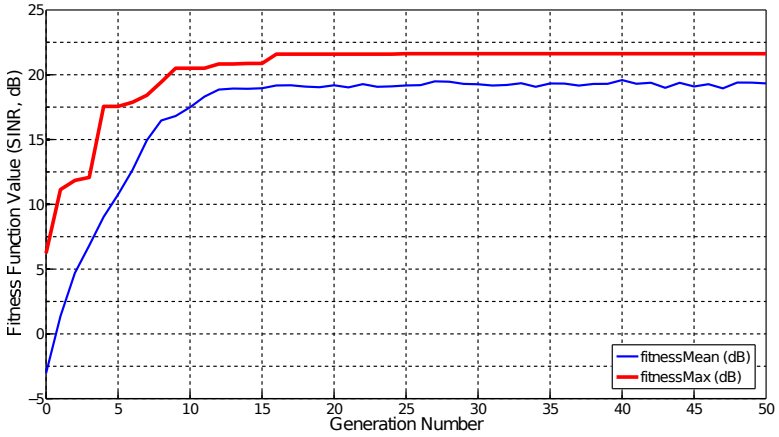


Figure 5.17: AntNet WIPL-D (third model) best-case learning curve of SGA with a single SOI at 0° and three jammers at 45°, 200°, and 300°. The SGA has a population size of 200 strings

is chosen purely at random, the mean Hamming distance as a percentage is 0.5.[1] The mating selection process causes multiple copies of the same string to occupy the population as the GA converges causing information contained by deceased strings to be lost. As is shown in Section 5.5, the mutation operator causes the mean and

[1]In Figure 5.16, the initial mean Hamming distance is 0.4 because the two MSBs (4 dB and 8 dB settings) for the three step-attenuators are set to logic 0 at initialization. Six out of thirty bits for every string are non-random corresponding to an 80% initially random population.

standard-deviation Hamming distance to remain small but non-zero during convergence. Convergence is also non-final in an AWGN channel due to the mutation operator's reintroduction of information into the population.

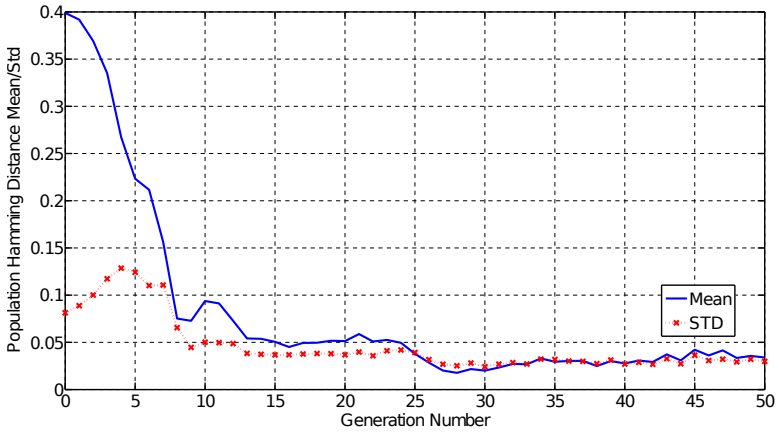


Figure 5.18: AntNet WIPL-D (third model) best-case Hamming distance curve of SGA with a single SOI at 0° and three jammers at 45°, 200°, and 300°. The SGA has a population size of 200 strings

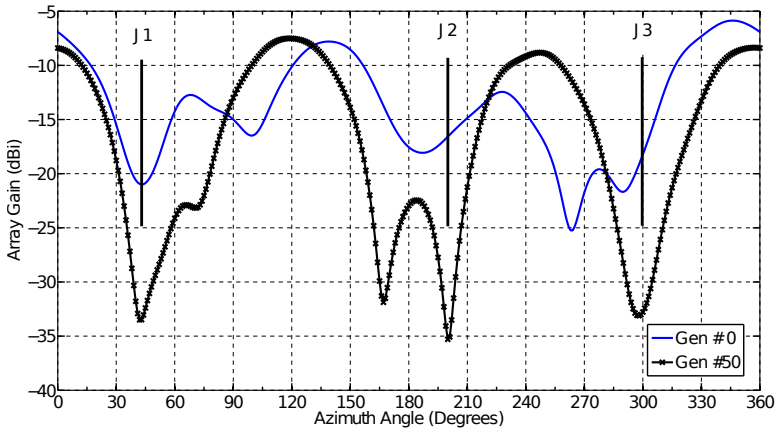


Figure 5.19: AntNet WIPL-D (third model) best-case SGA optimized azimuth radiation pattern with a single SOI at 0° and three jammers at 45°, 200°, and 300°. The SGA has a population size of 200 strings

Because the 95% confidence intervals for *in-situ* measurements using Gaussian and Student-t distributions are numerically close (see [79]), less than thirty indepen-

dent runs of the SOI at 0° and jammers at {45°, 200°, 300°} are collected *in-situ*. Because it is shown that the WIPL-D model produced azimuth patterns agreed well with *in-situ* measurements in Section 5.3, 30 independent simulations of the SGA for the same signal set are run since simulation data can be generated quickly. The best case performance graph, Hamming distance plots, and azimuth radiation patterns are shown in Figures 5.17 – 5.19. Performance graphs with statistical comparisons are plotted in Figure 5.20 based on the best performing solutions from each independent trial.

The performance curves shown in Figure 5.17 behave in the same manner as the *in-situ* performance curves of Figure 5.15. A similar pattern is observed in the Hamming distance plots of Figures 5.16 and 5.18. Like the *in-situ* run, the simulated SGA performed useful search in the first 15 generations and converges to a final solution by generation 20.

Because the SGA performed its optimization search on a population, it is useful to create a "best of the best" statistical performance graph by plotting several curves versus generation number over the 30 independent runs. In Figure 5.20, the maximum of each trial's best solution is plotted along with the average of the best and minimum of the best. The confidence intervals are also calculated. The shaded region surrounding the average of the best represents the 95% confidence intervals using the Student-t distribution, and the error bars represent the 95% confidence intervals assuming Gaussian distribution. The Student-t confidence interval is indistinguishable

from the Gaussian confidence interval which is consistent considering the number of samples used in the calculation.

The maximum of the best and minimum of the best curves along with the confidence intervals places bounds on the expected SINR fitness. The best overall solution has a 21.6 dB SINR while the worst of the best solutions has 16.1 dB SINR. This represents a 5.5 dB total spread between the set of best solutions finds in 30 independent trials. If a good solution is chosen at random from the converges solution space, 95% of solutions chosen are expected to be found within a confidence interval C around the average of the best. The 99% confidence intervals (not shown) are also calculated, and it is observed that those confidence intervals occupied more of the region between the maximum of and minimum of the best curves.

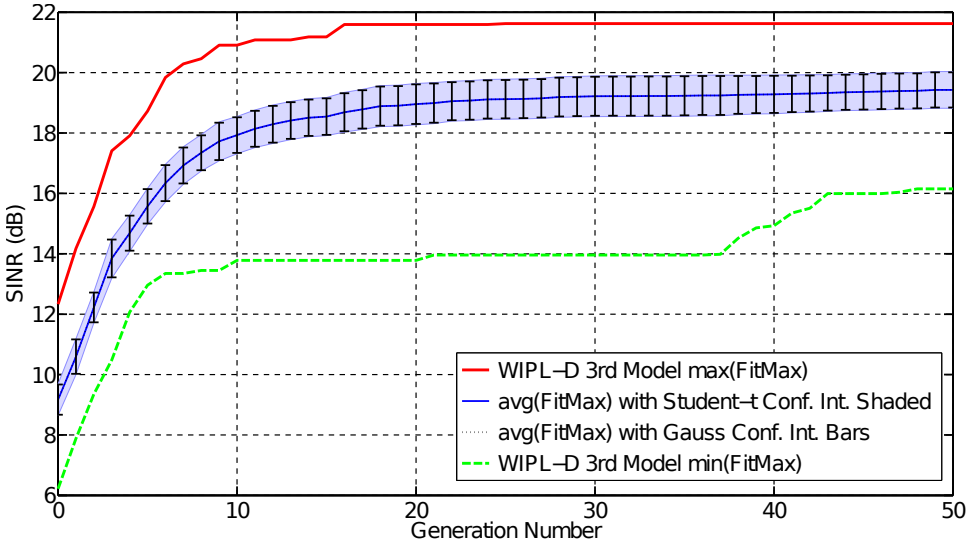


Figure 5.20: SGA simulated performance curves collected over 30 independent runs using AntNet WIPL-D (third model) with SOI at 0° and three jammers at 45°, 200°, and 300°. The SGA has a population size of 200 strings

Thirty independent trials are performed where the TDGA optimizes the antenna

array to focus EM energy on the SOI at 0° and to minimize energy towards the jammers at 45°, 200°, and 300°. The statistical performance curves and the initial and final radiation patterns for the best solution are plotted in Figures 5.21 and 5.22. Compared to the SGA, the TDGA converges roughly in the same number of generations to the same maximum SINR value. However, the minimum of the best solution found by the TDGA has a 17.6 dB SINR compared to the SGA's minimum of the best 16.1 dB SINR solution. The spread of the set of the best converges solutions for the TDGA is 4.1 dB: a one dB improvement compared to the SGA. The TDGA's Student-t and Gaussian 95% confidence intervals are indistinguishable and represent smaller bounds than the SGA's 95% confidence intervals. The converged best-case TDGA optimized azimuth radiation pattern shown in Figure 5.22 is similar to the best-case SGA optimized pattern plotted in Figure 5.19.

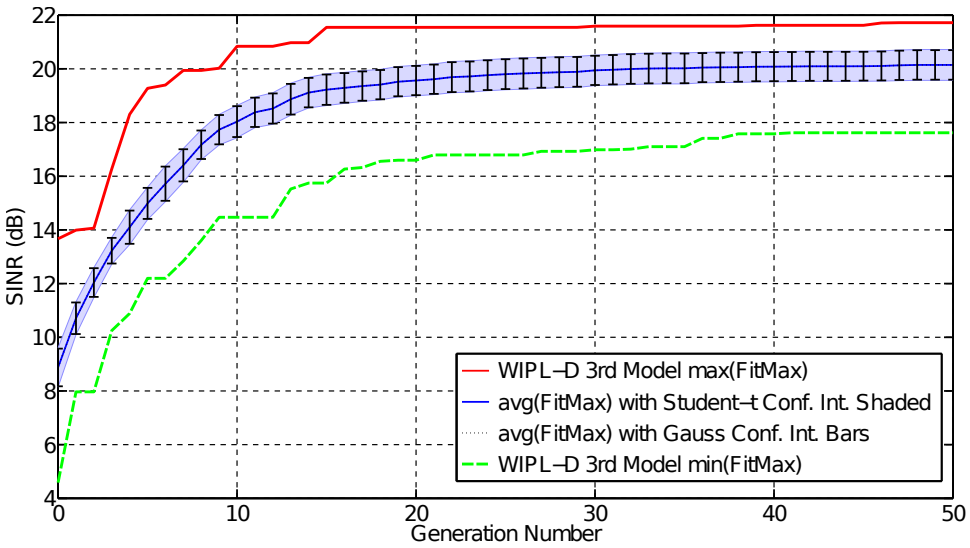


Figure 5.21: TDGA simulated performance curves collected over 30 independent runs using AntNet WIPL-D (third model) with SOI at 0° and three jammers at 45°, 200°, and 300°. The TDGA has a population size of 200 strings

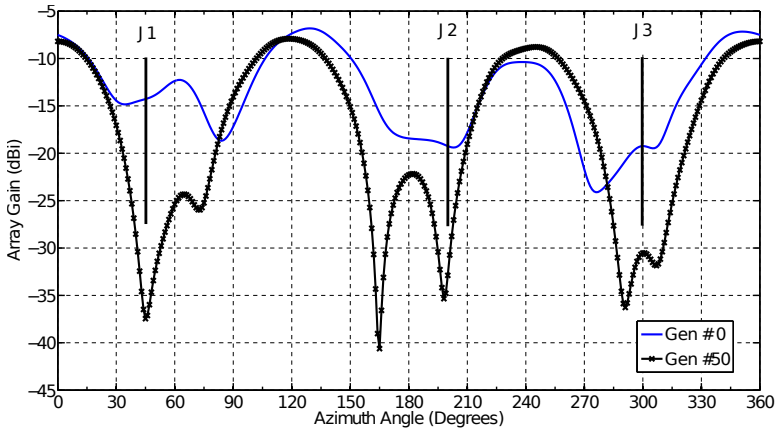


Figure 5.22: AntNet WIPL-D (third model) best-case TDGA optimized azimuth radiation pattern with a single SOI at 0° and three jammers at 45°, 200°, and 300°. The TDGA has a population size of 200 strings.

The remainder of this subsection analyzes the Simulated Annealing and Hill Climbing Algorithms in maximizing SINR when three static jammers are present. The HCA is SA with the Metropolis condition turned off such that only better solutions (compared to the most recent ones) are kept.

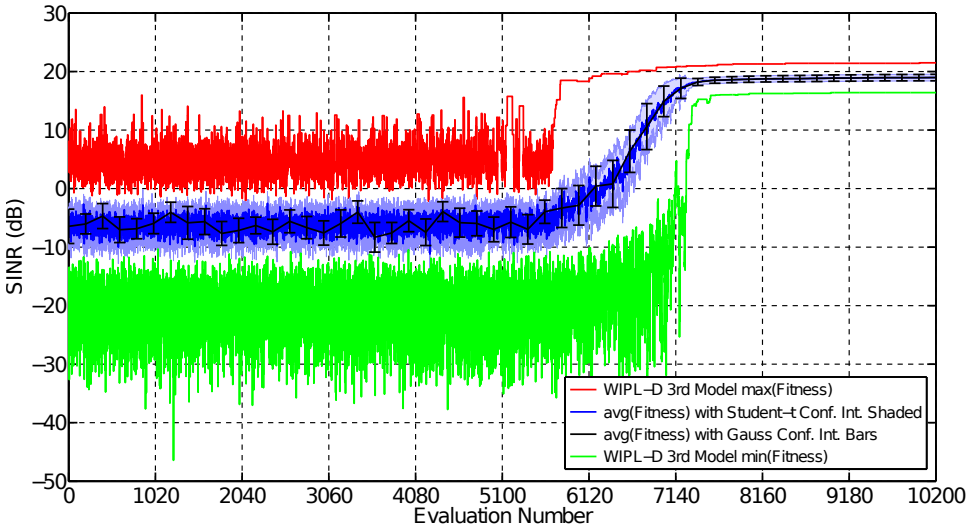


Figure 5.23: SA simulated performance curves collected over 30 independent runs using AntNet WIPL-D (third model) with SOI at 0° and three jammers at 45°, 200°, and 300°.

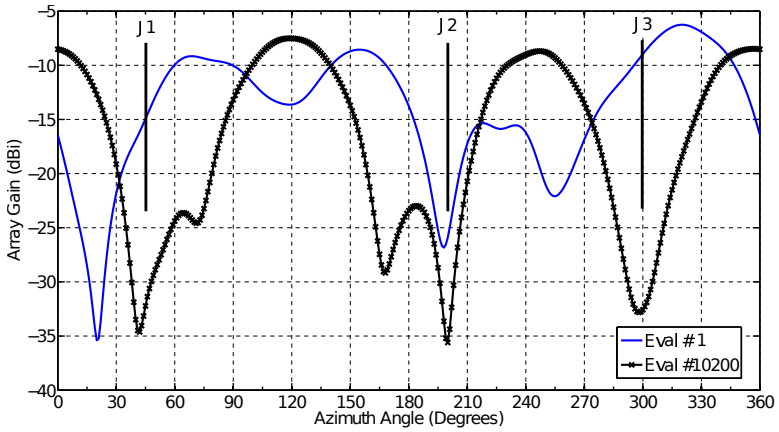


Figure 5.24: AntNet WIPL-D (third model) best-case SA optimized azimuth radiation pattern with a single SOI at 0° and three jammers at 45°, 200°, and 300°.

The performance curves for 30 independent simulated runs of the SA algorithm along with the best-case azimuth plots are shown in Figures 5.23 and 5.24. For consistency with the SGA and TDGA experiments, we run SA for 10,200 evaluations where every 200 evaluations represent one GA generation of 200 strings, and an evaluation represents the set of calculations necessary to calculate one solution's SINR value. Whereas the SGA and TDGA perform most of their useful search on average within ten generations for the three jammer case, SA requires on average 7,140 evaluations. SA is thus slower than the GA by a factor of approximately 3.5 since this represents an equivalent of almost 36 generations. Using the sigmoidal temperature schedule defined by Table 5.3, SA explores the search space for over 5,100 evaluations before it begins its exploitation stage.

When SA converges, the best solution has 21.5 dB SINR, and the worst solution has 16.4 dB SINR. This represents a 5.1 dB spread which is the same compared to the set of best solutions found by the SGA. The azimuth radiation patterns from the SA

trial that produced the best converges solution are shown in Figure 5.24. The azimuth plot from evaluation 10,200 is very similar to the best SGA and TDGA azimuth plots at convergence which is expected, as the three algorithms produce similar best-case results at convergence.

After performing SGA and TDGA experiments *in-situ* with the four-antenna array in an anechoic chamber, SA's performance is investigated in maximizing SINR first with two static jammers and then with three static jammers. Our results over thirty independent runs are shown in Figures 5.25 – 5.26.

Although the *in-situ* simulated annealing algorithm makes improvements, it did not perform as well compared to simulation results. The final best case azimuth plot shown in Figure 5.26 improves the SOI gain by a couple decibels and increases two null depths by 2 dB to 3 dB. Due to intermittent data collection issues, SA would have otherwise found better solutions, and this indicates a limitation to fault recovery with stochastic algorithms. As is shown in Chapter 6, stochastic algorithms can recover from hardware faults that occur within the RF paths between the antennas and the radio, but this assumes that the array controller is 100% reliable. Because the antenna array is in operation for several years prior to investigating SA, this assumption does not hold. This supports the discussion that although hardware fault recovery can keep the array operating longer under certain conditions, it does not negate the need to have an array repaired in the field after faults occur.

The performance curves for thirty independent trials of the HCA in simulation are

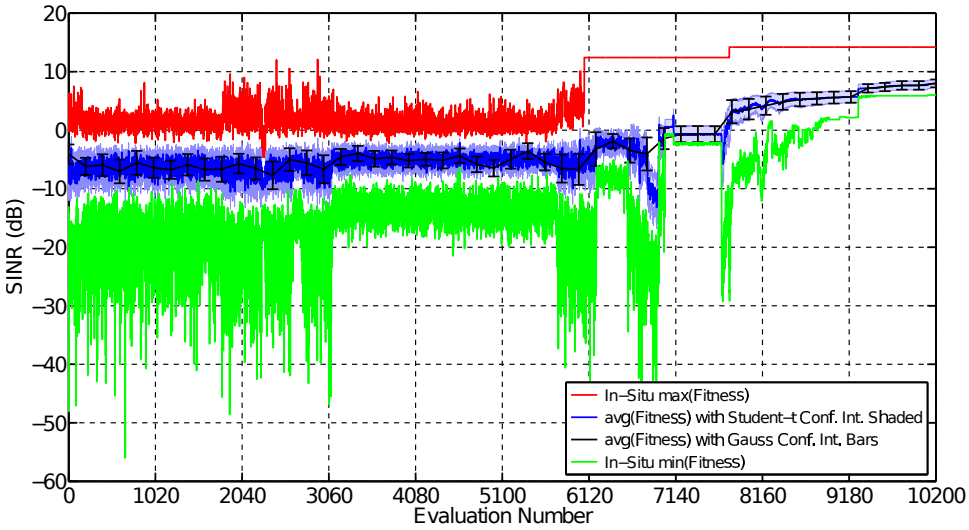


Figure 5.25: SA *in-situ* performance curves collected over 30 independent runs with SOI at 0° and three jammers at 45°, 200°, and 300°.

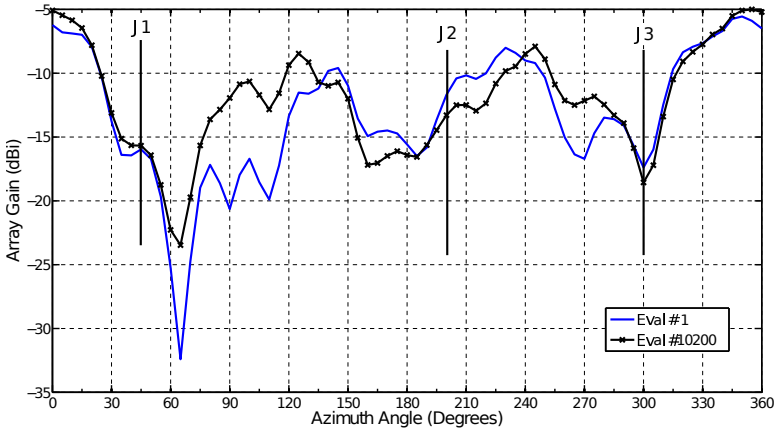


Figure 5.26: Best-case SA *in-situ* optimized azimuth radiation pattern with a single SOI at 0° and three jammers at 45°, 200°, and 300°.

shown in Figure 5.27, and the azimuth plots associated with the trial that contains the best solution are plotted in Figure 5.28. HCA is as slow in comparison to simulated annealing, as HCA converges by evaluation 7,140. However, HCA gets stuck at a local optima near evaluation 2,040 and stays there until evaluation 4,080 when it continues searching the parameter space. The maximum SINR value shown in the

performance graph at evaluation 10,200 is 21.33 dB, and the minimum SINR is 15.77 dB. This represents a 5.56 dB spread between the best and worst solutions that HCA finds. The best-case azimuth plot shown in Figure 5.28 is similar in comparison to the solutions found by SGA, TDGA, and SA.

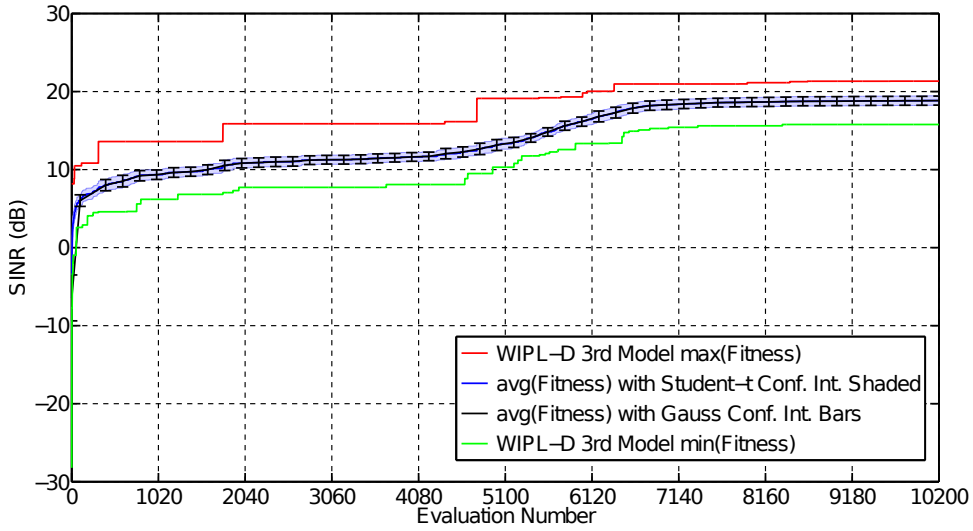


Figure 5.27: HCA simulated performance curves collected over 30 independent runs using AntNet WIPL-D (third model) with SOI at 0° and three jammers at 45°, 200°, and 300°.

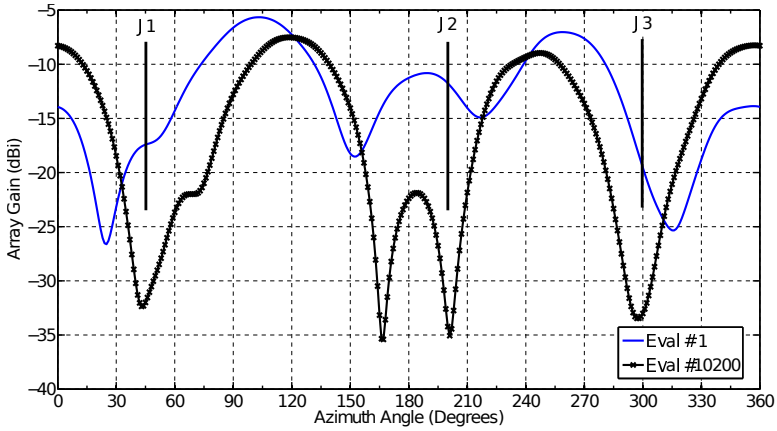


Figure 5.28: AntNet WIPL-D (third model) best-case HCA optimized azimuth radiation pattern with a single SOI at 0° and three jammers at 45°, 200°, and 300°.

The simulations show that the SGA and SA obtain higher converges solutions compared to *in-situ* results. Although it is shown that MOM based simulations have good fidelity when compared to *in-situ* measurements, there are other reasons why the simulations predict higher converges values. Aside from intermittent controller faults that interrupt the simulated annealing optimization process, there exist non-linearities within the RF path between the antennas and the RF power summer that cannot be modeled in simulation. Both WIPL-D and HFSS place the generators at the antennas whereas the hardware system has coax cables, phase shifters, step-attenuators, and adapters after the antennas. The phase-shifters are also non-linear devices whose impedance matches vary with their control voltages. Although the coax cables are the same length, phase variances in the phase shifters due to impedance mismatches affect the overall solution and cause the *in-situ* results to differ from simulations.

5.4.2 One SOI and Two Interfering Signals

Following [77], investigations are performed showing how well the SGA and TDGA performed in maximizing SINR when two static jammers are present, and better results are obtained at convergence. The double jammer case provides an extra degree-of-freedom which makes the problem easier to solve by the optimization algorithms. Simulations of the SGA, TDGA, SA, and HCA are run with 30 independent trials per algorithm.

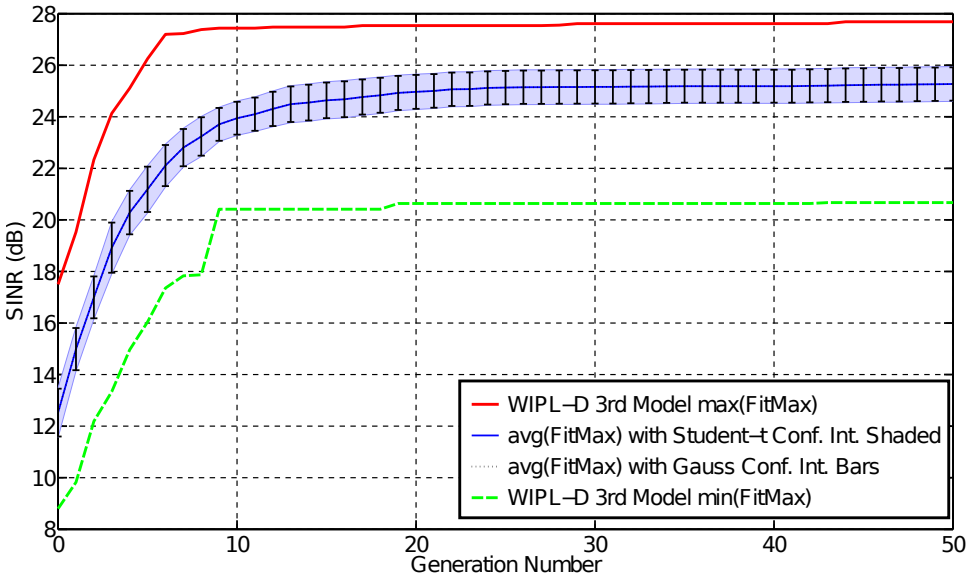


Figure 5.29: SGA simulated performance curves collected over 30 independent runs using AntNet WIPL-D (third model) with SOI at 0° and two jammers at 45° and 200°.

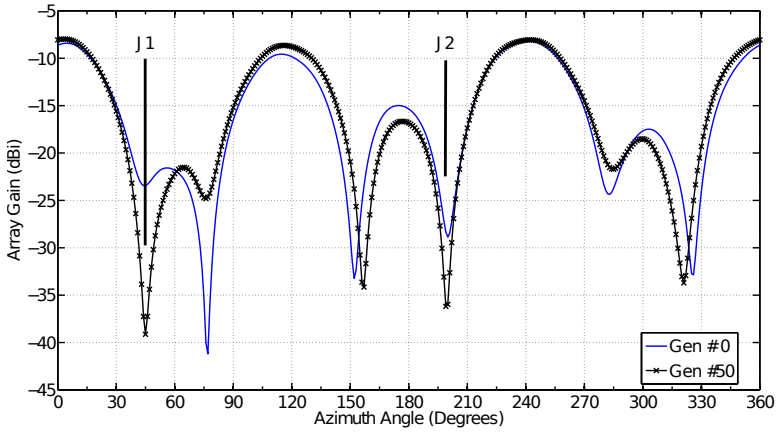


Figure 5.30: AntNet WIPL-D (third model) best-case SGA optimized azimuth radiation pattern with a single SOI at 0° and three jammers at 45° and 200°.

A performance graph for the SGA simulations with an SOI at 0° and two jammers at 45° and 200° is shown in Figure 5.29, and the best case optimized azimuth radiation pattern is shown in Figure 5.30. As expected, the best solutions that the SGA found for the two jammer case are significantly better than the three jammer case. The SGA

performed most of its useful search within the first 10 generations, and the SGA on average converges within 15 generations. The maximum of the best solutions found in the 30 trials has an SINR of 27.69 dB, and the minimum of the 30 best solutions found has a 20.67 dB SINR. The 95% confidence intervals are ±0.65 dB and ±0.67 dB for the Gaussian and Student-t distributions, respectively.

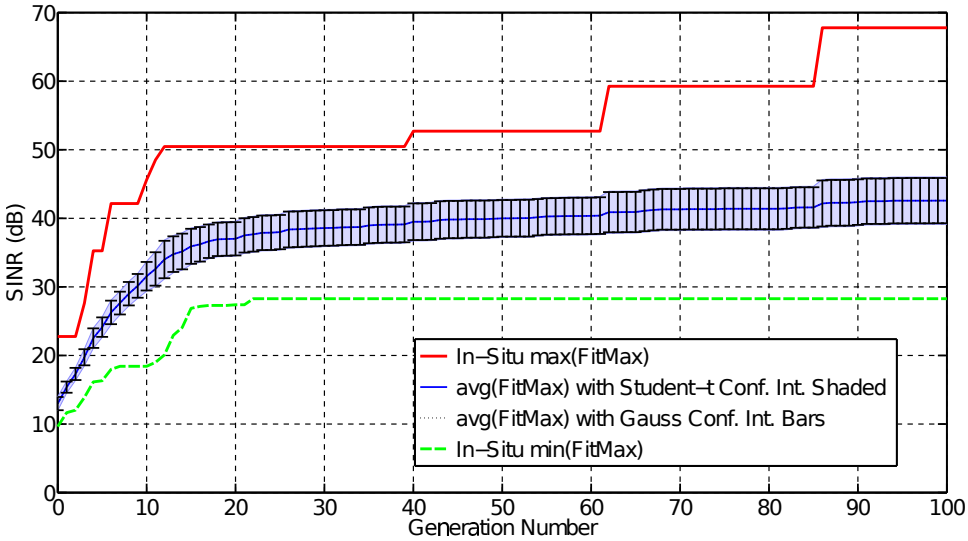


Figure 5.31: SGA *in-situ* performance curves collected over 30 independent runs with SOI at 0° and two jammers at 45° and 200°.

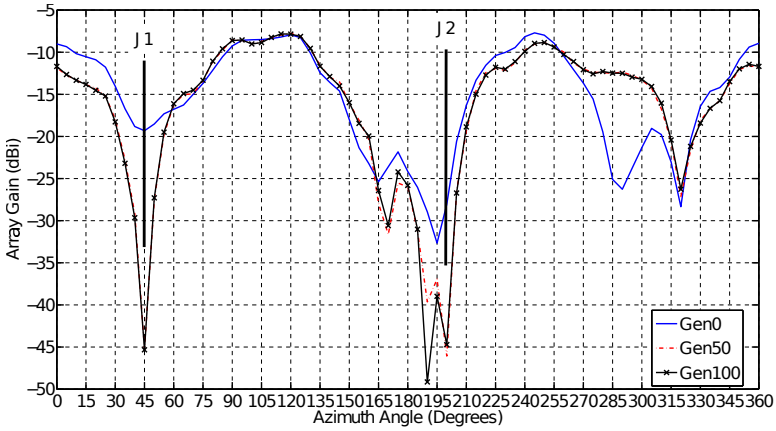


Figure 5.32: Best-case SGA *in-situ* optimized azimuth radiation pattern with a single SOI at 0° and two jammers at 45° and 200°.

Thirty independent runs of the SGA with 101 generations are run in the anechoic chamber. By generation 50, the best solution found by the SGA *in-situ* has a 52.7 dB SINR at generation 50, and the best-case SINR increases to 67.8 dB (see Figure 5.32). This is higher than predicted by the simulations. The 95% confidence interval around the mean of the best solutions shown in Figure 5.32 is ±2.67 dB at generation 50 and increases to ±3.30 dB by generation 100 assuming a Gaussian distribution. The confidence intervals using a Student-t distribution is ±2.79 dB at generation 50 and increases to ±3.45 dB by generation 100. The overall error between the two methods is a constant -4.17% overall all generations with the Student-t distribution predicting slightly smaller confidence intervals that the Gaussian distribution.

The azimuth plots for the best solution out of 30 runs are shown in Figure 5.32. The patterns plotted are consistent with the SINR values seen in the performance plots with the second jammer's null-depth increases by a few decibels. The best-case generation 50 and 100 *in-situ* radiation patterns are similar in shape compared to the radiation patterns predicted by the SGA simulations (see Figure 5.30).

The results from performing 30 independent simulations of the TDGA are shown in Figure 5.33. The overall performance is better compared to the SGA simulations where 30 independent runs of the TDGA produced a maximum 27.98 dB SINR, and a minimum (out of 30 best solutions) 22.49 dB SINR. The 95% confidence intervals at generation 50 are ±0.55 dB and ±0.57 dB for Gaussian and Student-t distributions. The best-case generation 0 and generation 50 azimuth radiation plots are shown in

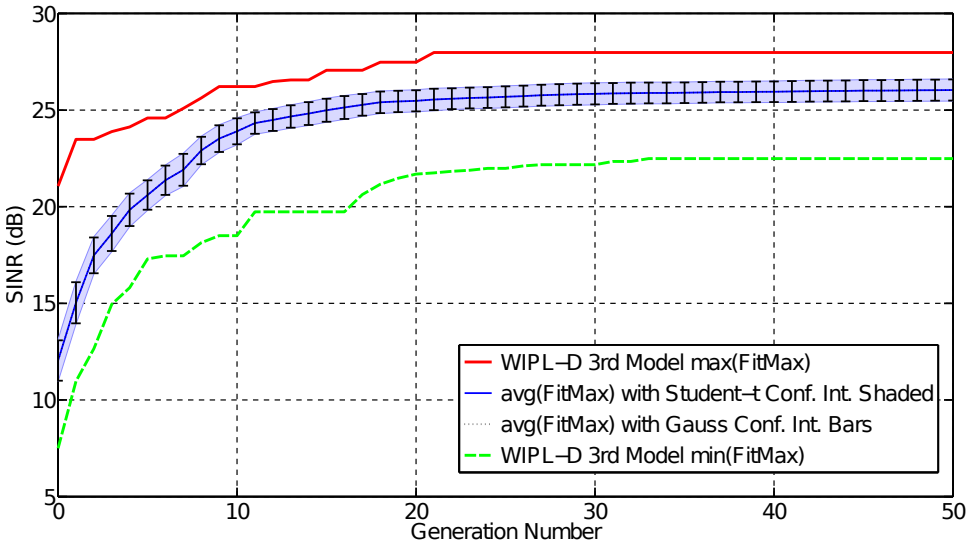


Figure 5.33: TDGA simulated performance curves collected over 30 independent runs using AntNet WIPL-D (third model) with SOI at 0° and two jammers at 45° and 200°.

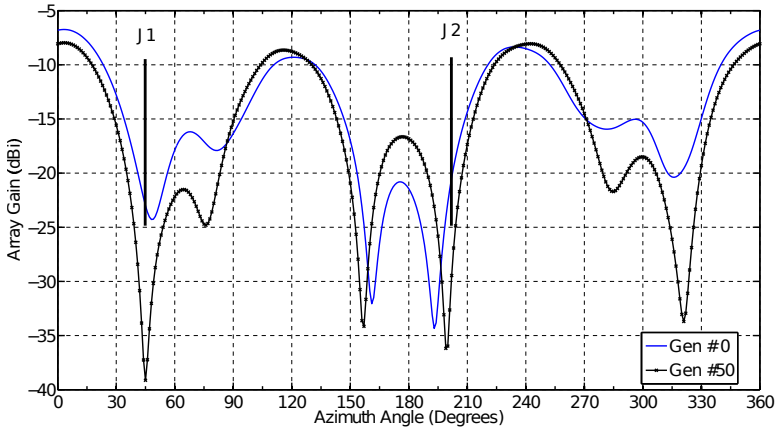


Figure 5.34: AntNet WIPL-D (third model) best-case TDGA optimized azimuth radiation pattern with a single SOI at 0° and three jammers at 45° and 200°.

Figure 5.36. The patterns produced by AntNet WIPL-D are consistent with the SINR level produced by the best solution that the TDGA found in this set of 30 independent runs, and the pattern is similar to the patterns produced in the best case by both SGA simulations and *in-situ* measurements.

We present the performance graphs for 30 independent *in-situ* runs of our TDGA

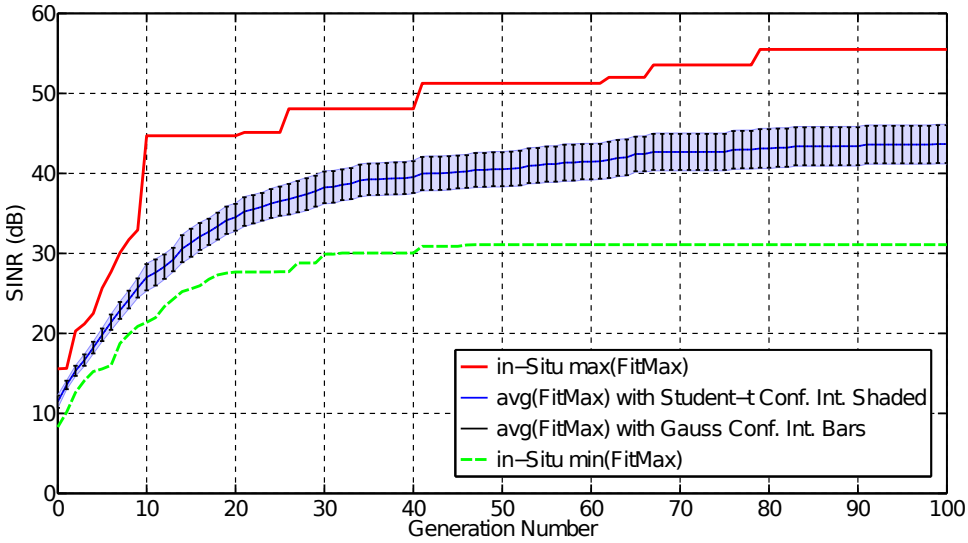


Figure 5.35: TDGA *in-situ* performance curves collected over 30 independent runs with SOI at 0° and two jammers at 45° and 200°.

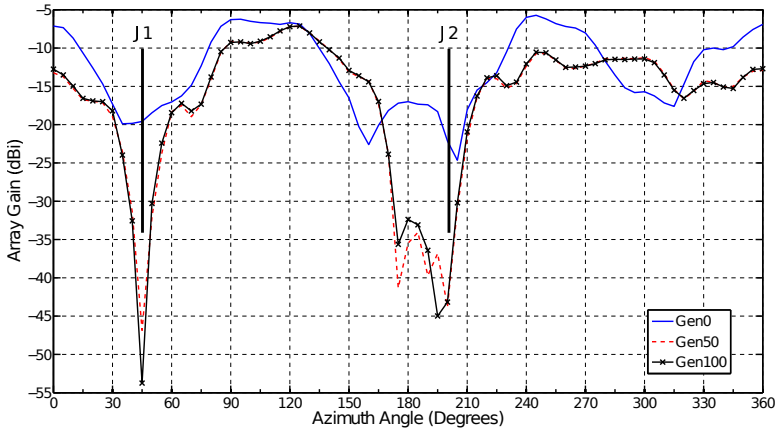


Figure 5.36: Best-case TDGA *in-situ* optimized azimuth radiation pattern with a single SOI at 0° and two jammers at 45° and 200°.

along with the best-case azimuth before / after optimization plots in Figures 5.37 and 5.38 with the algorithm run for 101 generations. The maximum SINR values at generation 50 and 100 are 51.26 dB and 55.50 dB. The 95% confidence intervals are ±2.12 dB and ±2.21 dB at generation 50 and ±2.47 dB and ±2.37 dB at generation 100 for the Gaussian and Student-t distributions respectively. The TDGA simulations

are more conservative in predicting the converges SINR values, and the best-case converges *in-situ* azimuth plots shown in Figure 5.36 are similar in shape to the AntNet WIPL-D model's azimuth plots. However, the 95% confidence intervals based on *in-situ* measurements are roughly 4 times larger than the confidence intervals predicted by the simulations.

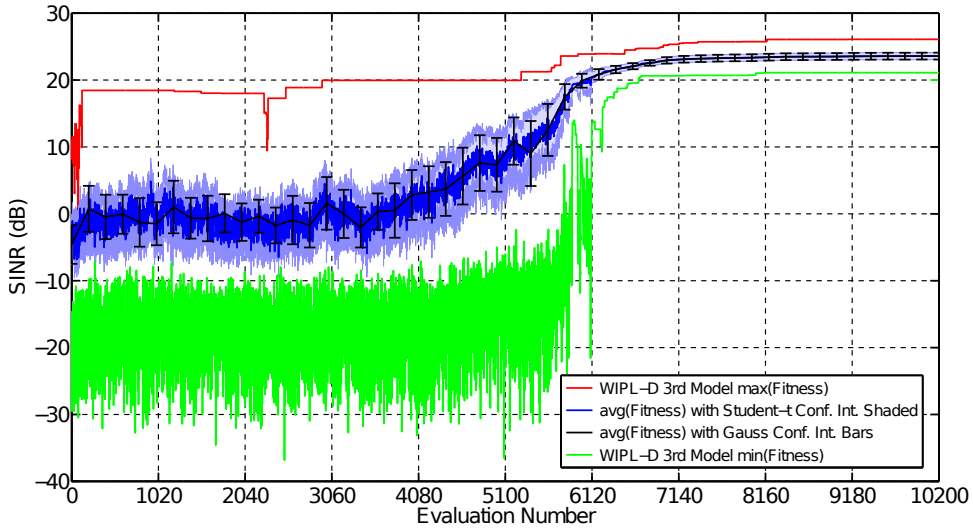


Figure 5.37: SA simulated performance curves collected over 30 independent runs using AntNet WIPL-D (third model) with SOI at 0° and two jammers at 45° and 200°.

The results for 30 simulations of SA are shown in Figures 5.37 and 5.38, and the resulting performance graphs as well as the best case azimuth plots for the *in-situ* measurements are shown in Figures 5.39 and 5.40. The simulations agreed well with *in-situ* measurements where the simulated SA algorithm gave conservative values for the maximum SINR values at convergence. Like the static three jammer case, SA is slower than both the SGA and TDGA to converge, but SA converges slightly faster when only two jammers are present.

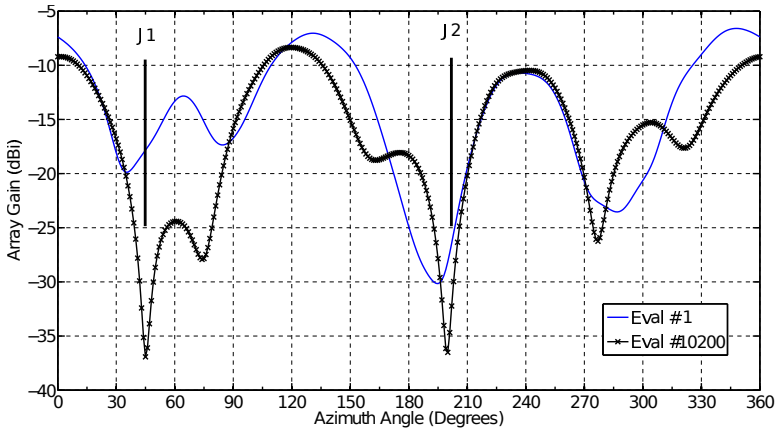


Figure 5.38: AntNet WIPL-D (third model) best-case SA optimized azimuth radiation pattern with a single SOI at 0° and three jammers at 45° and 200°.

The maximum SINR value predicted by the SA simulations is 26.09 dB at evaluation 10,200, and the minimum SINR value is 21.09 dB. The 95% confidence intervals based on 30 independent runs is ±0.51 dB and ±0.53 dB for Gaussian and Student-t distributions. This is a 4.17% difference which is consistent with the number of independent trials used in the calculations. The *in-situ* measurements finds a solutions with a maximum SINR of 48.2 dB at evaluation 10200, and the worst case solution it finds during that evaluation has a 13.83 dB SINR value. The 95% confidence intervals based on 30 independent *in-situ* trials are ±2.52 dB and ± 2.63 dB assuming Gaussian and Student-t distributions, respectively. The *in-situ* confidence intervals are larger than the simulated confidence intervals by a factor of 5. This is likely due to step-attenuator and phase shifter tolerances that can not be accounted for in simulations.

Unlike the three-jammer case, SA begins its exploitation stage both in simulation after evaluation 3060 and *in-situ* after evaluation 4080. Because the two-jammer static

case is easier to solve, the average fitness begins increasing linearly earlier during the algorithm's search. The best-case converges azimuth plots for (shown in Figures 5.38 and 5.40) are similar in shape with the AntNet WIPL-D model predicting shallower null-depths.

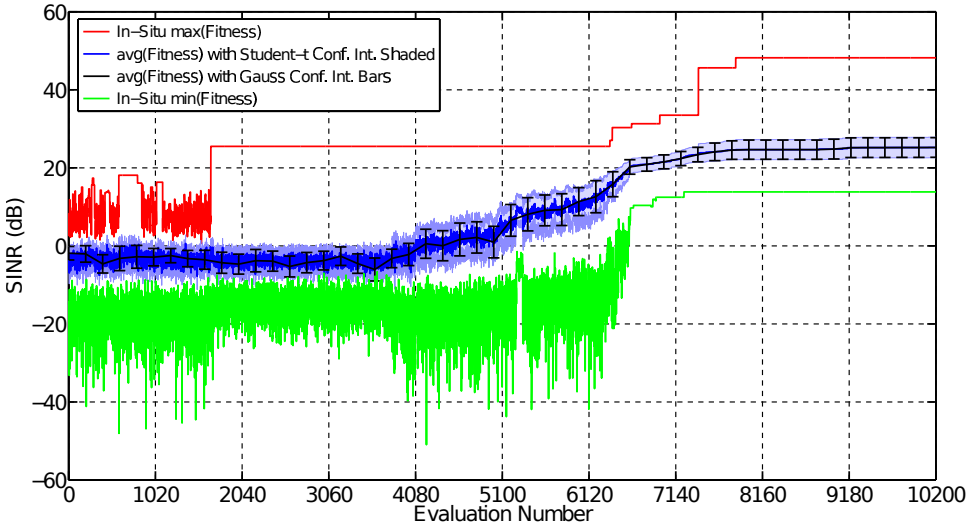


Figure 5.39: SA *in-situ* performance curves collected over 30 independent runs with SOI at 0° and two jammers at 45° and 200°.

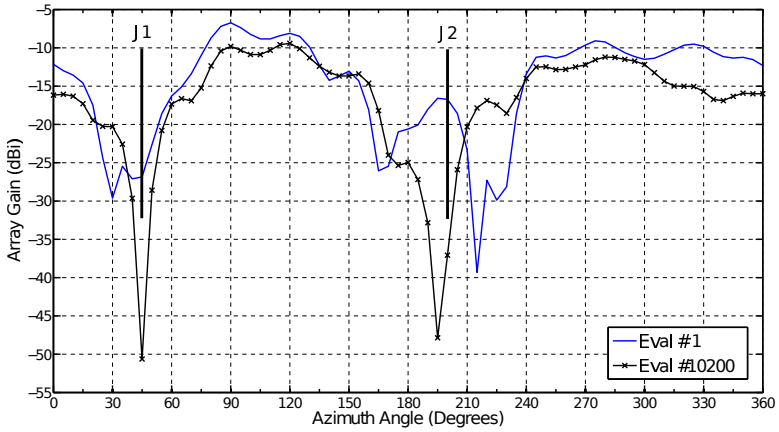


Figure 5.40: Best-case SA *in-situ* optimized azimuth radiation pattern with a single SOI at 0° and two jammers at 45° and 200°.

The performance curves and azimuth plots for the HCA simulations are shown in

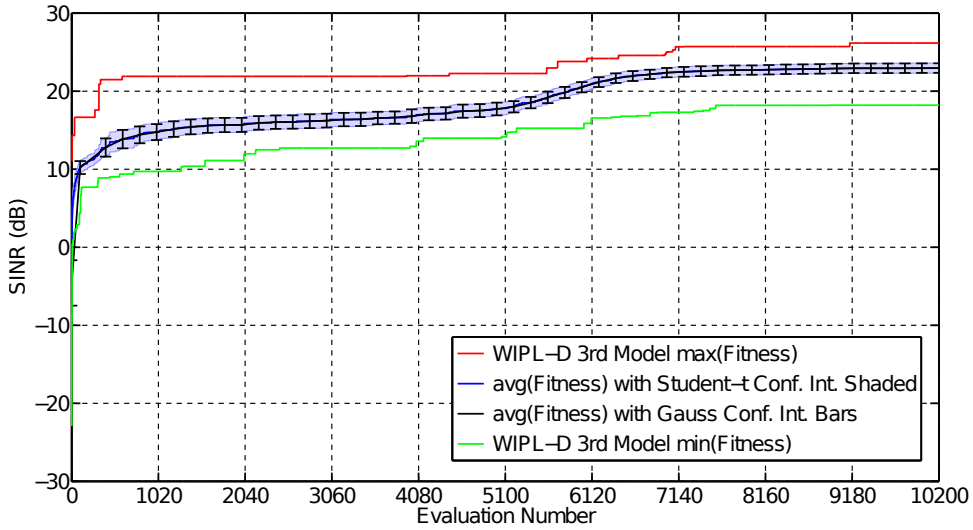


Figure 5.41: HCA simulated performance curves collected over 30 independent runs using AntNet WIPL-D (third model) with SOI at 0° and two jammers at 45° and 200°.

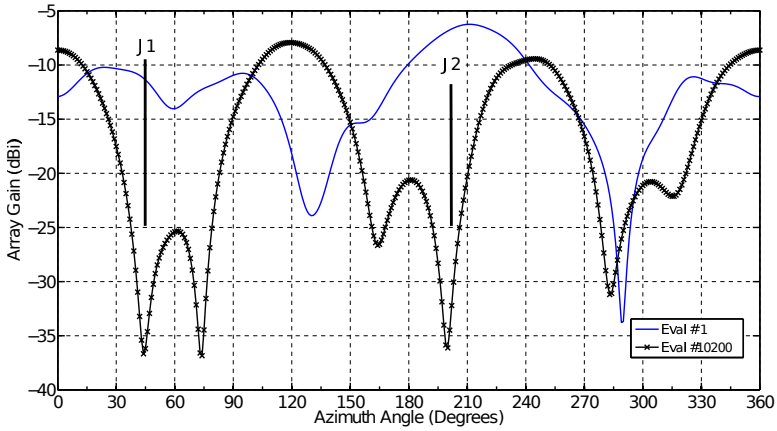


Figure 5.42: AntNet WIPL-D (third model) best-case HCA optimized azimuth radiation pattern with a single SOI at 0° and three jammers at 45° and 200°.

Figures 5.41 and 5.42. At evaluation 10,200, the best solution has a 21.16 dB SINR, and the worst solution has a SINR of 18.20 dB. The 95% confidence intervals at the final evaluation are ±0.61 dB and ±0.63 dB for Gaussian and Student-t distributions. The final azimuth plot for the best converges solution is similar in shape and null-depths compared to the azimuth plots found by the SGA, TDGA, and SA simulations.

5.5 Algorithm Performance With Stepped Mobile Signals

It is desirable to investigate the performance of stochastic algorithms in maximizing an array's SINR when the signals are mobile. Weile and Michielssen discuss the simulated performance of a GA with dominance and diploidy [28]. Because they invested a case where a set of interfering signals changed directions every 20 generations, and their GA is able to adapt in simulations, it is a natural place to verify their results *in-situ* with a hardware system.

A switching cycle of 10 generations is chosen because this presents a situation where the algorithm has not converged before the signals change directions. The situation of one SOI with two mobile jammers is considered, and performance is evaluated for the SGA, TDGA, SA, and HCA. The SOI is held constant at 0° and the interfering signals originate at 45° and 200°. The jammers are switched to 120° and 300° after 10 generations, are held at those positions for another 10 generations, and are switched back to their original directions.

Performance curves for 30 independent trials of the SGA adapting to this situation are shown in Figure 5.43. It can be seen that the SGA adapts to mobile jammers that change directions every 10 generations. The mobility of signals keep the SGA in flux, as the mean and standard-deviations of the population's Hamming distance generally increase after the jammers change directions (Figure 5.43). The SGA clearly does not converge by the time the jammers change directions because

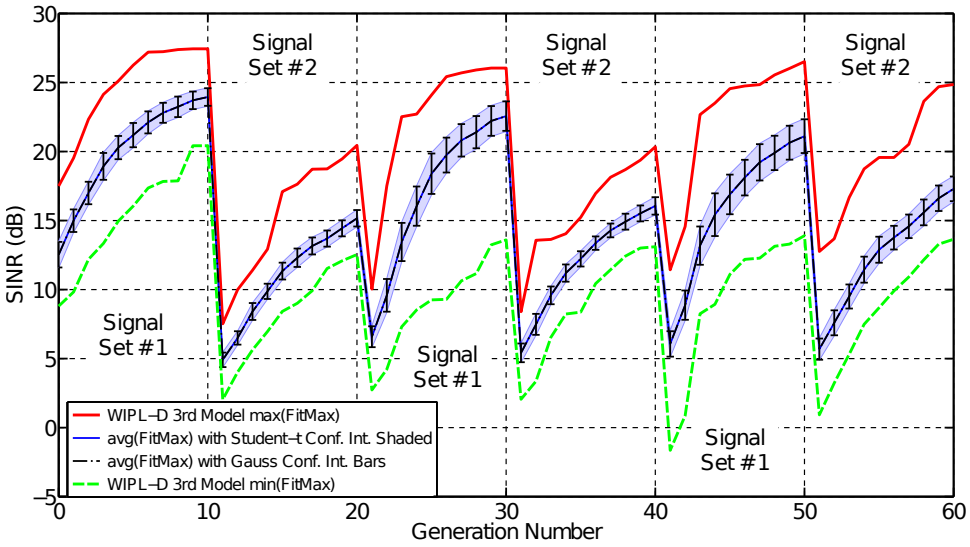


Figure 5.43: SGA simulated performance curves collected over 30 independent runs using AntNet WIPL-D (third model) with static SOI at 0° and mobile two jammers that switch between [45°, 200°] and [120°, 300°].

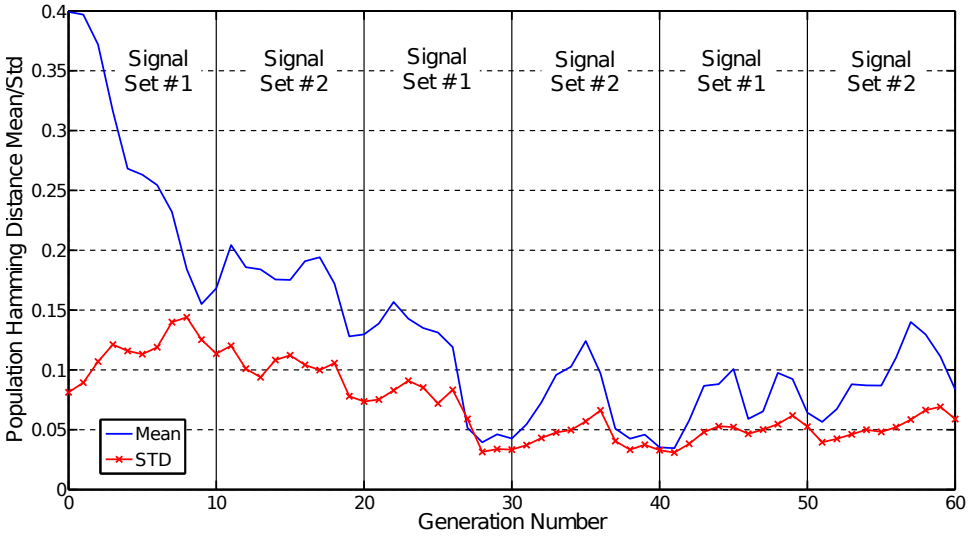


Figure 5.44: Hamming plot for best out of 30 independent SGA simulations using AntNet WIPL-D (third model) with static SOI at 0° and mobile two jammers that switch between [45°, 200°] and [120°, 300°].

the average SINR fitness of the best solutions do not level out, and the population mean and standard-deviation Hamming distance do not drop and stay below 0.05. The resulting azimuth plots for the best solution found by the SGA are shown in

Figure 5.45. The SGA clearly moves the two nulls from 45° and 120° to 200° and 300° even though the SGA sacrificed the SOI's gain by about 3 dB in the process.

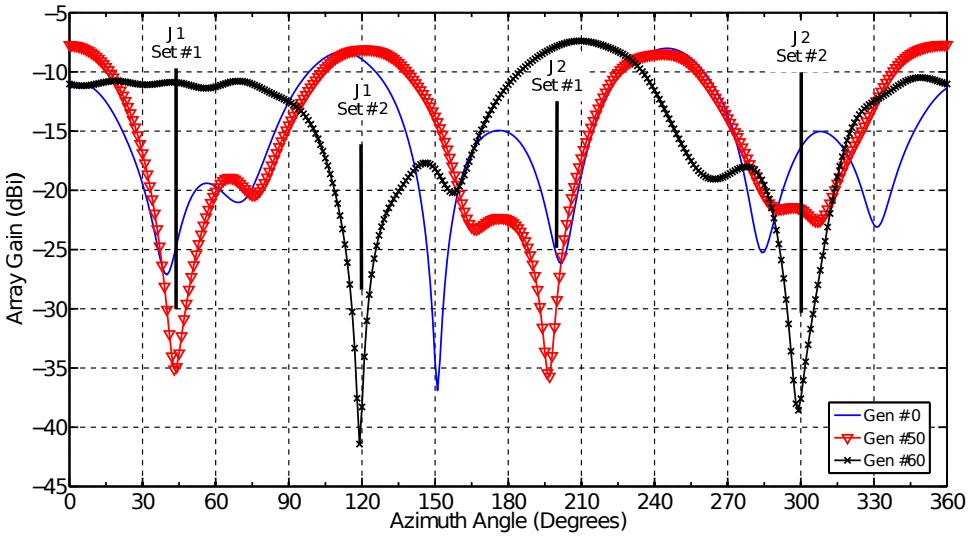


Figure 5.45: Best-case azimuth radiation patterns out of 30 independent SGA simulations using AntNet WIPL-D (third model) with static SOI at 0° and mobile two jammers that switch between [45°, 200°] and [120°, 300°].

The results shown in Figures 5.43 – 5.45 are unexpected, as it is expected that the SGA cannot adapt to mobile signals because the crossover operator tends to create multiple copies of the same string as the SGA moves towards convergence [24, 26, 27]. To understand what allowes the SGA to adapt to mobile signals within 10 generations, the experiment is modified. With 30 independent simulation trials, the SGA is run for a total of 101 generations. For the first 51 generations (including 0^{th} generation), the SOI and jammers originated at 0°, 45°, and 200°. For the final 50 generations, the SOI's direction is held constant, but the jammers move to 120° and 300°.

Two separate experiments are conducted with these signal direction sets. In the first experiment, the same probability of mutation ($P_m = 0.01$) is used in the final 50

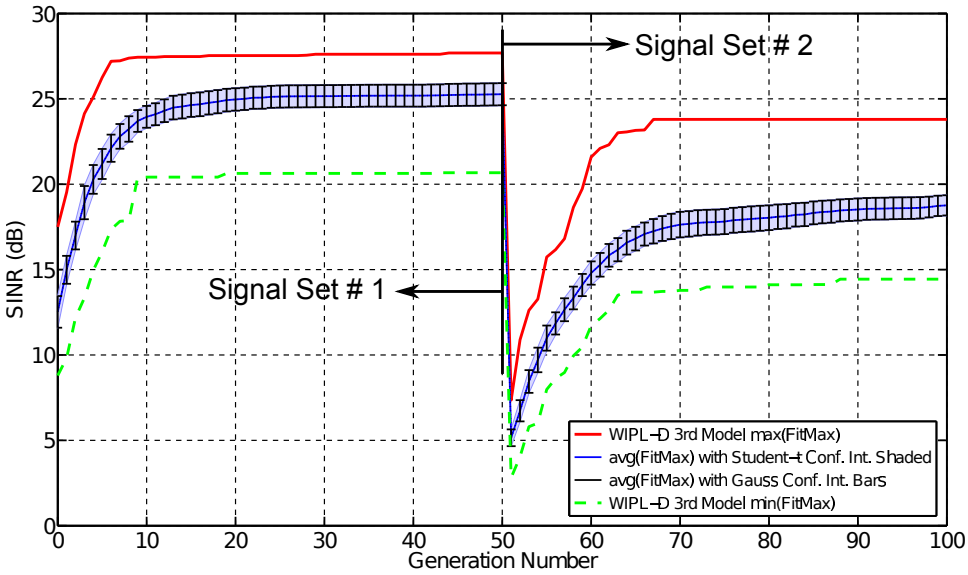


Figure 5.46: SGA simulated performance curves collected over 30 independent runs using AntNet WIPL-D (third model) with static SOI at 0° and mobile two jammers that switch between [45°, 200°] and [120°, 300°]. The SGA is run for 101 generations, and jammers changed directions after 51 generations with mutation turned-on after generation 50.

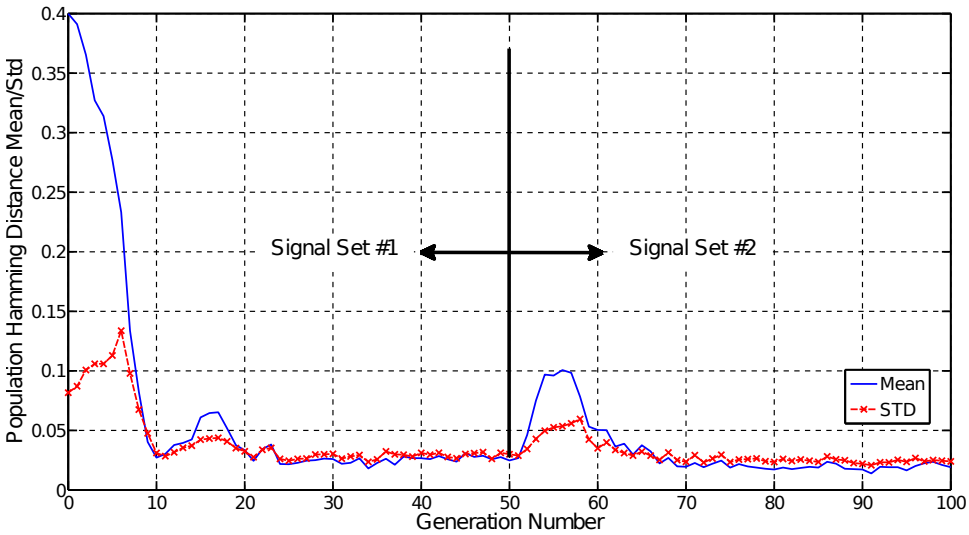


Figure 5.47: Hamming plot for best out of 30 independent SGA simulations using AntNet WIPL-D (third model) with static SOI at 0° and mobile two jammers that switch between [45°, 200°] and [120°, 300°]. The SGA is run for 101 generations, and jammers changed directions after 51 generations with mutation turned-on after generation 50.

generations with the resulting performance curves and Hamming distance plots shown in Figures 5.46 and 5.47. In the second experiment, the probability of mutation is turned off (i.e., $P_m = 0$) for the final 50 generations, and the resulting performance graphs are shown in Figures 5.48 and 5.49.

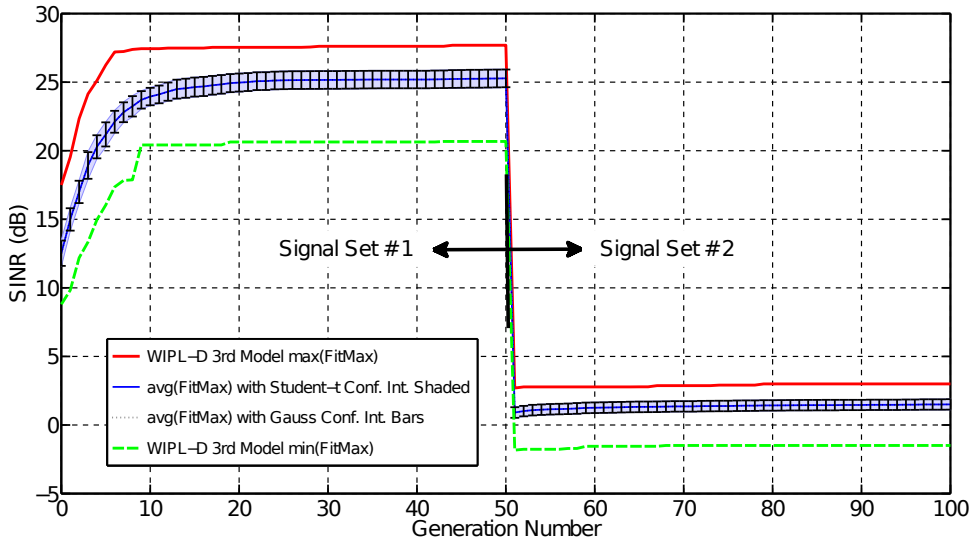


Figure 5.48: SGA simulated performance curves collected over 30 independent runs using AntNet WIPL-D (third model) with static SOI at 0° and mobile two jammers that switch between [45°, 200°] and [120°, 300°]. The SGA is run for 101 generations, and jammers changed directions after 51 generations with mutation turned-off after generation 50.

The purpose of these two experiments is two-fold. First, the SGA is allowed to operate on the first signal set until it converges, so its performance when the jammers changed direction can be observed. Second, it is necessary to understand the mechanism that allows the SGA to re-adapt to mobile signals in an AWGN channel. In fact, after mutation is turned off, the SGA's SINR fitness drops and remains between ±3 dB from generation 51 onward. The population's mean and standard-deviation Hamming distance also dropped to 0, as this indicates that the

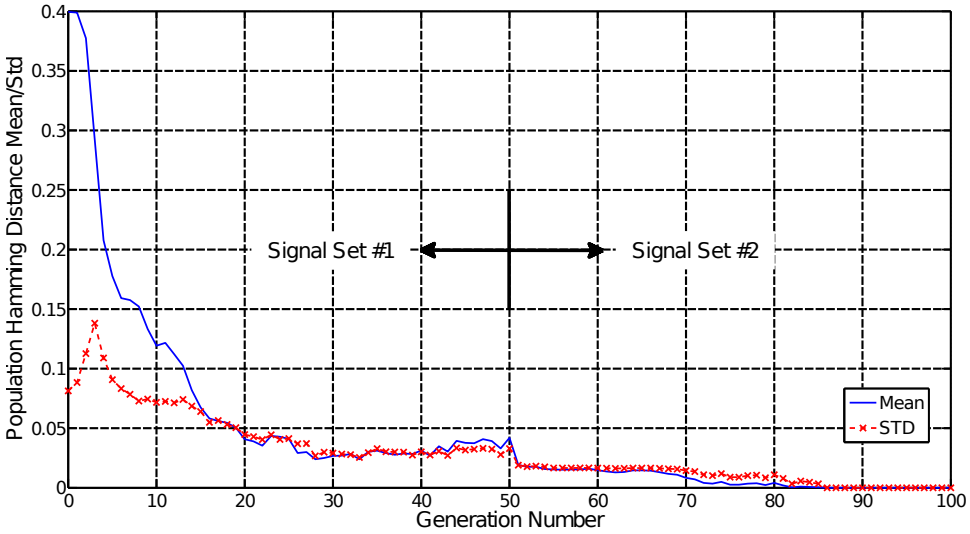


Figure 5.49: Hamming plot for best out of 30 independent SGA simulations using AntNet WIPL-D (third model) with static SOI at 0° and mobile two jammers that switch between [45°, 200°] and [120°, 300°]. The SGA is run for 101 generations, and jammers changed directions after 51 generations with mutation turned-off after generation 50.

SGA eventually converges to a population that consists of 200 copies of the same string. This is consistent with the theory of GAs using mate-selection and crossover only as De Jong discusses in [27]. It is important to note that we do not claim that the SGA can adapt to mobile signals in a Rayleigh fading channel, but the importance of the GA's mutation operator should not be underestimated in understanding how the GA operates in non-ideal environments.

The TDGA is simulated in maximizing SINR with one static SOI and two mobile jammers. Measurements on the TDGA are collected *in-situ* in an anechoic chamber with mobile jammers when the SOI is static or mobile [79,80]. In the simulations, the SOI is static at 0°, and the jammers switch between {45°, 200°}, and {120°, 300°} with performance curves shown in Figure 5.50. The TDGA performance graphs are

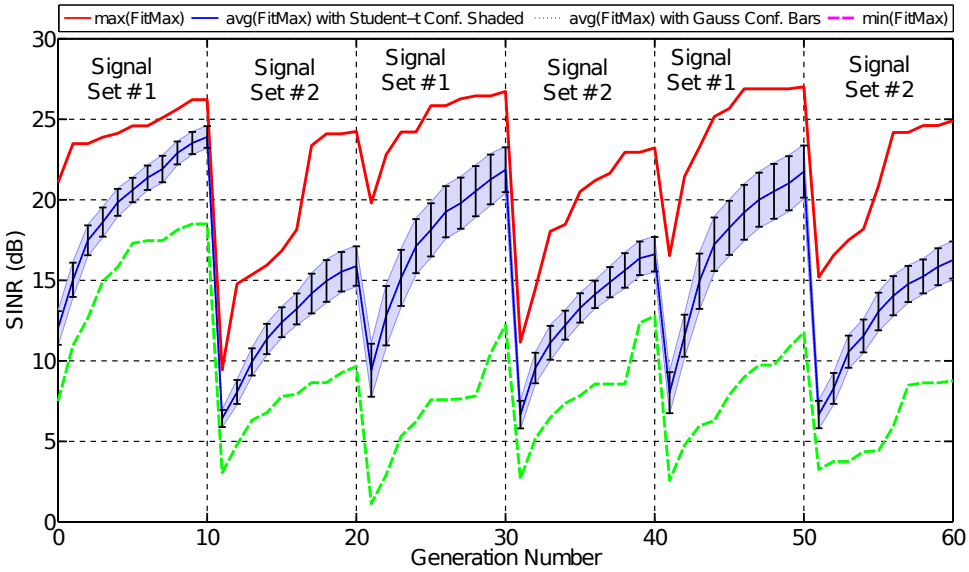


Figure 5.50: TDGA performance curves collected over 30 independent simulations using AntNet WIPL-D (third model) with static SOI at 0° and mobile two jammers that switch between [45°, 200°] and [120°, 300°] every 10 generations.

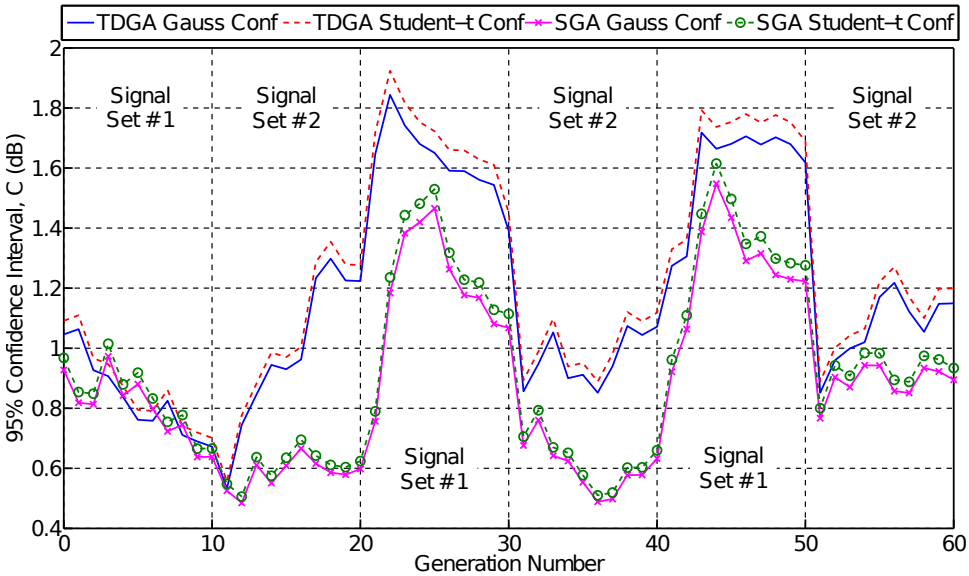


Figure 5.51: Comparison of SGA and TDGA confidence intervals over 30 independent simulations of each with static SOI at 0° and mobile two jammers that switch between [45°, 200°] and [120°, 300°] every 10 generations.

slightly better than the SGA performance graphs in Figure 5.43 because the TDGA has maximum out of the top 30 solutions with SINRs of 27.01 dB and 24.91 dB at

generations 50 and 60, and the SGA has maximum SINRs of 26.51 dB and 24.85 dB at those generations. The SGA, however, has worst of the best solutions that dropped below 0 dB SINR whereas the TDGA's worst of the best stayed above 1 dB SINR after the jammers changed directions. The 95% confidence intervals for both the SGA and TDGA in simulation are similar with the TDGA having confidence intervals that are 0.4 dB higher than the SGA after generation 20 (see Figure 5.51).

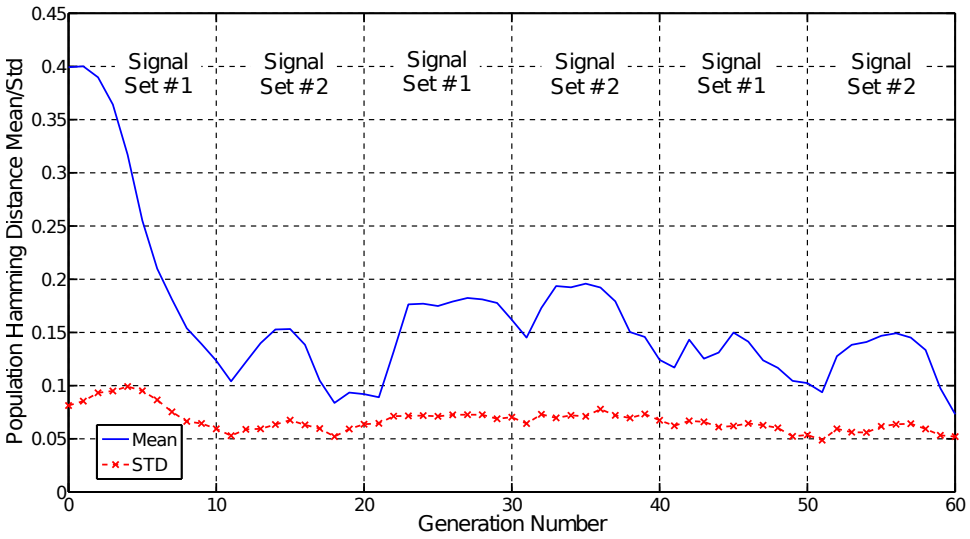


Figure 5.52: TDGA Hamming distance plots for best solution found in 30 independent simulations using AntNet WIPL-D (third model) with static SOI at 0° and mobile two jammers that switch between [45°, 200°] and [120°, 300°] every 10 generations.

The Hamming distance vs. generation plots for the best solution that the TDGA finds out of 30 independent runs is shown in Figure 5.52. The resulting azimuth plots for the best solution are shown in Figure 5.53. The TDGA optimizes the patterns at generations 50 and 60 to effectively null the jammers. The SOI gain decreases by 2 dB by generation 60, but its gain at generation 50 is the same compared to its initial value. If the SOI changes directions, the mainlobe is wide enough such that the SOI

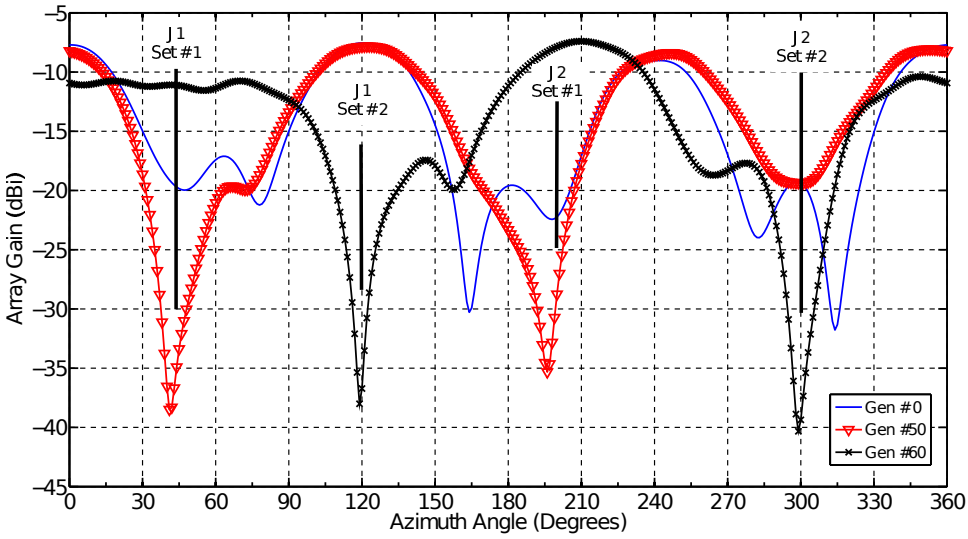


Figure 5.53: TDGA azimuth radiation plots for best solution found in 30 independent simulations using AntNet WIPL-D (third model) with static SOI at 0°. Shown are best generation 0 (initial) plot, generation 50 with two jammers nulled at [45°, 200°], and generation 60 with jammers nulled at [120°, 300°].

can move to 75° without being significantly attenuated by the antenna array.

5.6 Algorithm Tracking of Continuously Mobile Interference

In previous sections, mobile signals move instantaneously and allow the stochastic algorithms time to settle into optimal solutions before the signals change directions again. In this section we investigate how well the SGA and TDGA tracks mobile interference that continually change directions, as this would prevent both algorithms from finding optimal solutions before the interference changes directions.

In the first experiment, we run 30 independent runs of the SGA with the SOI static at 0°, and two jammers that move continuously over a period of 50 generations (plus generation 0) from $\{45°, 200°\}$ to $\{120°, 300°\}$. The resulting performance graphs

for the best 30 solutions out of these 30 simulations are shown in Figure 5.54.

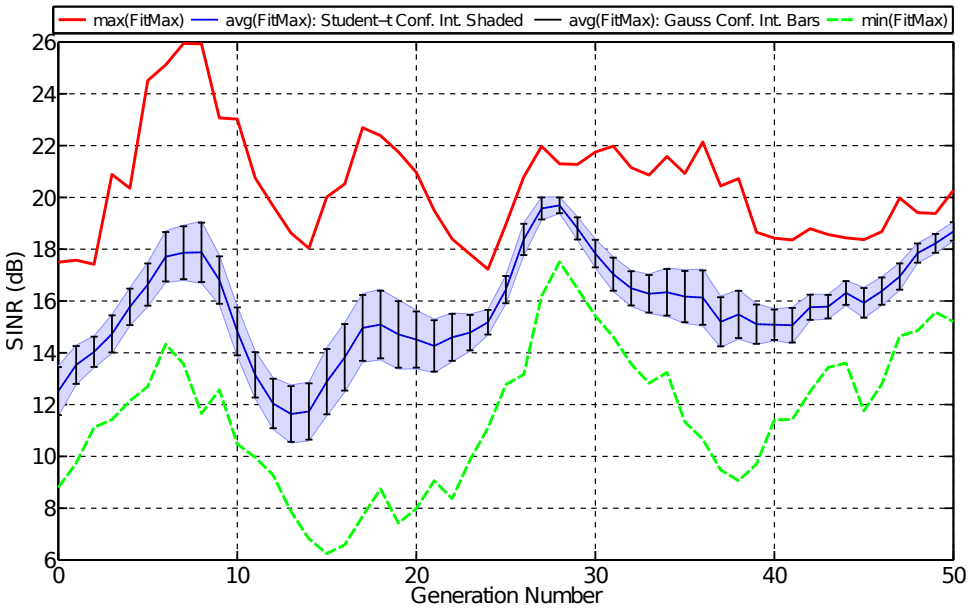


Figure 5.54: SGA performance graphs for 30 independent runs with static SOI at 0° and jammers that continuously move over a course of 50 generations from $\{45°, 200°\}$ to $\{120°, 300°\}$.

It is clear that from Figure 5.54 that the SGA can track the two mobile signals. The SGA obtains a maximum (best of the best 30 solutions) SINR of 26 dB by generation 7, and its fitness varies in a sinusoidal fashion with varying frequency until the simulations end. The best case solution has a SINR of just over 20 dB, and the worst of the best solutions has 15 dB SINR by generation 50. The worst of the best 30 solutions maintains a SINR above 6 dB throughout the simulations. It is also interesting to note that the Student-t and Gaussian 95% confidence intervals are indistinguishable, and they vary widely versus generation number.

The resulting Hamming distance plots for the best final solution that the SGA finds are shown in Figure 5.55. As with static signals, the SGA's diversity quickly

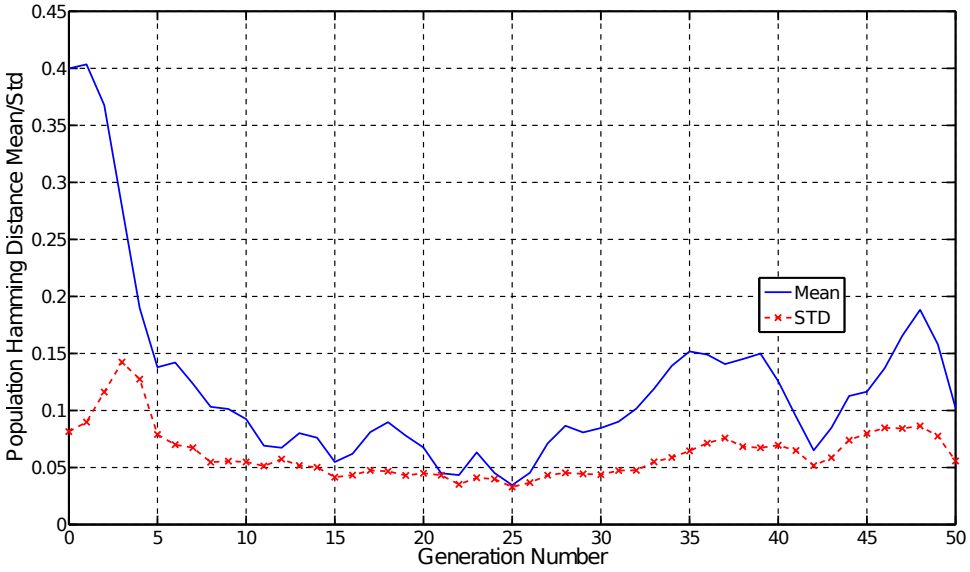


Figure 5.55: SGA best-case Hamming distance plots with static SOI at 0° and jammers that continuously move over a course of 50 generations from $\{45°, 200°\}$ to $\{120°, 300°\}$.

decreases since its mean Hamming distance dropped from 0.4 to less than 0.1 within 10 generations. For most of the run, its standard-deviation Hamming distance remains around 0.05 but peaks near 0.075 twice after generation 30. The mean Hamming distance also increases to 0.15 by generation 35, dipped to around 0.07 by generation 42, and peaks to a local maximum of roughly 0.18 by generation 48. This means that the mutation operator allows enough population diversity for the crossover operator to continuously track the mobile jammers.

It is clearly visible that the SGA successfully tracks the two continuously mobile jammers as they move across the azimuth plane (see Figure 5.56). In this best-case example, the SGA finds solutions that null the jammers in their initial positions. The dashed vertical lines indicate the initial jammer azimuth positions, and the dashed

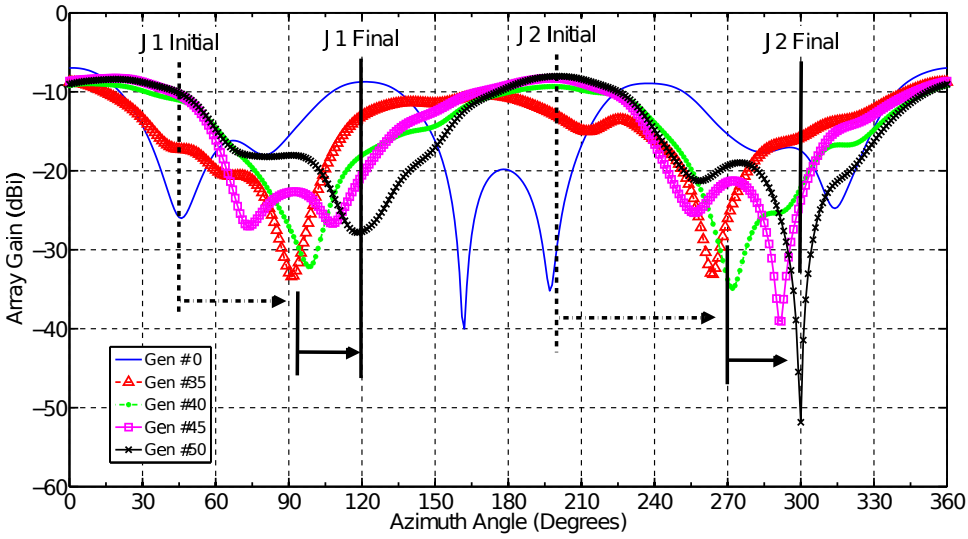


Figure 5.56: SGA best-case azimuth radiation plots with static SOI at 0° and jammers that continuously move over a course of 50 generations from $\{45°, 200°\}$ to $\{120°, 300°\}$.

horizonal arrows indicate the jammers' movements towards their generation 35 locations. It can be seen from Figure 5.56 that the SGA tracks the two mobile signals from $\{45°, 200°\}$ and places nulls with at least 15 dB depth with respect to the SOI at the jammers' generation 35 azimuth positions of $\{97.5°, 270°\}$. The SGA continues tracking the two jammers until they reach their final positions where the SGA nulls the two jammers 18.67 dB and 42.81 dB with respect to the SOI's array gain. The SOI array gain remains relatively constant for the azimuth plots shown, and the SGA did a far better job of nulling the first jammer compared to the second jammer.

Before investigating the TDGA, it is necessary to evaluate and compare the SGA's performance when the two jammers' rate of movement in azimuth are halved, and the SGA is allowed to run for 100 generations (plus generation 0). This experiment determines if the SGA can null the first jammer better with respect to the SOI

array gain if the SGA is given more time to adapt to slower signals. The performance curves for the best 30 solutions out of 30 independent runs are shown in Figure 5.57.

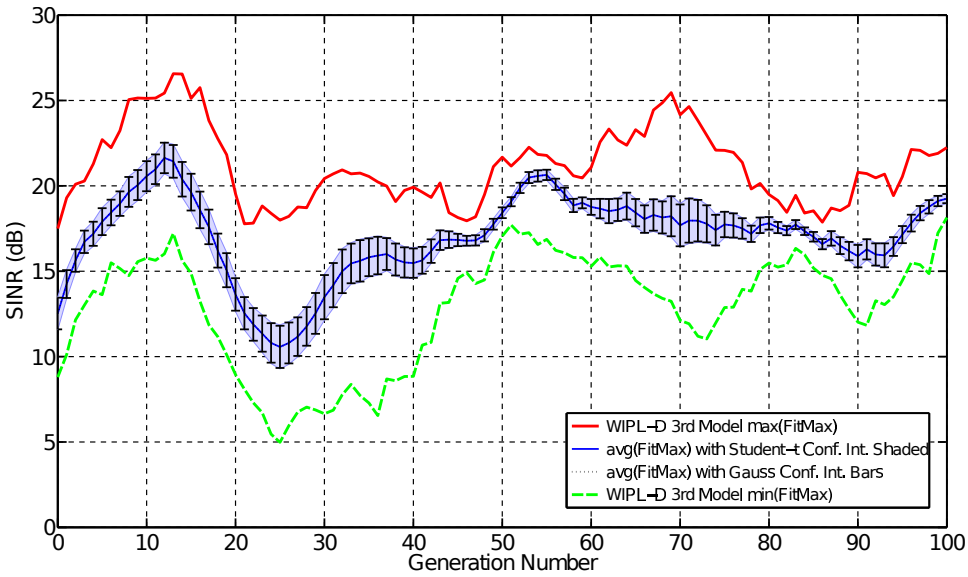


Figure 5.57: SGA performance curves for 30 independent runs with static SOI at 0° and jammers that continuously move over a course of 100 generations from $\{45°, 200°\}$ to $\{120°, 300°\}$.

Although the 30 best solutions that the SGA finds at generation 100 are better compared to the solutions shown in generation 50 (see Figure 5.54), the SGA's fitness varies roughly in the same manner while it evolves solutions from generation 0 to 100. The best overall solution peaks above 26 dB SINR by generation 13, but the worst of the best solutions has a SINR that drops to 5 dB around generation 25. The Student-t and Gaussian 95% confidence intervals are also indistinguishable and vary considerably versus generation number.

The best solution's Hamming distance vs. generation number behaves in a manner similar to the SGA 50 generation experiment (see Figure 5.58). The mean Ham-

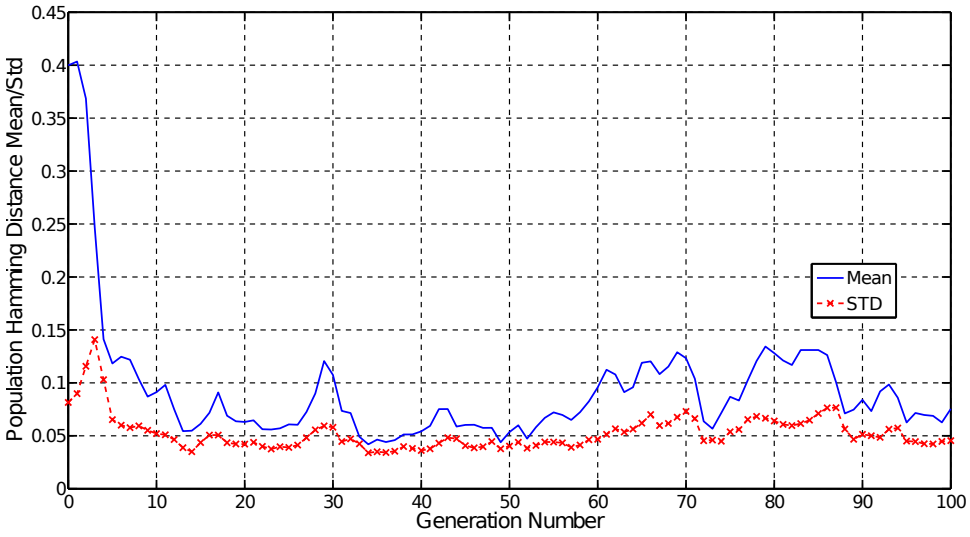


Figure 5.58: SGA best-case Hamming distance plots with static SOI at 0° and jammers that continuously move over a course of 100 generations from $\{45°, 200°\}$ to $\{120°, 300°\}$.

ming distance quickly drops below 0.1 by generation 10, and it oscillates until the end of the simulation peaking above 0.1 mean Hamming distance several times. The standard-deviation Hamming distance varies somewhat but remains close to a value of 0.05. This behavior comes from the mutation operator that reintroduces enough diversity back into the population such that the crossover operator can track the two continuously mobile jammers.

For the best-case solution that the SGA finds, the azimuth radiation patterns in Figure 5.59 indicate that the SGA performs better when the two jammers continuously move over a period of 100 generations. The slower speed allows the SGA more time to track the signals and find better solutions. By generation 100, the SGA places a null at 120° that is 23.37 dB down from the SOI, and the null placed at 300° is 25.76 dB down from the SOI. The generation 100 SOI gain shown in Figure 5.59 is

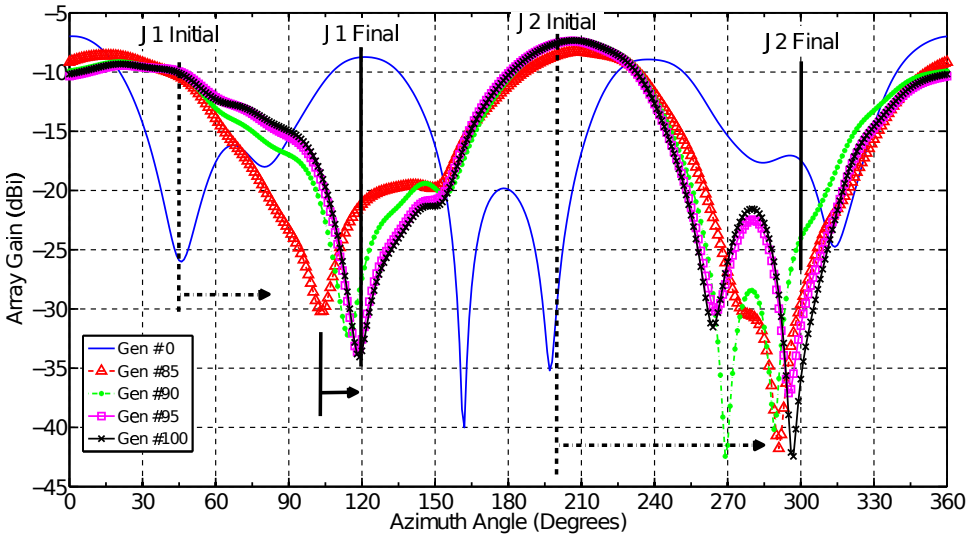


Figure 5.59: SGA best-case azimuth radiation plots with static SOI at 0° and jammers that continuously move over a course of 100 generations from $\{45°, 200°\}$ to $\{120°, 300°\}$.

roughly the same as the generation 50 SOI gain in Figure 5.56. Although running the SGA for 100 generations improves the null directed towards the first jammer, the SGA sacrifices the second jammer's null by 17.05 dB compared to the generation 50 azimuth pattern in Figure 5.56.

In the next set of experiments, thirty independent simulations of the TDGA are run with the SOI static at 0°, and two jammers that move continuously over a period of 50 generations (plus generation 0) from $\{45°, 200°\}$ to $\{120°, 300°\}$. This experiment is repeated by running the TDGA over a period of 100 generations, as this effectively slows the jammers' movement down by a factor of two, and it allows the TDGA more time to adapt. As with the SGA, the goal is to see if the TDGA can find better solutions if the signals move at a slower rate, and if the TDGA has more time to adapt to them. The results for the TDGA run over 50 generations are shown

in Figures 5.60 and 5.62, and the results for the TDGA run over 100 generations are then shown in Figures 5.63 and 5.65.

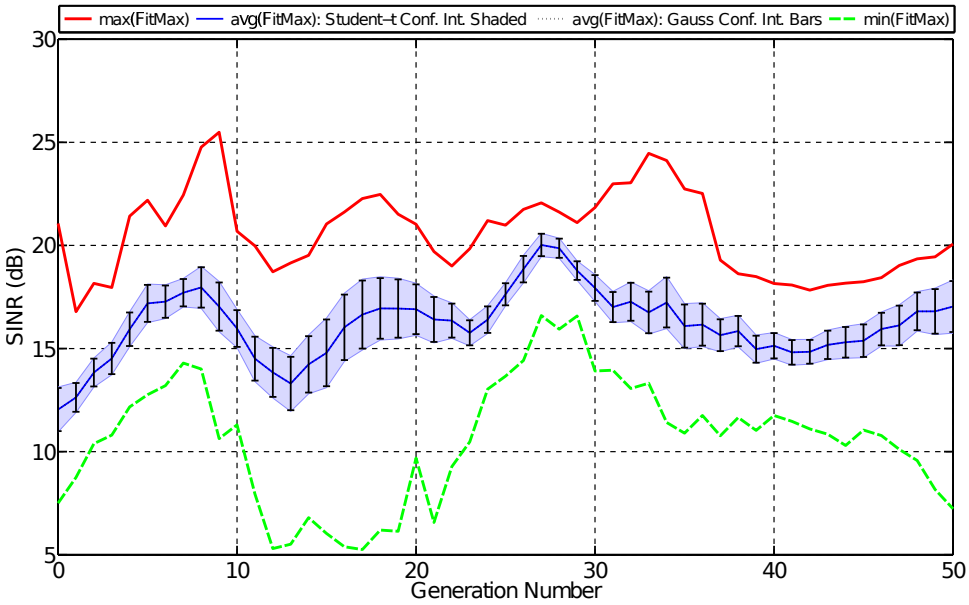


Figure 5.60: TDGA performance graphs for 30 independent runs with static SOI at 0° and jammers that continuously move over a course of 50 generations from $\{45°, 200°\}$ to $\{120°, 300°\}$.

The TDGA's performance curves in Figure 5.60 varies in a sinusoidal fashion like the SGA counterpart. The generation varying maximum curve peaked around 25 dB at generation 9 (less than 1 dB than the SGA maximum curve in Figure 5.54). The minimum of the best 30 solutions vs. generation dips towards 5 dB which is also less than the corresponding SGA performance graph. The TDGA's 95% confidence intervals are roughly the same as the SGA's confidence intervals. Thus from a fitness perspective, there is not a significant difference between the TDGA and SGA when they are run for 50 generations with two continuously mobile jammers.

Unlike the SGA, the TDGA maintains higher diversity through the entire sim-

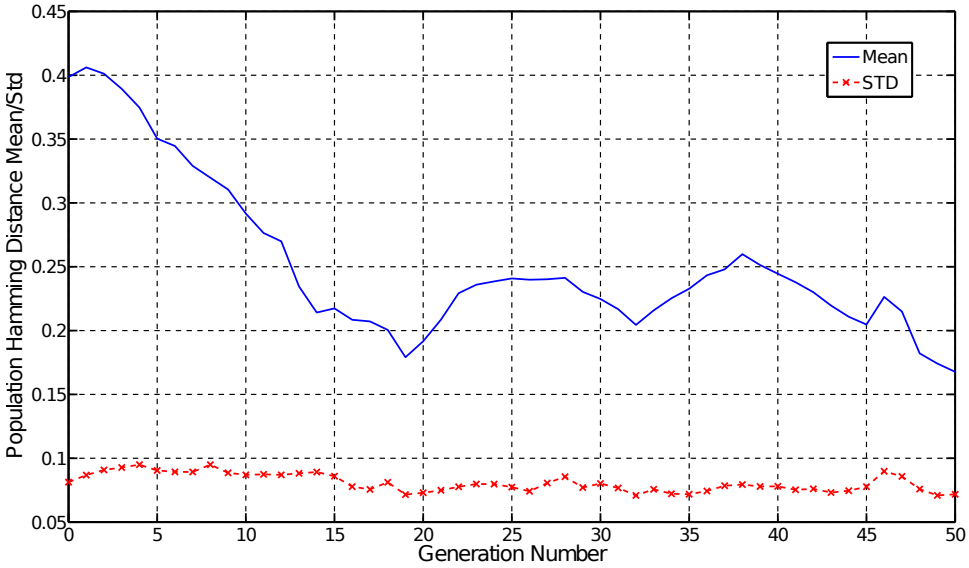


Figure 5.61: TDGA best-case Hamming distance plots with static SOI at 0° and jammers that continuously move over a course of 50 generations from $\{45°, 200°\}$ to $\{120°, 300°\}$.

ulation (see Figure 5.61 for the best solution's Hamming distance vs. generation curves). The TDGA's mean Hamming distance linearly decreases and drops below 0.2 after generation 18, and it varies between 0.2 and 0.25 for most of the remaining generations. However, the standard deviation of the population Hamming distance remains roughly constant around 0.075 throughout the entire simulation.

The TDGA is also able thwart continuously mobile jammers as they move from $\{45°, 200°\}$ to $\{120°, 300°\}$ with the azimuth plots for the best solution shown in Figure 5.62. It is evident that the best solution that the TDGA finds places nulls at 45° and 200° with the shallowest null 14.84 dB below the SOI at generation 0. The TDGA tracks the two jammers with the first jammer's null deepening at first then getting shallower at 120°. A similar behavior is observed for the second jammer where the

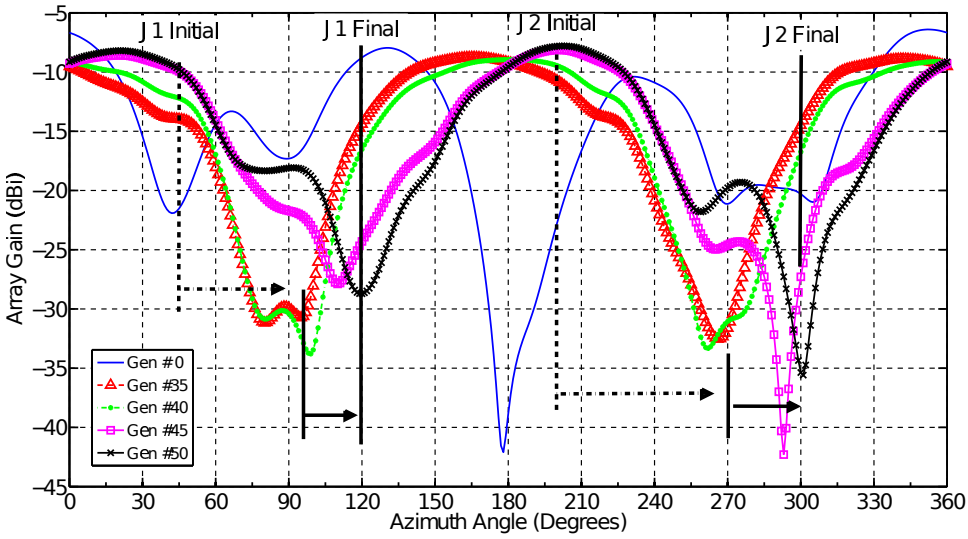


Figure 5.62: TDGA best-case azimuth radiation plots with static SOI at 0° and jammers that continuously move over a course of 50 generations from $\{45°, 200°\}$ to $\{120°, 300°\}$.

null first deepens then becomes less deep as it approaches 300°. This is related to the sinusoidally varying nature of the performance graphs in Figure 5.61. When the simulation ends, the TDGA places nulls towards the two jammers that are 19.58 dB and 26.24 dB down from the SOI.

The next step is to investigate the TDGA's behavior when the mobile jammers are slowed down and allowed to vary from $\{45°, 200°\}$ to $\{120°, 300°\}$ over 100 generations. The resulting performance graphs shown in Figure 5.63 has a noisy sinusoidal behavior similar to the SGA that is run over 100 generations. The overall limits for the TDGA are roughly the same as the SGA. The sinusoidal nature of these curves is reflected in null depths that have varying depths as the jammers sweep across the azimuth plane.

The Hamming distance plots for the best solution found by the TDGA are shown

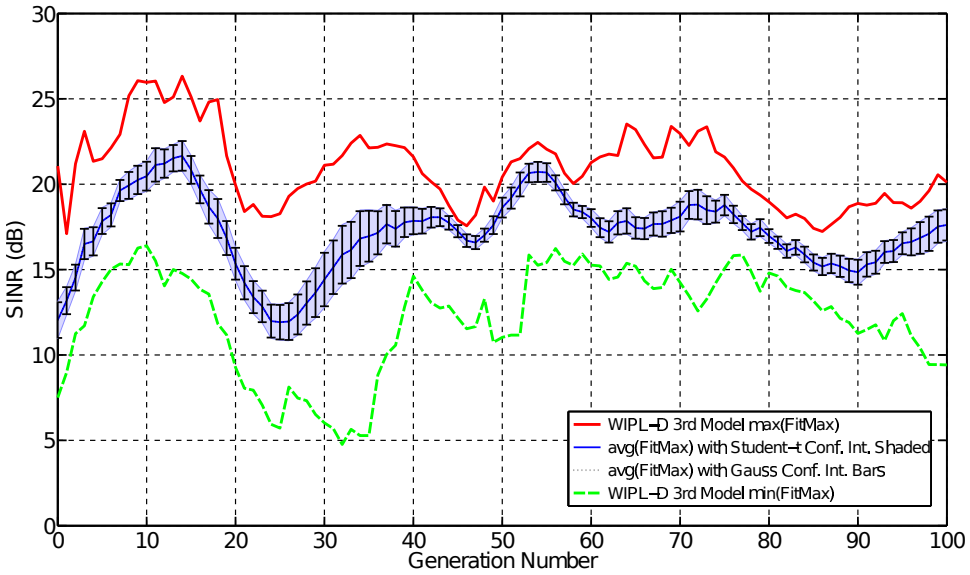


Figure 5.63: TDGA performance graphs for 30 independent runs with static SOI at 0° and jammers that continuously move over a course of 100 generations from $\{45°, 200°\}$ to $\{120°, 300°\}$.

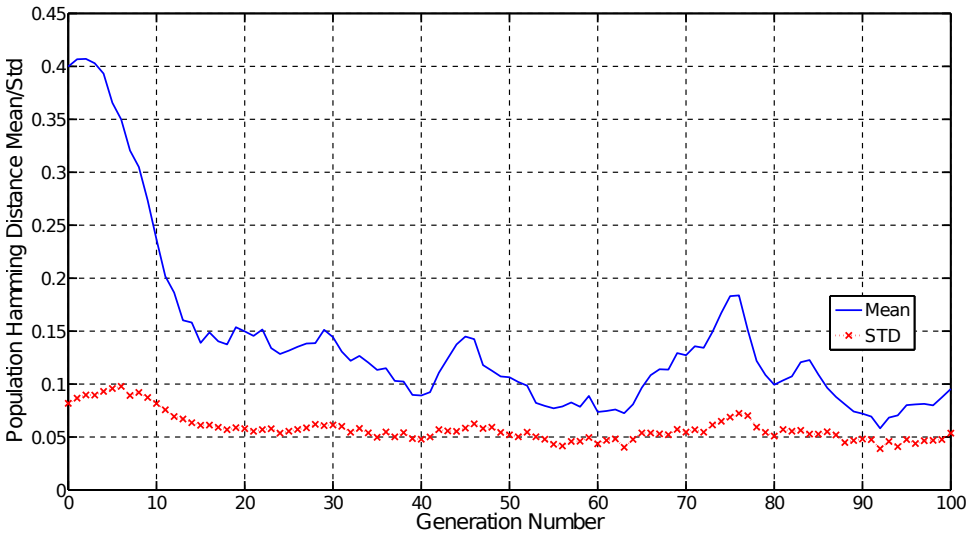


Figure 5.64: TDGA best-case Hamming distance plots with static SOI at 0° and jammers that continuously move over a course of 100 generations from $\{45°, 200°\}$ to $\{120°, 300°\}$.

in Figure 5.64. Although the mean Hamming distance for the 100 generation TDGA run is considerably lower than the 50 generation TDGA run, this distance is still

higher than its SGA counterpart. The mean Hamming distance initially decreases faster and drops below 0.15 by generation 15. Although the mean Hamming distance drops below 0.1 after generation 38, the descent is more gradual between Generations 15 and 38.

The azimuth plots for the best-case solution found by TDGA in generation 100 are shown in Figure 5.65. The TDGA tracks both jammers as they sweep across the azimuth plane. While the nulls directed at the first jammer vary by a few decibels, it can be seen that the nulls directed at the second jammer vary more.

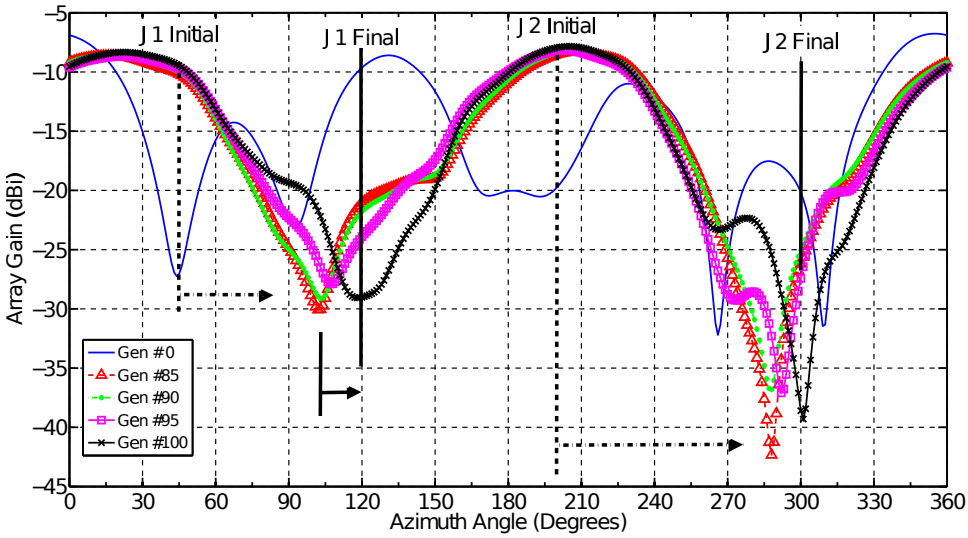


Figure 5.65: TDGA best-case azimuth radiation plots with static SOI at 0° and jammers that continuously move over a course of 100 generations from $\{45°, 200°\}$ to $\{120°, 300°\}$.

The act of slowing down the jammer's movements and running the TDGA improves the best-case azimuth plots slightly. In their final positions, the first jammer is 19.47 dB down from the SOI, and the second jammer is 29.05 dB down from the SOI. Finally, it should be noted that although both the SGA and the TDGA are sim-

ulated assuming an AWGN wireless channel and has similar performance, the TDGA is expected to perform better than the SGA in a non-ideal environment such as a Rayleigh fading channel. The diversity added by using Triallelic Diploid instead of Binary Haploid strings makes the TDGA more robust in such an environment. However, the TDGA is not perfect since its diversity decreases (although at a slower rate than the SGA) as the TDGA evolves populations across the generations.

5.7 Summary

In this chapter, data in simulation and *in-situ* with hardware is collected and compared on four stochastic algorithms in in optimizing anti-jamming beamforming antenna arrays. The algorithms fare very well in anti-jamming when a single SOI and two static jammers are present, but they show limited performance when three static jammers are present. The SGA and TDGA are both much faster than SA and HCA, as SA and HCA took over three times the evaluations (compared to 200 evaluations per generation) before converging to their final solutions in the three-jammer case. SA exhibits noisy behavior as expected and spends roughly half of the 10,200 evaluations temperature schedule searching for a solution before entering its exploitation stage. Although HCA is much smoother in its fitness performance behavior, it tends to get stuck on local maxima before finding and converging to better solutions.

The SGA and TDGA both adapt to stepped mobile signals. SA and HCA are able to thwart stepped mobile jammers, but the temperature schedule is repeated

multiple times during the simulations. There is a sacrifice between the number of temperature schedule repeats and the quality of the best solutions found by these algorithms. Although SA and HCA can re-adapt to stepped mobile signals by resetting the temperature values, this amounts to using faster temperature schedules that result in SA and HCA converging on local maxima instead of a global optimum.

In addition, the SGA and TDGA are both evaluated in adapting to two continuously mobile jammers when a single SOI is static in an AWGN wireless channel. It is found that both the SGA and TDGA are able to continuously mobile signals, and both algorithms find best-case solutions that have SINRs of at least 20 dB by the end of the simulations. This is a very important result because this is the first time that that it is demonstrated that the GA can adapt to signals that continuously change their directions. Previous research into GA adaptability to mobile signals focuses on signals that change directions instantaneously then stay static for a period of time. This allows the GA time to adapt to the signal. The reality is that signals move continuously in the field, so it is useful to demonstrate that the GA can adapt quickly enough to maintain good SINR levels.

Chapter 6

Dynamic, *In-Situ* and Simulated Hardware Fault Recovery of Beamforming Arrays

This chapter presents the research results in hardware fault recovery using the stochastic search algorithms utilized in the previous chapters. Hardware fault recovery is important because components can and will fail during operation in the field, and detection and recovery of these faults via a software algorithm keep the beamforming system operating (at least partially) until repairs can be performed. In the case of a satellite-based system, hardware fault recovery helps to achieve longer hardware lifespans. The results presented in this chapter show that search algorithms are capable of automatically recovering from hardware faults in a single antenna path whether the signals present are static or mobile.

The results presented here support the following contributions: First, it is demonstrated that stochastic algorithms can recover anti-jamming functionality after occurrence of hardware failures and localized faults in the array. Second, predictive models that detect occurrence of hardware faults in the array due to a drop in SINR fitness are developed. These models quantify changes in SOI gains and interfering signal null depths when an antenna element (i.e., antenna, phase shifter, or attenuator) faulted.

The models quantify multiple stochastic algorithms' (SGA, TDGA, SA, and HCA) abilities to recover from antenna element faults whether the signals are stationary or mobile.

The remainder of this chapter is organized as follows. Section 6.1 describes the setup for the *in-situ* experiments and simulations, and it lists the types of hardware faults that are emulated in the array. Section 6.2 discusses the performance of the stochastic algorithms in recovering from hardware faults when the signals are static, and section 6.3 discusses the *in-situ* experiment and simulation results when hardware fault recovery is performed in the presence of mobile signals.

6.1 Experimental and Simulation Setup

The setup for performing experiments in hardware fault recovery is the same as the anti-jamming setup described in Chapter 5. Various kinds of hardware faults in the RF path are emulated by setting a step attenuator or phase-shifter to values not chosen by the algorithm. The algorithm parameters from Section 5.1 will not be repeated here, but a matrix of simulations and *in-situ* experiments for evaluating the algorithms' abilities in anti-jamming with hardware fault recovery are listed in Table 6.1.

The types of faults considered are listed below in Table 6.2. The faults can be grouped into three categories regardless whether they are step-attenuator and phase shifter faults: hard, semi-hard, and soft. Hard faults represent hardware faults (i.e.,

Table 6.1: List of hardware fault recovery test cases explored by experiments and simulations.

	Static		**Stepped Mobile**
Algorithm	**2 jammer**	**3 Jammer**	**2 Jammer**
SGA	Sim	Sim	Sim
TDGA	Exp & Sim	Sim	Exp & Sim
SA	Exp & Sim	Exp & Sim	Sim
HCA	Sim	Sim	Sim

antenna, step-attenuator or phase shifter) that occur in such a way that either a short-circuit or an open-circuit exists between the antenna and summing port. Semi-hard faults represent faults that occur and cause a non-controlled impedance state (between antenna and summing port) that is neither a RF open-circuit or a RF short-circuit. Soft faults are stuck-at-previous setting faults that can be caused by a damaged control cable where the pre-fault setting is held in the device's memory despite the issuance of updated commands by the algorithm.

6.2 Recovery in the Presence of Static Signals

Hardware faults that occur in the presence of static signals are the easiest to detect especially after the algorithm converges to a steady state. Hard and semi-hard hardware faults are emulated as the algorithms nears convergence. Hard faults include step-attenuators and phase shifters set to their maximum values, and semi-hard faults include hardware that are set to values not chosen by the algorithm. These faults

Table 6.2: Types of hardware faults emulated and investigated.

Fault	**Classification**	**In-field Equivalent**	**Detectable Situation**
2 step attenuators set to max (31 dB)	Hard	Attenuator fault or disconnected / broken antenna	Static or Mobile Signals
1 step attenuator stuck-at-random setting	Semi-hard	Attenuator fault or damaged antenna	Static or Mobile Signals
1 step attenuator stuck-at-previous setting	Soft	Attenuator fault	Mobile Signals
1 phase shifter stuck-at-random setting	Semi-hard	Phase shifter fault	Static or Mobile Signals
1 phase shifter stuck-at-previous setting	Soft	Phase shifter fault	Mobile Signals

render all solutions found by the algorithm invalid. The results discussed in this section are an extension of the single step-attenuator hard fault recovery discussed in [82] in which it is demonstrated that the TDGA can recover from a hardware fault.

6.2.1 One Static SOI and Two Static Jammers

As discussed in Chapter 5, one SOI and two jammers present the algorithms with one degree of freedom (DOF) because the number of antennas in the array outnumber the number of signals by one. However, the addition of a hardware fault in the worst case can decrease the DOF to zero by effectively removing an antenna from the array. This is true in the case of two step-attenuators in the same path set to 31.0 dB which emulates an antenna fault where the antenna is completely removed from the

beamforming array.

The performance curves for 30 independent simulations of the SGA recovering from a 2 step-attenuator hard fault is shown in Figure 6.1. The best case Hamming distance and azimuth plots are shown in Figures 6.2 and 6.3.

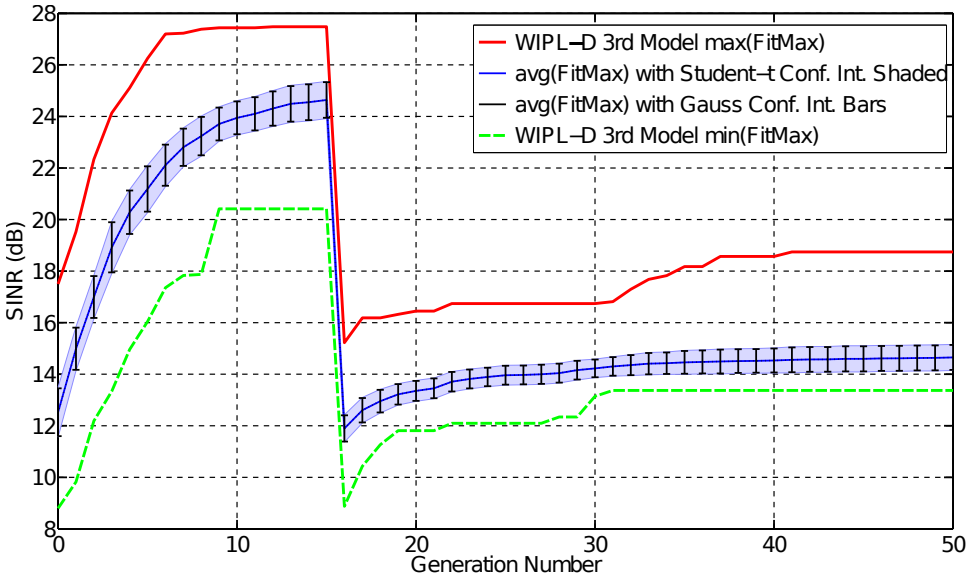


Figure 6.1: SGA simulated performance curves collected over 30 independent runs with SOI at 0° and two jammers at 45° and 200°. Emulated antenna fault (2 step attenuators in one path set to 31.0 dB) at generation 16.

The performance graphs show that the SGA recovers partially from the hard fault with roughly 3 dB SINR recovery after the fault in the best case. The Hamming distance plots for the best solution show an increase in the mean Hamming distance from generation 16 until generation 34 with a spike in both the mean distance and standard deviations that occurred between Generations 31 and 35. This spike corresponds with a 2 dB SINR increase in the best-case solution. The Hamming mean and standard-deviation distances level off after generation 35 corresponding with a

converged state despite attempts by the SGA's mutation operator to find better solutions. The azimuth plots in Figure 6.3 also show partial recovery in the best case solution. Although the SOI gain losses 2 dB during the recovery, the SGA recovers at least 3 dB in the jammer null depths.

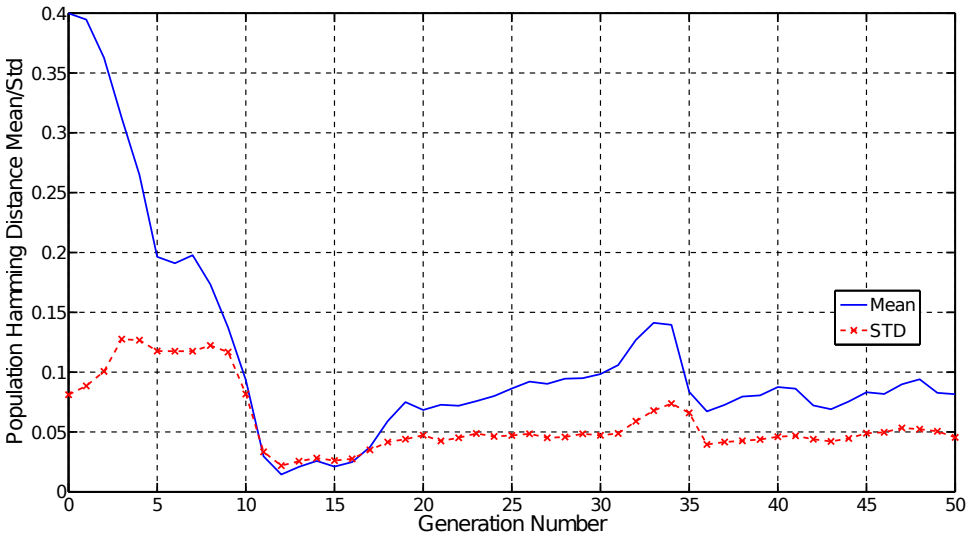


Figure 6.2: Best-case SGA simulated recovery Hamming distance from emulated 2 step attenuator hard fault seen in 30 independent runs with SOI at 0° and two jammers at 45° and 200°. Emulated fault set at generation 16.

The TDGA's performance is also evaluated in recovering from an emulated double step-attenuator hard fault. Two step-attenuators are set to 31 dB despite the setting chosen by the TDGA, and this emulates a fault in which an antenna is effectively removed from the array. The resulting performance curves are shown in Figure 6.4, and the Hamming distances and azimuth plots associated with the best solution found by the TDGA are shown in Figures 6.5 and 6.6. These results indicate that the TDGA is unable to recover from such a fault. However, the simulations give conservative results as discussed in Chapter 5.

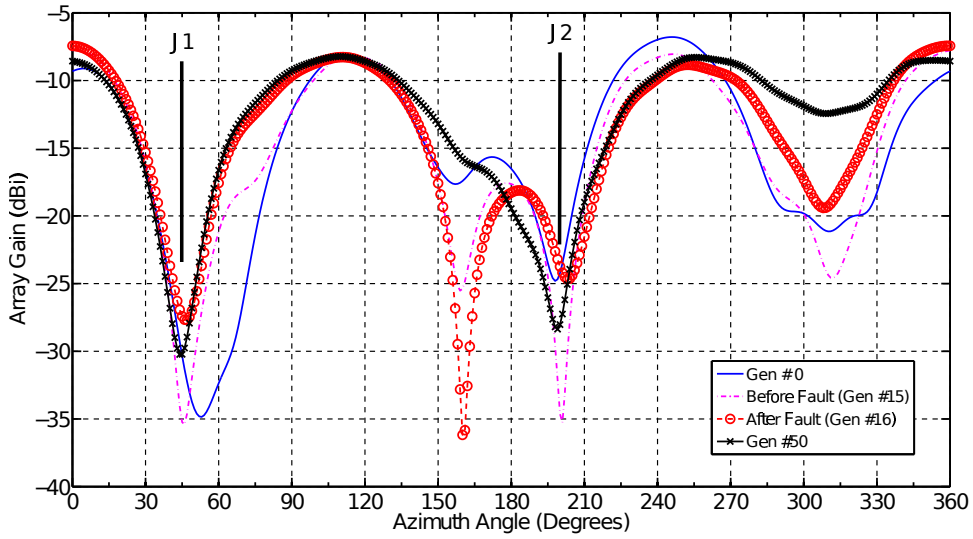


Figure 6.3: Best-case SGA simulated recovery Azimuth patterns from emulated 2 step attenuator hard fault seen in 30 independent runs with SOI at 0° and two jammers at 45° and 200°. Emulated fault set at generation 16.

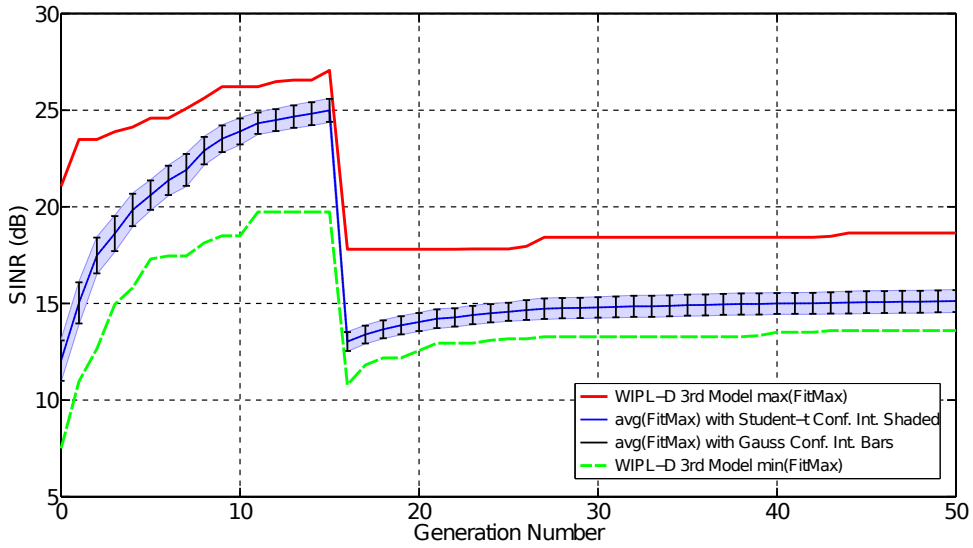


Figure 6.4: TDGA simulated performance curves collected over 30 independent runs with SOI at 0° and two jammers at 45° and 200°. Emulated antenna fault (2 step attenuators in one path set to 31.0 dB) at generation 16.

Fifteen independent runs of the TDGA are run in the anechoic chamber, and the array is optimized to recover from the double step-attenuator fault. As can be seen in Figure 6.7, the TDGA performs much better at fault recovery *in-situ* compared to

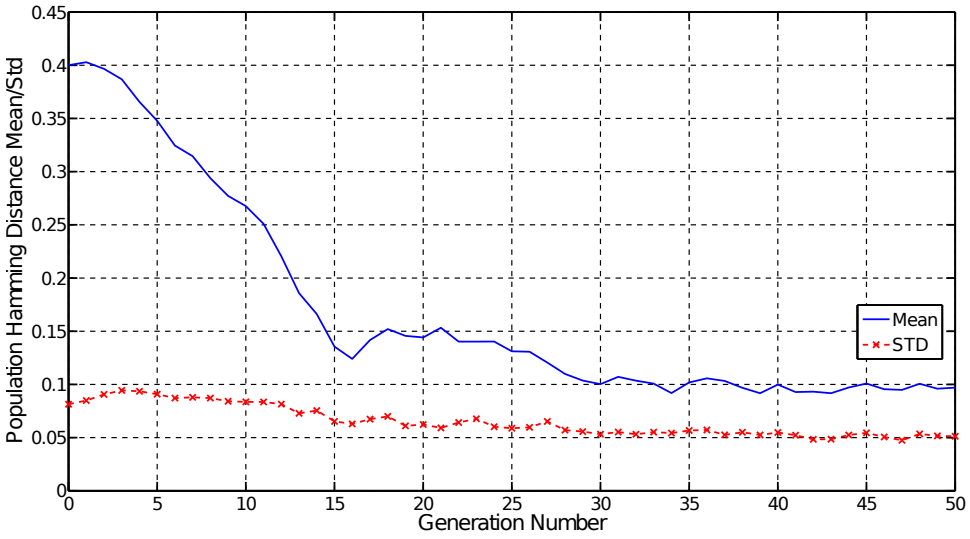


Figure 6.5: Best-case TDGA simulated recovery Hamming distance from emulated 2 step attenuator hard fault seen in 30 independent runs with SOI at 0° and two jammers at 45° and 200°. Emulated fault set at generation 16.

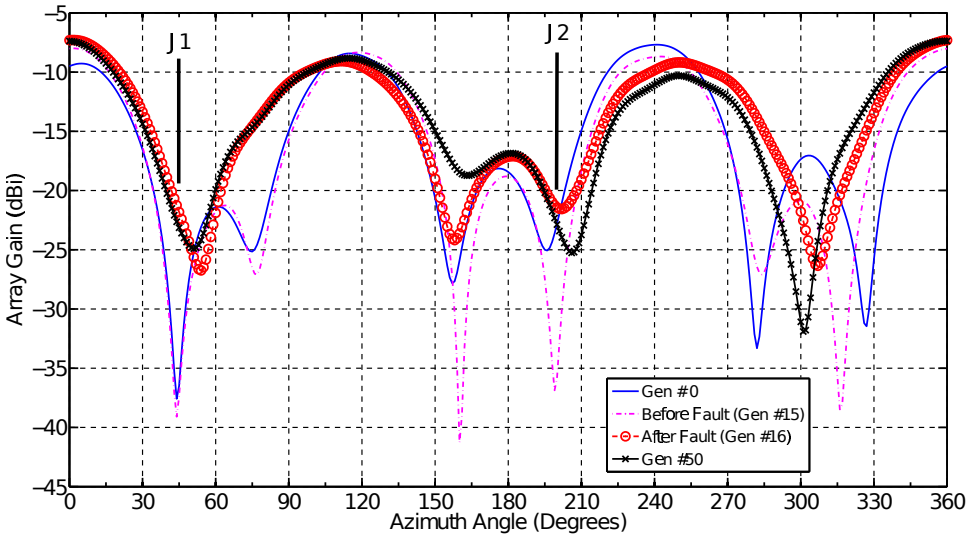


Figure 6.6: Best-case TDGA simulated recovery azimuth patterns from emulated 2 step attenuator hard fault seen in 30 independent runs with SOI at 0° and two jammers at 45° and 200°. Emulated fault set at generation 16.

the simulation results. It obtains a best-case (maximum out of the maxima from 15 runs) SINR of 47.0 dB by generation 50. Although the worst of the best 15 solutions is close to 1 dB SINR, the 95% confidence interval by generation 50 predicted using

the Student-t distribution is ±7.37 dB around the 20.76 dB mean SINR of the 15 best solutions. The Student-t based confidence interval is a few decibels higher than the ±4.77 dB 95% confidence interval calculated assuming a Gaussian distribution. This is expected because a small sample size is used, but the sample size is large enough to produce reliable Student-t confidence interval calculations [70].

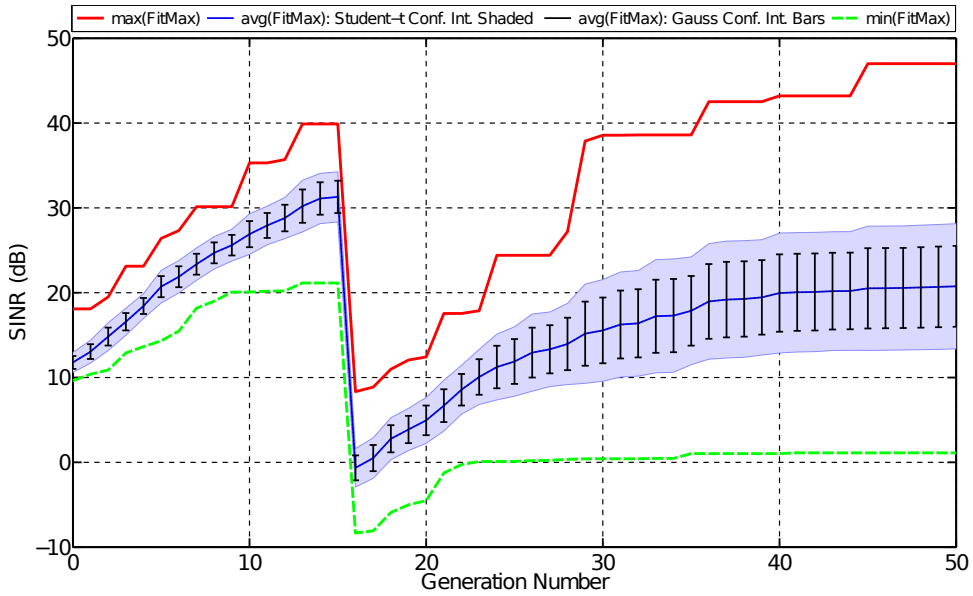


Figure 6.7: TDGA *in-situ* performance curves collected over 15 independent runs with SOI at 0° and two jammers at 45° and 200°. Emulated antenna fault (2 step attenuators in one path set to 31.0 dB) at generation 16.

The mean and standard-deviation Hamming distance for the best *in-situ* solution found by the TDGA in 15 runs is shown in Figure 6.8. After the emulated fault occurs at generation 16, the standard deviation increases slightly, but the mean Hamming distance increases to above 0.15 and stays at that level until around generation 33 where it begins to linearly decrease. Comparing the Hamming distance to the performance plots in Figure 6.7, it can be seen that this corresponds to a re-optimization

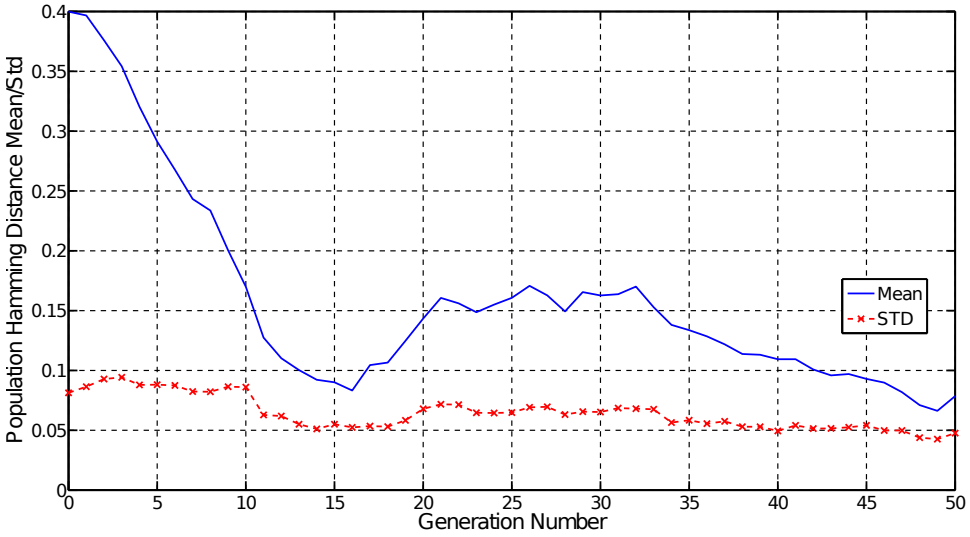


Figure 6.8: Best-case TDGA *in-situ* recovery Hamming distance from emulated 2 step attenuator hard fault seen in 15 independent runs with SOI at 0° and two jammers at 45° and 200°. Emulated fault set at generation 16.

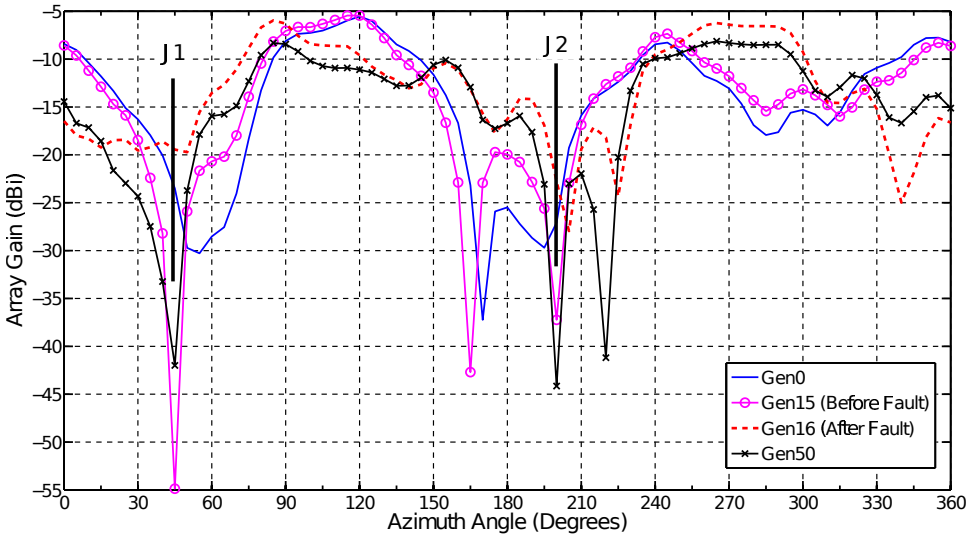


Figure 6.9: Best-case TDGA *in-situ* recovery azimuth patterns from emulated 2 step attenuator hard fault seen in 15 independent runs with SOI at 0° and two jammers at 45° and 200°. Emulated fault set at generation 16.

of the array settings to compensate for the fault.

The azimuth radiation patterns for the best *in-situ* solutions are shown in Figure

6.9. The patterns are consistent with the maximum of the best SINR performance curve shown in Figure 6.7 because the TDGA found a solution in generation 15 that focus energy in the SOI's direction and null the jammers at 45° and 200°. The solution is invalidated at generation 16 since the SOI gain dropped 5 dB, the null at 45° is removed, and the null at 200° is decreased by almost 15 dB. Although the TDGA recovered roughly 2 dB of SOI gain by generation 50, it repositioned nulls at 45° and 200° with the second null having more than 5 dB depth compared to the generation 15 solution.

The performance curves for 30 independent simulations of the SA algorithm in performing hardware fault recovery in the case of a double hard step-attenuator fault are shown in Figure 6.10. The temperature schedule repeats five times over 10,200 evaluations. Because SA convergs as the temperature schedule reached 0% mutation probability, the schedule needs to be repeated in order for SA to re-optimize the array settings such that they compensat for the fault. The fault is set to occur much later at evaluation 6,200 (GA equivalent generation 31) because it is shown in Chapter 5 that SA converges on average much slower than the SGA and TDGA. This represents a fallback to the SA algorithm, as SA cannot compensate for a fault if the fault occurred after the temperature schedule reach 0% mutation probability.

The azimuth radiation plots for the best solution found by SA are shown in Figure 6.10. By evaluation 6,200, SA steers null in both jammers' directions. It loses both nulls after the fault occurred at evaluation 6,201, but by evaluation 10,200, SA

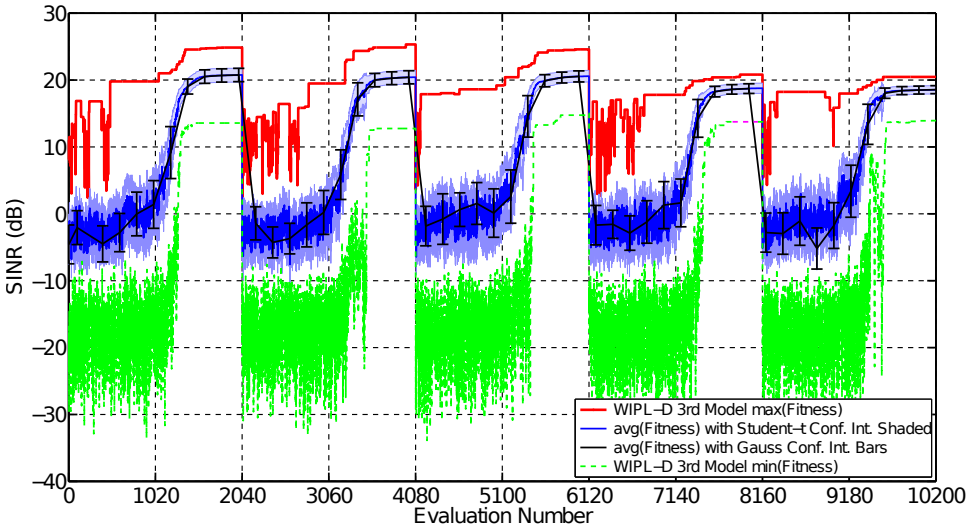


Figure 6.10: SA simulated performance curves collected over 30 independent runs with SOI at 0° and two jammers at 45° and 200°. Emulated antenna fault (2 step attenuators in one path set to 31.0 dB) at generation 16.

recovers the nulls that it previously steered towards both jammers. The array gain towards the SOI is not affected by the emulated loss of an antenna in this case.

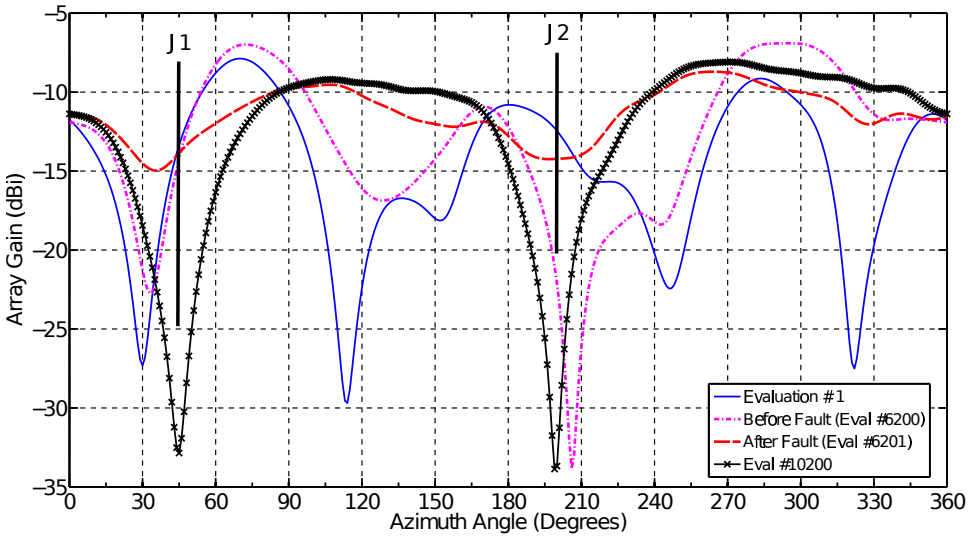


Figure 6.11: Best-case SA simulated recovery azimuth patterns from emulated 2 step attenuator hard fault seen in 30 independent runs with SOI at 0° and two jammers at 45° and 200°. Emulated fault set at generation 16.

The performance curves for HCA anti-jamming and recovering from a dual hard

step-attenuator fault are shown in Figure 6.12. HCA is also capable of performing these dual functions like SA. Also like SA, HCA is temperature schedule dependent because HCA is basically SA with the Metropolis condition turned off, so HCA would not be able to recover from a hardware fault if it occurs after HCA converged based on the temperature scheduling setting the probability of mutation close to 0%. However, as can be seen in Figures 6.10 and 6.12, both SA and HCA can perform hardware fault recovery if the temperature schedule is repeated ad infinitum.

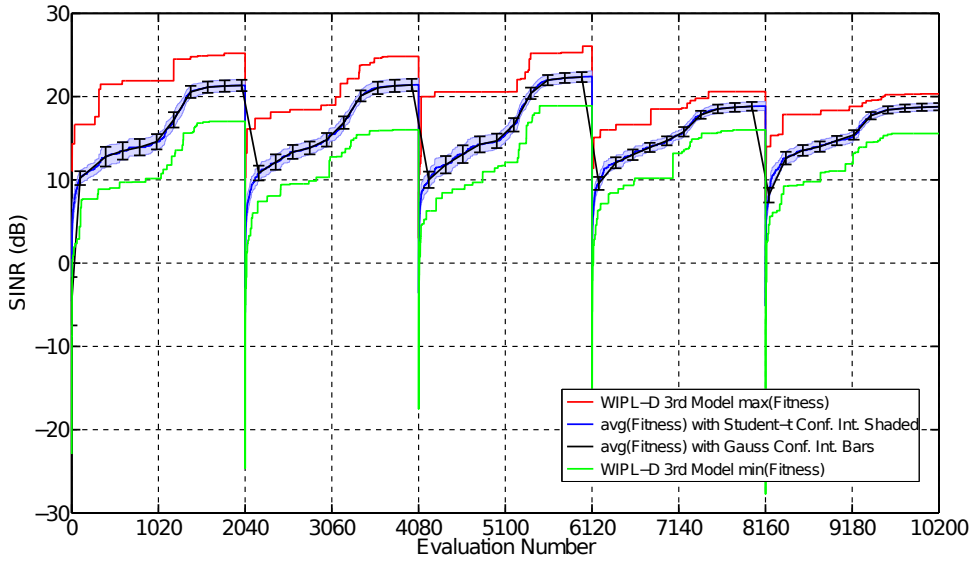


Figure 6.12: HCA simulated performance curves collected over 30 independent runs with SOI at 0° and two jammers at 45° and 200°. Emulated antenna fault (2 step attenuators in one path set to 31.0 dB) at generation 16.

The tradeoff in repeating the temperature schedule is in the quality of the solution found versus the HCA's and SA's reaction time to a fault, as it is shown that both algorithms are much slower than the SGA and TDGA. It is also shown in Chapter 5 that HCA gets stuck in local optima, and this behavior is visible in Figure 6.12. It is feasible that if the temperature schedule repeat factor is changed to 10 times

per 10,200 evaluations, HCA would on average get stuck around 15 dB SINR instead of reaching SINRs greater than 20 dB on average before being reset. Although SA does not get stuck on local maxima, it spends roughly half of each temperature schedule period on average before exploiting the search space (see Figure 6.11), so it is reasonable to expect SA to find solutions with lower SINR value if the temperature schedule repeat factor is doubled.

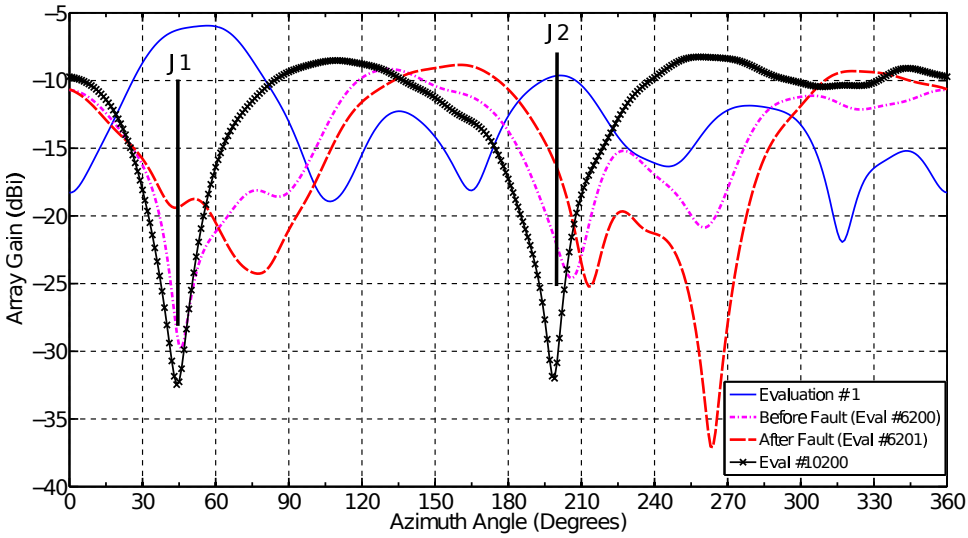


Figure 6.13: Best-case HCA simulated recovery azimuth patterns from emulated 2 step attenuator hard fault seen in 30 independent runs with SOI at 0° and two jammers at 45° and 200°. Emulated fault set at generation 16.

The best case HCA azimuth radiation patterns with recovery from the fault are shown in Figure 6.13. The fault effectively removes the null directed at jammer 1, and it shifted the null at jammer 2 by roughly 10° such that the null depth towards jammer 2 is increased by roughly 5 dB after the emulated fault occurred. The HCA is able to recover from the fault, and it increased the SOI gain by 1 dB by evaluation 10,200.

6.2.2 One Static SOI and Three Static Jammers

It is not possible for the software to fully recover from a hardware fault if one SOI and three jammers are present. A fault is akin to physically removing or damaging an antenna, and it presents a situation in which more signals are present than fully-functional antennas. However, it is shown below that the algorithms recover some SINR lost after a double step-attenuator hard fault (i.e., antenna physically removed) occurs. The SGA, TDGA, SA, and HCA are evaluated in this case with the SOI at 0° and three jammers located at 45°, 200°, and 300°in the azimuth plane.

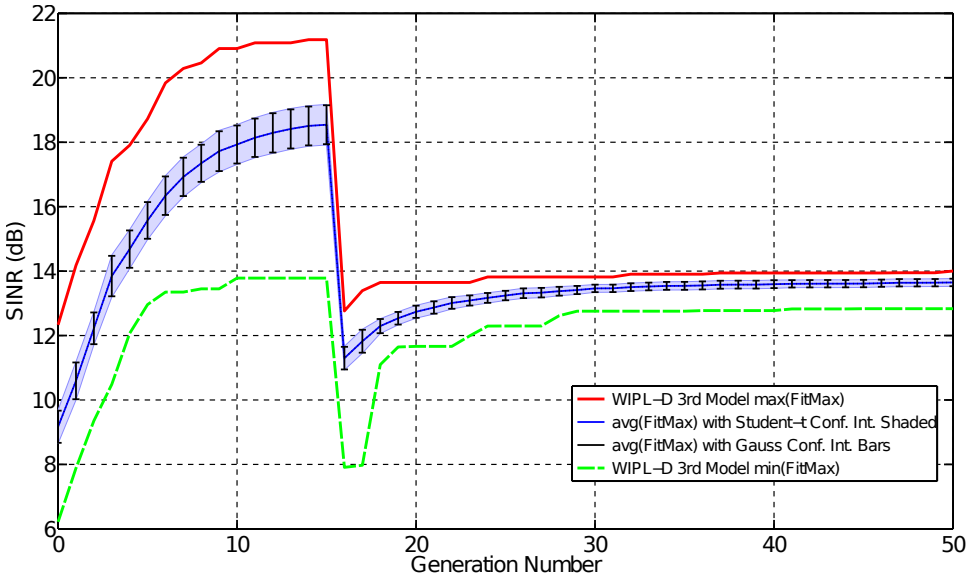


Figure 6.14: SGA simulated performance curves collected over 30 independent runs with SOI at 0° and three jammers at 45°, 200°, and 300°. Emulated antenna fault (2 step attenuators in one path set to 31.0 dB) at generation 16.

The performance curves for the SGA are shown in Figure 6.14. In the best case, the SGA recovers 1.24 dB SINR after the fault and obtains a final maximum SINR of 14.00 dB. The recovery in the worst case (worst of the 30 best solutions) is 4.92

dB SINR recovered with a final 12.83 dB SINR. Although positive SINR values are good, the calculations indicate that roughly 25 dB SINR is needed to ensure that all three jammers are nulled at least 20 dB below the SOI.

The resulting Hamming distance and azimuth radiation plots for the best solution found by the SGA are shown in Figures 6.15 and 6.16. There is a slight increase in the mean Hamming distance after the emulated fault occurred, but this is not enough diversity in the population to fully recover from the fault. Although the SGA shows improvements in the nulls directed at jammers 1 and 3 by generation 50 (see Figure 6.16), it decreases the null depth directed at jammer 1. It happens that the best solution found directly after the fault occurs improves the null depth directed at jammer 1, and the recovery by generation 50 serves to sacrifice jammer 1 attenuation while maximizing attenuation directed at jammers 2 and 3.

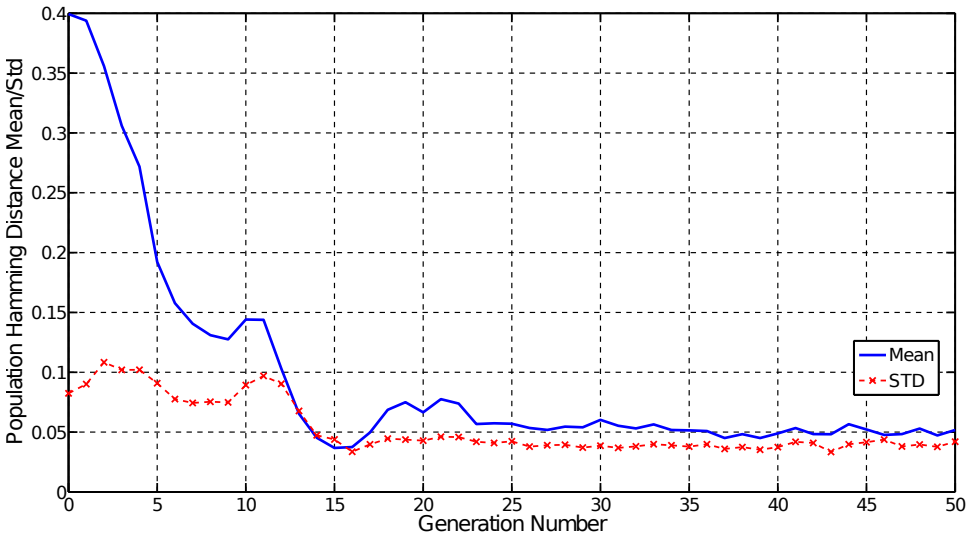


Figure 6.15: SGA simulated best-case Hamming distance plots with SOI at 0° and three jammers at 45°, 200°, and 300°. Emulated antenna fault (2 step attenuators in one path set to 31.0 dB) at generation 16.

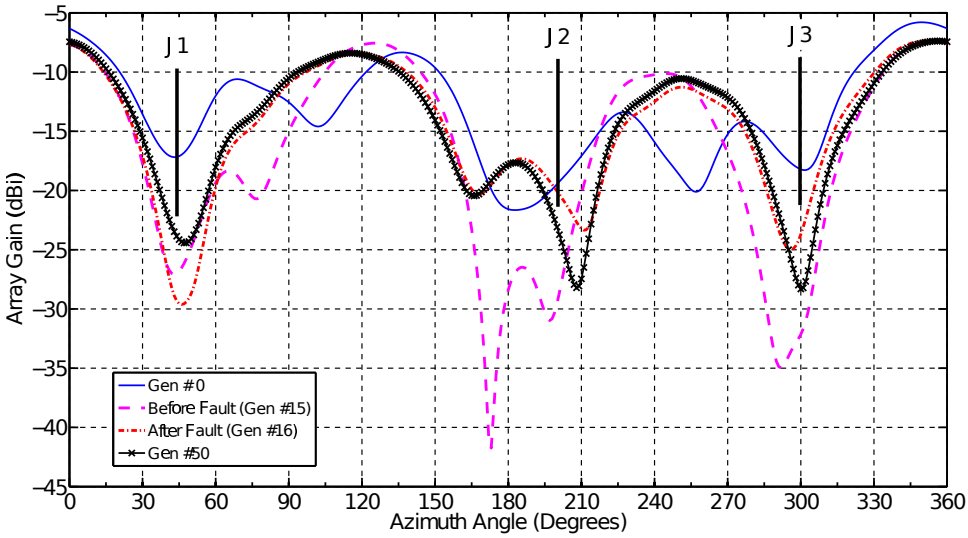


Figure 6.16: SGA simulated best-case azimuth radiation plots with SOI at 0° and three jammers at 45°, 200°, and 300°. Emulated antenna fault (2 step attenuators in one path set to 31.0 dB) at generation 16.

The performance curves for the best 30 solutions found by the TDGA in simulation are shown in Figure 6.17. Like the SGA, the TDGA in the best case recovers 1.11 dB SINR to achieve a final 14.05 dB SINR after the emulated fault occurs. In the worst case, the TDGA recovers 3.15 dB SINR to achieve a final worst (of the best) case 13.07 dB SINR. Both values are slightly higher compared to the SGA, but the recovery is still insufficient to achieve jammer array gains that are at least 20 dB down from the SOI (see Figure 6.18).

The best case Hamming distance and azimuth radiation plots for the simulated TDGA are shown in Figures 6.17 and 6.18. Unlike the SGA, the TDGA's best-case mean Hamming distance curve almost doubles within a few generations after the emulated fault occurred, and the mean Hamming distance remains above 0.125 for the remainder of trial. However, the standard deviation of population Hamming

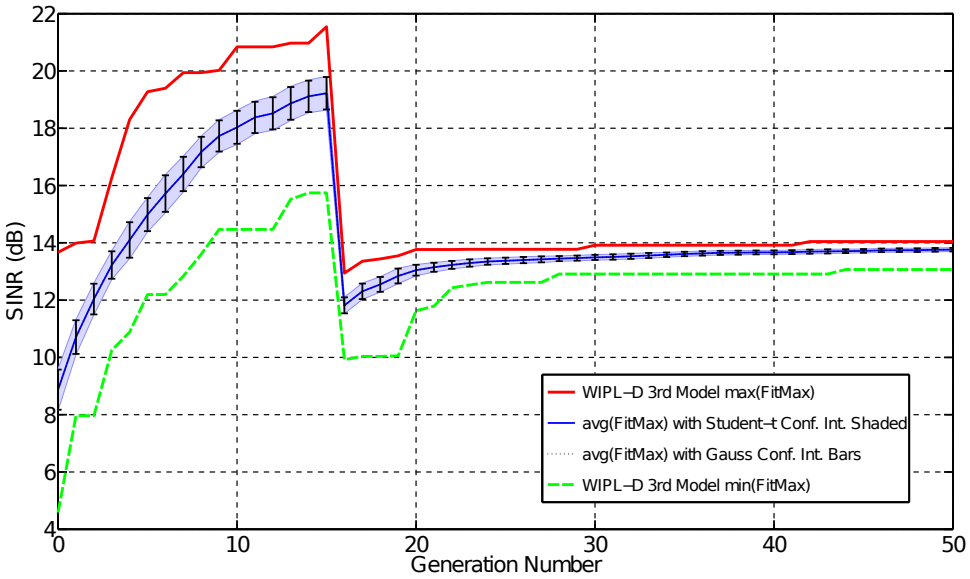


Figure 6.17: TDGA simulated performance curves collected over 30 independent runs with SOI at 0° and three jammers at 45°, 200°, and 300°. Emulated antenna fault (2 step attenuators in one path set to 31.0 dB) at generation 16.

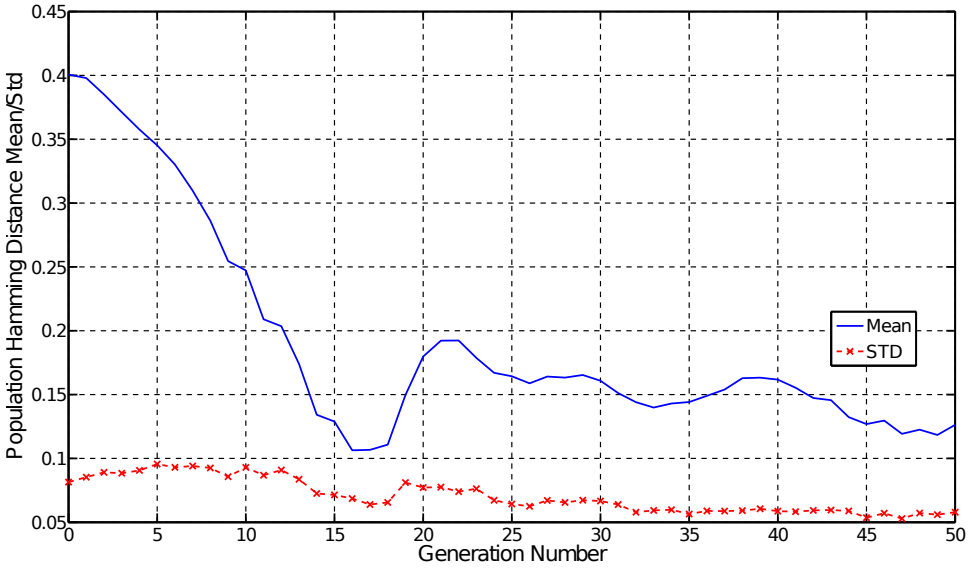


Figure 6.18: TDGA simulated best-case Hamming distance plots with SOI at 0° and three jammers at 45°, 200°, and 300°. Emulated antenna fault (2 step attenuators in one path set to 31.0 dB) at generation 16.

distance remains low. This means that the TDGA is ineffective in its search to find a good solution that recovers from the fault. It is visible in Figure 6.18 that only

the null depth directed at jammer 3 is significantly improved by generation 50. The other nulls remain roughly the same after the fault.

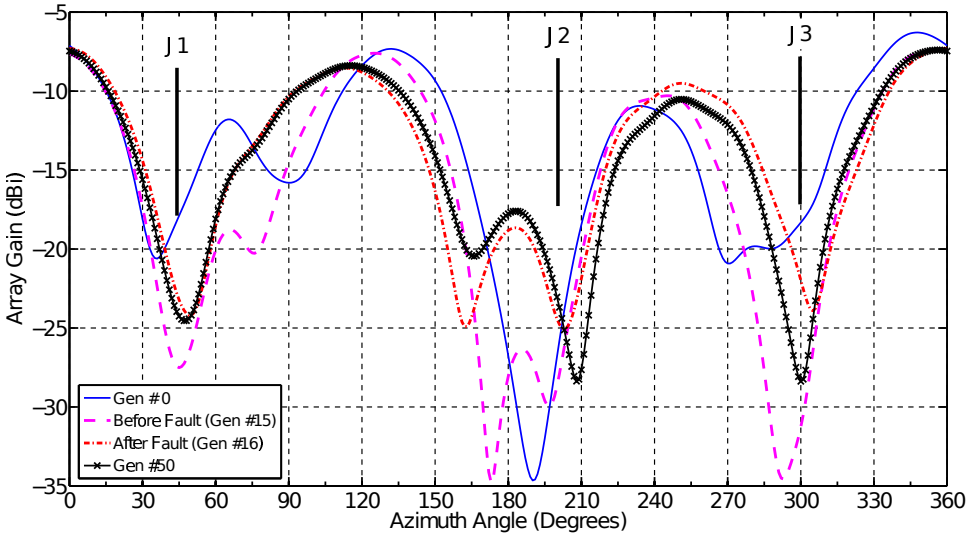


Figure 6.19: TDGA simulated best-case azimuth radiation plots with SOI at 0° and three jammers at 45°, 200°, and 300°. Emulated antenna fault (2 step attenuators in one path set to 31.0 dB) at generation 16.

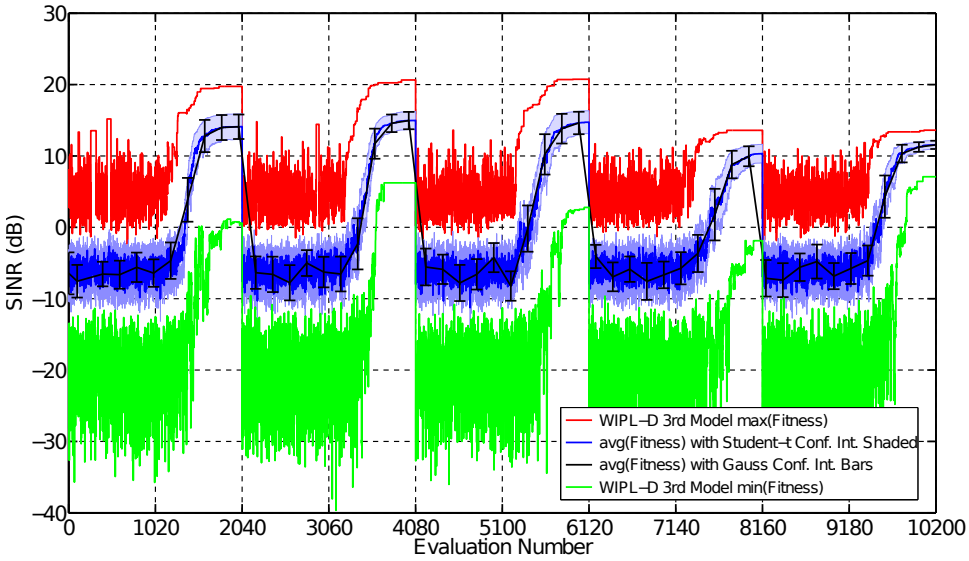


Figure 6.20: SA simulated performance curves collected over 30 independent runs with SOI at 0° and three jammers at 45°, 200°, and 300°. Emulated antenna fault (2 step attenuators in one path set to 31.0 dB) at evaluation 6,201.

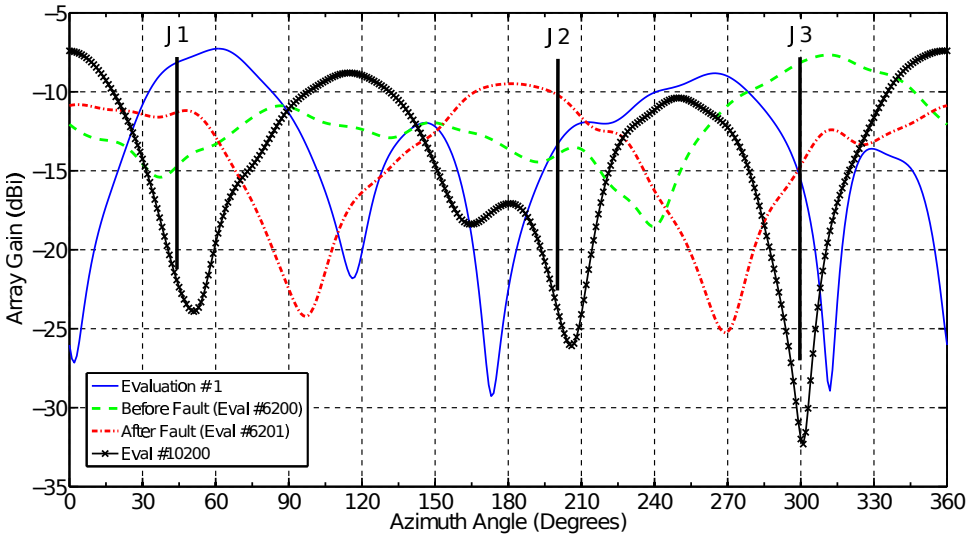


Figure 6.21: SA simulated best-case azimuth radiation plots with SOI at 0° and three jammers at 45°, 200°, and 300°. Emulated antenna fault (2 step attenuators in one path set to 31.0 dB) at evaluation 6,201.

The simulated annealing algorithm's performance graphs are shown in Figure 6.20. The best solution that SA found before the emulated fault occurs has a maximum SINR of 20.75 dB at evaluation 6,120 before the temperature schedule reset itself. The maximum SINR at evaluation 10,200 is 13.61 dB. It should be noted that the azimuth plots at evaluations 6,200 and 6,201 shown in Figure 6.21 have SINRs of 4.39 dB and 4.80 dB which are not significantly different. Because SA is in the early stage of a repeated temperature schedule when the emulated fault occur, it is clear that both the fault and the algorithm's search state affected the algorithm's performance at evaluation 6,201.

Finally, the performance graphs for 30 independent HCA simulations and the best-case azimuth radiation plots are shown in Figures 6.22 and 6.23. The HCA's performance behavior (i.e., SINR vs. evaluation number) is smoother compared to SA

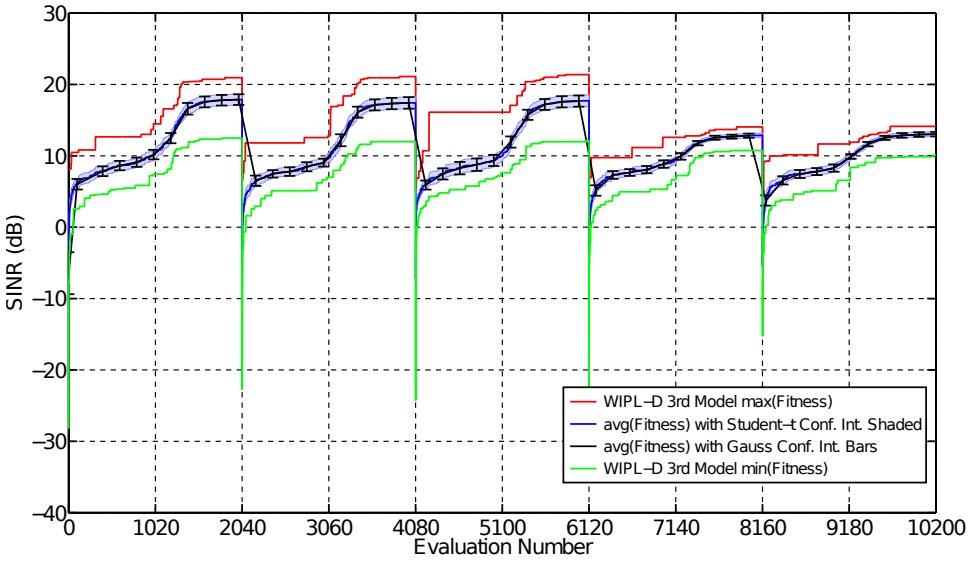


Figure 6.22: HCA simulated performance curves collected over 30 independent runs with SOI at 0° and three jammers at 45°, 200°, and 300°. Emulated antenna fault (2 step attenuators in one path set to 31.0 dB) at evaluation 6,201.

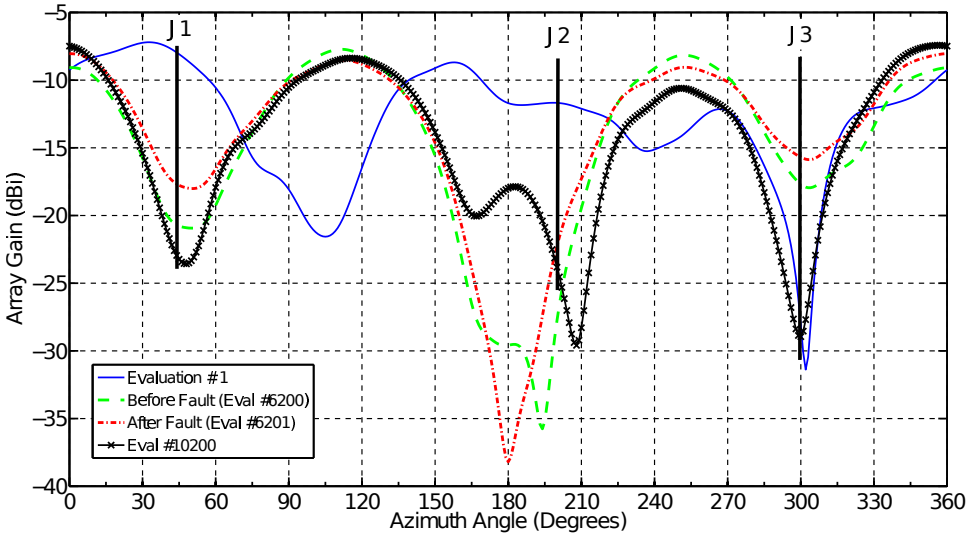


Figure 6.23: HCA simulated best-case azimuth radiation plots with SOI at 0° and three jammers at 45°, 200°, and 300°. Emulated antenna fault (2 step attenuators in one path set to 31.0 dB) at evaluation 6,201.

since HCA uses the same repeated temperature schedule with the Metropolis condition turned off. HCA achieves a maximum 21.38 dB SINR at evaluation 6,120 which

is the last full temperature cycle before the emulated fault occurs. The maximum SINR that HCA achieves by evaluation 10,200 is 14.13 dB. This is slightly higher than SA.

It can be observed from the azimuth plots shown in Figure 6.23 that the best solution found by the HCA recovers from the emulated fault.T here are significant improvements in the null depths directed at jammers 1 and 3 although the null depth directed towards jammer 2 do not show significant improvement.

6.3 Recovery in the Presence of Mobile Signals

In this section, we show that the algorithms being studied can recover from hardware faults in the presence of mobile signals, but the faults are difficult to detect because they are masked. The experiments include stuck-at-previous setting faults that are difficult to detect when the signals present are static. Stuck-at-previous setting faults have the property of invalidating the current best solution only if the signals change direction. For example, if the stuck-at fault occurs after the algorithm converges, it would be undetectable because there is no pressure on the algorithm to find another solution.

The performance curves for the best 30 solutions that the SGA found in 30 independent runs are plotted in Figure 6.25. Despite the occurrence of a stuck-at attenuator fault at generation 16, the SGA behaves as if no fault had occurred. The best solution at generation 50 has a SINR of over 25 dB which is consistent with

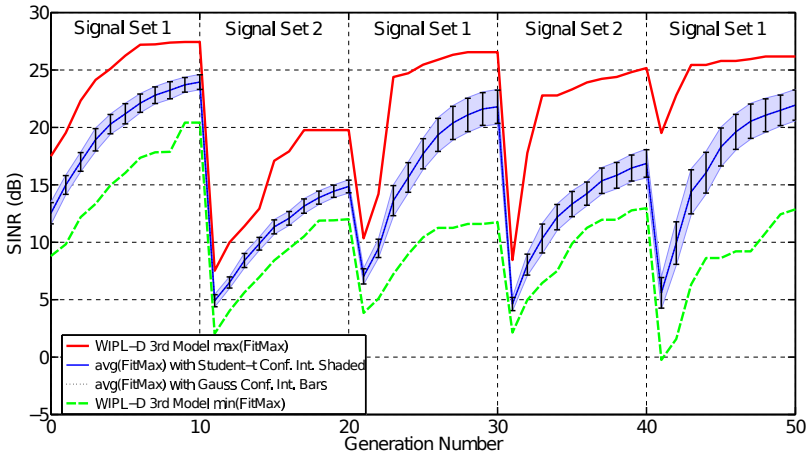


Figure 6.24: SGA simulated performance curves collected over 30 independent runs with SOI at 0° and two mobile jammers. Emulated step-attenuator stuck-at fault emulated at generation 16.

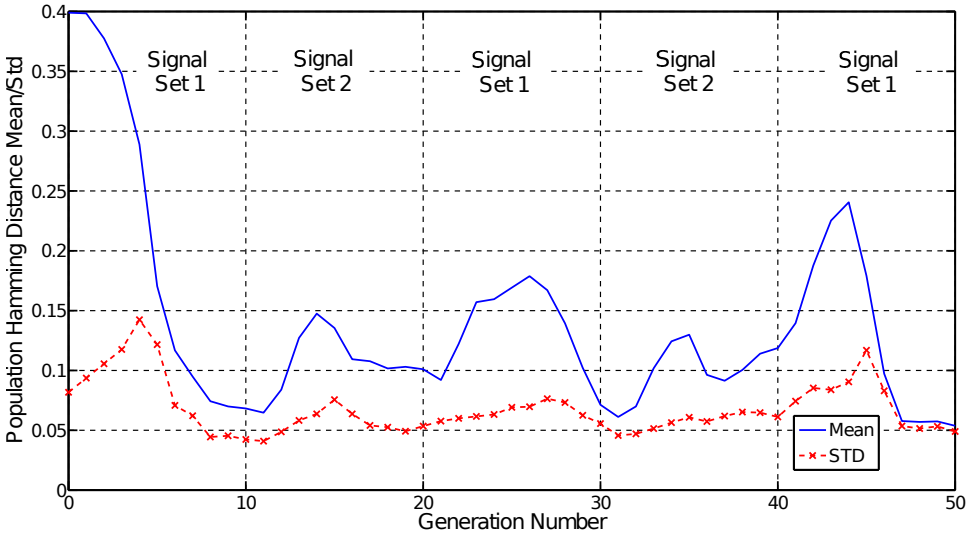


Figure 6.25: SGA simulated Hamming distance plots for best-case solution out of 30 independent runs with SOI at 0° and two mobile jammers. Emulated step-attenuator stuck-at fault emulated at generation 16.

non-faulted runs of the SGA. The Hamming distance plots for the best out of the 30 best solutions shown in Figure 6.26 also behave as if nothing had happened. The SGA increases its population diversity to account for both signal stepped mobility

and the stuck-at fault.

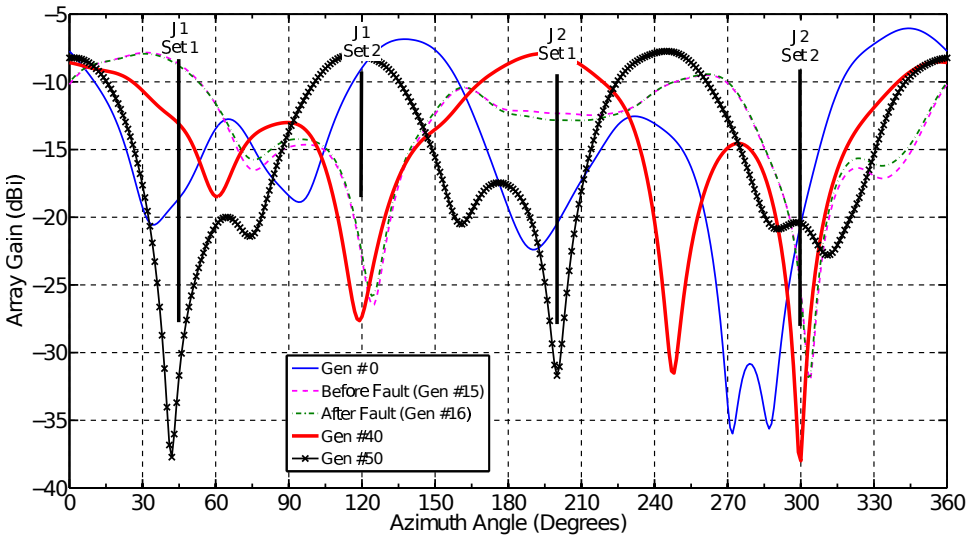


Figure 6.26: SGA simulated azimuth radiation plots for best-case solution out of 30 independent runs with SOI at 0° and two mobile jammers. Emulated step-attenuator stuck-at fault emulated at generation 16.

The best-case azimuth patterns for the SGA are shown in Figure 6.26. As expected, the azimuth patterns before and after the stuck-at fault occur are hardly distinguishable from each other. The SGA is also able to adapt to both signal sets by placing nulls at 45° and 200° at generation 40, and the SGA places deep nulls towards the second jammer set at 120° and 300°. Although these simulations are accidentally run for 50 generations instead of the usual 60 generations, the results obtained are still valid and indicative of expected SGA behavior.

The TDGA is also simulated with the same signal sets and an emulated step-attenuator stuck-at previous setting fault that occurs at generation 16. The resulting performance curves for the best 30 solutions found by 30 independent simulations of the TDGA are shown in Figure 6.27.

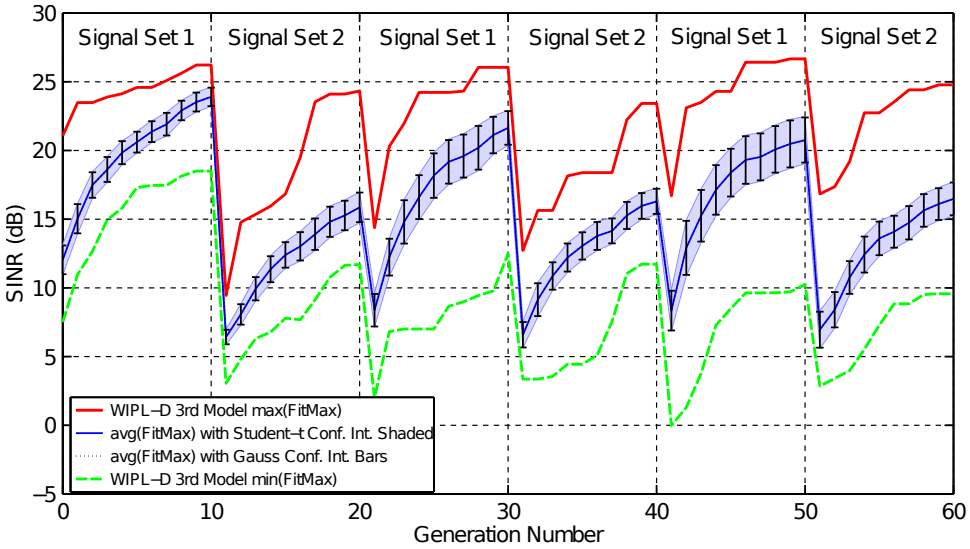


Figure 6.27: TDGA simulated performance curves collected over 30 independent runs with SOI at 0° and two mobile jammers. Emulated step-attenuator stuck-at fault emulated at generation 16.

The simulated TDGA behaves in a manner very similar to the SGA, as the TDGA treats the stuck-at fault like an environment change. The TDGA continues optimizing the array to anti-jam despite the presence of the fault with mobile signals. The TDGA gives similar performance compared to the SGA, as the TDGA maximum SINR curves shown in Figure 6.27 have values of at least 25 dB at convergence with the given signal sets. The minimum (of the best 30) have slightly lower SINRs at 10 dB compared to the SGA which has minimum values around 12 dB SINR.

The Hamming distances for the best out of 30 TDGA solutions are shown in Figure 6.28. The TDGA's Hamming distance behaves in a manner similar to the SGA although the SGA exhibits higher peak mean Hamming distances. Because the TDGA achieves roughly the same performance as the SGA, the higher peaks in the SGA's mean Hamming distances may indicate that the SGA needs to work harder

(i.e., create more diversity) in order for it to compensate for both a stuck-at fault and stepped mobile signals.

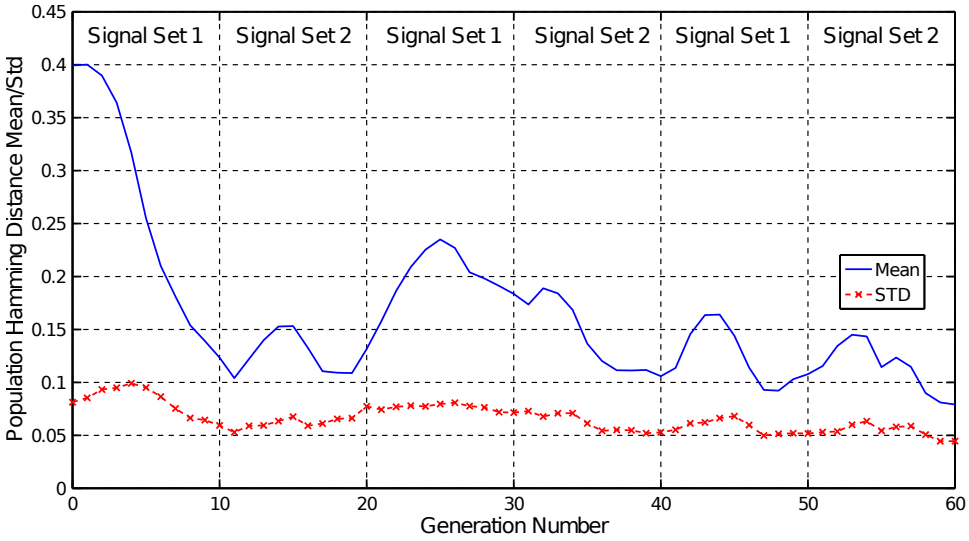


Figure 6.28: TDGA simulated Hamming distance plots for best-case solution out of 30 independent runs with SOI at 0° and two mobile jammers. Emulated step-attenuator stuck-at fault emulated at generation 16.

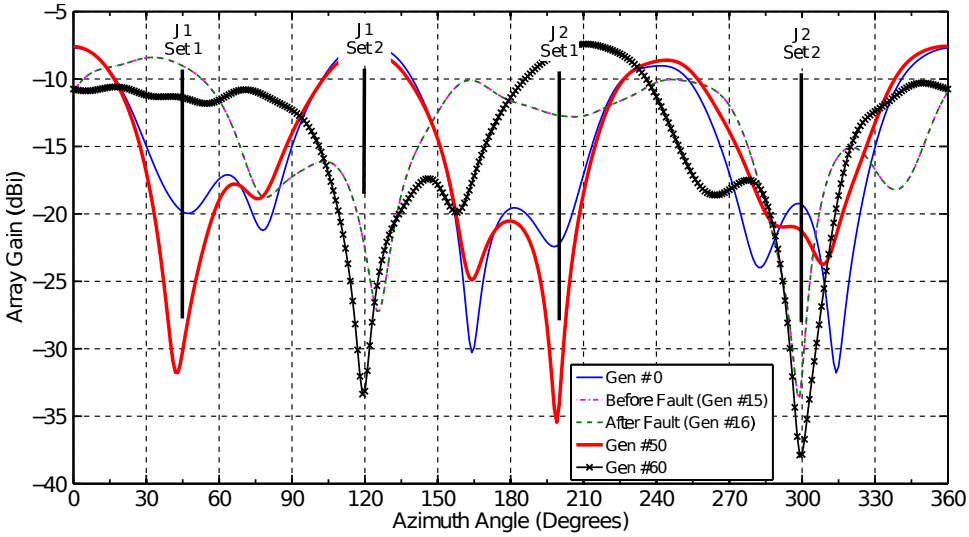


Figure 6.29: TDGA simulated azimuth radiation plots for best-case solution out of 30 independent runs with SOI at 0° and two mobile jammers. Emulated step-attenuator stuck-at fault emulated at generation 16.

The best case azimuth plots for the TDGA simulations are shown in Figure 6.29. The emulated stuck-at previous setting step attenuator fault does not cause the TDGA to behave differently, as the radiation patterns shown at generation 15 (before fault) and at generation 16 (after fault) are virtually the same. The TDGA also places nulls towards the four jammers from the two signal sets combined with null depths comparable to the SGA. The main difference is that the TDGA decreases the array gain towards the SOI at generation 60 by a few decibels compared to the SGA. The TDGA also widens the mainlobe such that the SOI can move towards 90° and not have its power level significantly degraded.

Fifteen independent runs of the TDGA operating *in-situ* with hardware in an anechoic chamber are also performed using the same signal sets and fault type. The resulting performance graph of the 15 best solutions found by the TDGA are shown in Figure 6.30. It is important to note that these performance graphs place bounds on the TDGA's expected performance, and a solution with maximum SINR value for the second signal set does not necessarily represent a solution from the same run for the first signal set. However, the overall performance exhibited by the *in-situ* TDGA is similar to the behavior that the simulated TDGA exhibits. The difference is that the Student-t distribution predicts higher 95% confidence interval bounds than the Gaussian distribution. This is expected due to the small sample size.

The resulting Hamming distances and azimuth plots for the best signal set 2 solutions are shown in Figures 6.31 and 6.32. Unlike the TDGA simulations, the

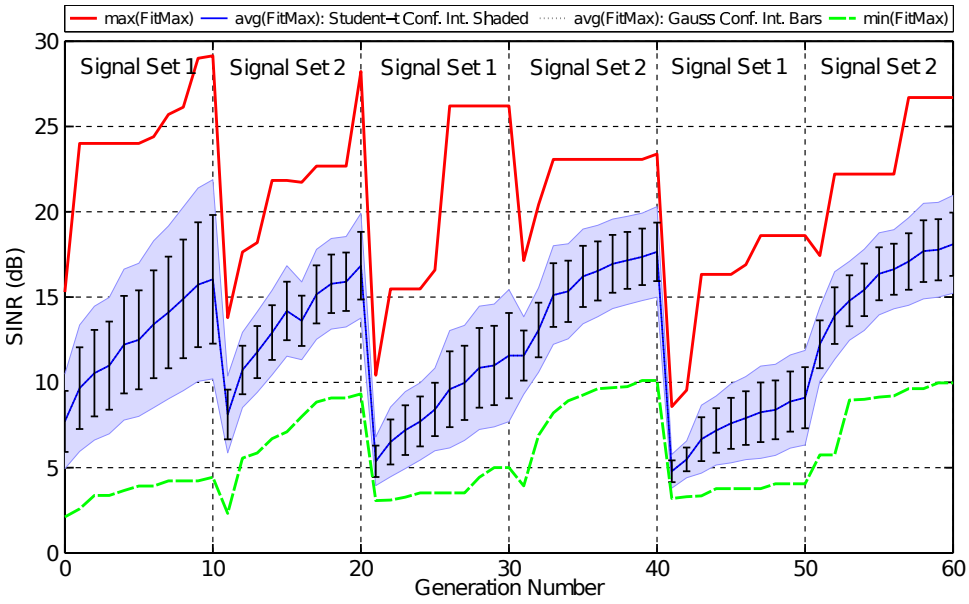


Figure 6.30: TDGA *in-situ* performance curves collected over 15 independent runs with SOI at 0° and two mobile jammers. Emulated step-attenuator stuck-at fault emulated at generation 16.

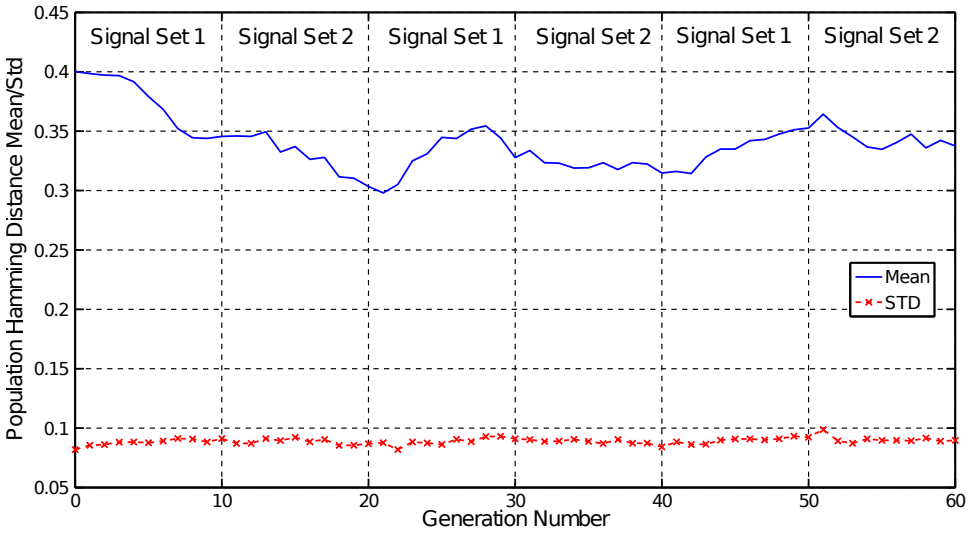


Figure 6.31: TDGA *in-situ* Hamming distance plots for best-case solution out of 15 independent runs with SOI at 0° and two mobile jammers. Emulated step-attenuator stuck-at fault emulated at generation 16.

TDGA *in-situ* maintain a far higher mean Hamming distance versus generation number. There are also differences between the generation 15 (before emulated fault)

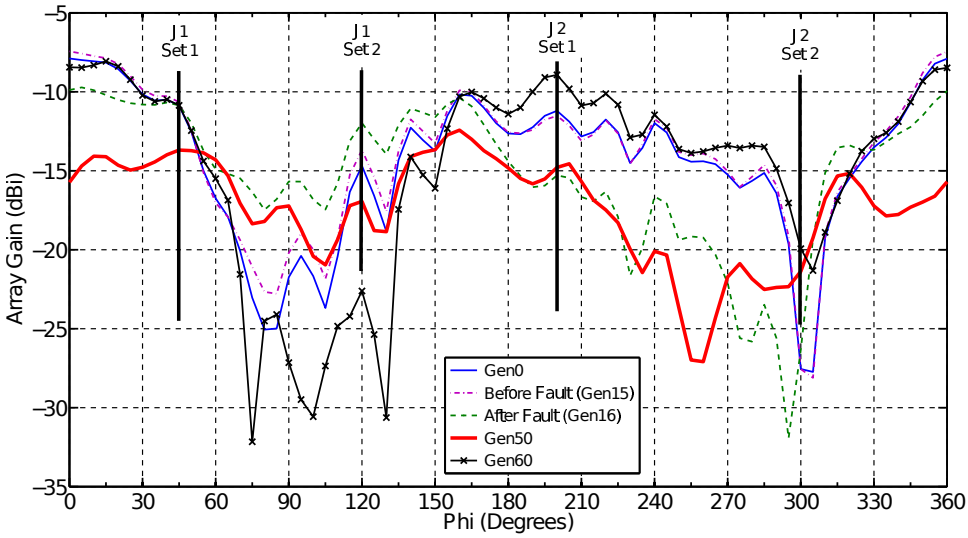


Figure 6.32: TDGA *in-situ* azimuth radiation plots for best-case solution out of 15 independent runs with SOI at 0° and two mobile jammers. Emulated step-attenuator stuck-at fault emulated at generation 16.

and generation 16 (after emulated fault) azimuth plots shown in Figure 6.32. In this run, it is also observed that the TDGA does a better job at steering nulls towards the jammers in the second signal set compared to the jammers in the first signal set. This behavior likely comes from the nature of the small sample size, as the TDGA would have likely found a better solution in future runs that maintain higher SINRs for both signal sets despite the emulated stuck-at fault.

6.4 Summary

In this chapter, it is shown how well stochastic algorithms perform hardware fault recovery in anti-jamming beamforming arrays. The SGA, TDGA, SA, and HCA are compared in simulations and in some cases for the TDGA *in-situ* measurements are

collected and compared. We evaluate these algorithms' abilities in performing fault recovery with both static and mobile signals. The SGA and TDGA automatically detect and recover from faults when the signals are static. The SGA uses mutation in its recovery to reintroduce enough diversity back into the population to allow useful crossovers, and the TDGA uses Triallelic Diploid strings (on top of mutuation) to increase population diversity in performing fault detection and recovery. As is expected, both algorithms automatically recover from stuck-at faults when the signals are mobile, but the faults are undetectable by the observer. Stuck-at faults only affect the array when the signals change directions. Because SGA and TDGA treat faults in the same manner as environmental changes, it is not possible to detect a fault unless it is known for certain that the channel is AWGN (i.e., no fades), and the signals are static.

Although SA and HCA are both able to detect and recover from faults, both algorithms are temperature schedule dependent, so the temperature schedule is repeated. Otherwise, these algorithms get stuck in pre-fault solutions that are no longer valid due to the presence of faults. Due to the noisy nature of SA, it is be difficult to discern a fault from the algorithm's behavior. The temperature schedule repeat factor also means that a lag exists between the time that a fault occurs and when SA or HCA responds to the fault. This means that it would likely be difficult for either algorithm to issue a fault detection signal when the fault occurs.

Chapter 7

Conclusions and Future Work

This chapter concludes this thesis with a summary of our finding and contributions, and it discusses possible future work to further the research in this area. Section 7.1 states our conclusions to the research presented in this thesis, and section 7.2 discusses our future work in advancing anti-jamming beamforming and hardware fault recovery.

7.1 Conclusions

In this thesis, we show that it is possible to design stochastic algorithms that perform anti-jamming beamforming. Namely, the algorithms focus electromagnetic energy on a signal of interest while simultaneously minimizing energy towards multiple interfering signals. The SGA, TDGA, SA, and HCA are compared in performing this capability by maximizing the array's output SINR. It is not only possible for these algorithms to anti-jam when the signals are static, it is possible for them to anti-jam when the signals are mobile. This result is novel because this is the first time that anti-jamming is performed *in-situ* with an antenna array without knowing the signal DOAs *a priori*.

Two types of mobile signals are considered. First, algorithm performance is

measured with stepped-mobile signals. This means that the signals stay static in a set of directions for an 10 generations, change directions instantaneously, and stay static in the new directions for another 10 generations before switching back to the original directions. Second, the case in which signals continually change their directions is simulated. Although it is shown that all four algorithms discussed in this thesis can anti-jam when signals are stepped mobile, this requires time for each algorithm to adapt to the new signal directions. This is a novel result because prior research in anti-jamming assumes that the signals are static, but it is more realistic to assume that signals constantly move instead of moving instantaneously and staying still.

This means that the algorithm does not have time to converge before the signals change direction. However, our simulations show that the SGA and TDGA can adapt to signals whose directions change constantly in time. We show that our simulations have enough fidelity compared to *in-situ* measurements, and our simulations are conservative in predicting algorithm performance. To the best of our knowledge, this is the first time that it has been shown that the evolutionary algorithms can adapt to signals whose directions constantly change.

Finally, it is shown that these algorithms perform hardware fault recovery with both static and stepped mobile signals. Theory is developed on hardware fault detection, recovery, and localization. *In-situ* measurements show that the TDGA can recovery from a single hardware fault *in-situ* with a four antenna array operating at 2.4 GHz. This result is also novel because this is the also the first time that hardware

fault recovery is demonstrated *in-situ* with a four-antenna array. Further research via simulations show that other stochastic algorithms (SGA, SA, and HCA) can perform both anti-jamming and hardware fault recovery. Although the algorithms find solutions with similar SINR values at convergence, simulations and *in-situ* measurements show that the SGA and TDGA are much faster at optimizing the antenna array compared to SA and HCA.

The algorithms are able to recovery from an emulated hardware fault when one SOI and two jammers are present. With static signals, the SGA and TDGA both indicate significant drops in SINR fitness values at generation 16 when the emulated faults occur, and it is possible to detect the faults. Because SA and HCA are slower than SGA and TDGA by a factor of approximately 3.5 in configuring the array for anti-jamming with three-jammers, a fault that occurs at evaluation 3,201 occurs before SA and HCA nears convergence, so this would not provide useful results. Instead, the fault is emulated at evaluation 6,201 (equivalent to GA generation 30) after SA and HCA complete at least one temperature cycle. SA and HCA recover from the emulated fault, but immediate detection is not possible. The simulation results for the TDGA are conservative compared to *in-situ* measurements because our simulations predict that the TDGA would reconfigure the array with lower SINR values at generation 50. When the two jammers are mobile, all of the algorithms recover from the fault. However, the algorithms treat faults in the same manner as mobile signals, so it is not likely to detect the faults when they occurred.

Our results also show that it is not possible for the algorithms to recovery some anti-jamming functionality from faults when three jammers are present. Because the number of signals equaled the number of antennas in the array, there are no degrees of freedom remaining before the faulted occurred. The fault effectively removed an antenna from the array, so more signals are present that operational antennas.

7.2 Future Work

This section discusses possible directions in which this research can be further developed. Possible directions for future work focus on the following areas: Further hardware development of a larger modular array, real-time anti-jamming beamforming, addition of signal direction finding and adversarial capabilities, and development of fault detection and localization algorithms.

7.2.1 Real-Time Anti-Jamming Beamforming Capabilities

The the *in-situ* measurements and Matlab simulations discussed in this thesis are based on results obtained through an anechoic chamber, and it is assumed that the channel is AWGN. These results are novel because this is the first attempt to the best of our knowledge at optimizing a four-antenna 2.4 GHz beamforming array inside an anechoic chamber, and it is shows that both anti-jamming and hardware fault recovery can be achieved using stochastic algorithms. However, these methods assume that the signal power levels can be calculated and classified using a spectral estimation algorithm (such as the Multiple Signal Classification, MUSIC, algorithm).

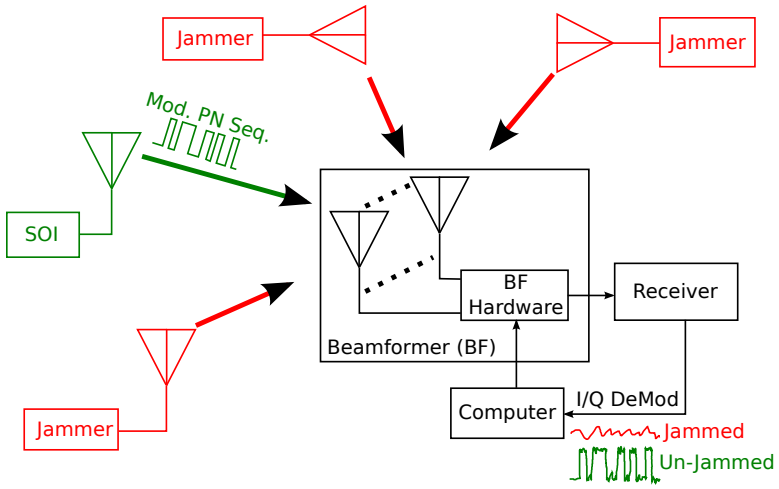


Figure 7.1: High-level block diagram of beamforming system using demodulated PN sequence from SOI to perform anti-jam beamforming.

The discussion in Chapter 3 assumes that the beamforming receiver is able to calculate signal power levels and classify signal types between SOI and interference. Signal classification requires that SOI and interference possess significantly different properties such as power spectral densities. If the SOI and a jammer possess similar power spectral densities, it is difficult to classify them and calculate the receiver's SINR. An alternative method is to use a known property of the SOI to optimize the beamformer's settings and mitigate interference. For example, the SOI transmits a unique PN sequence, and the optimization algorithm compared the demodulated signal at the receiver's output with a copy of the sequence as shown in Figure 7.1.

This work expands upon the previous results discussed in Chapter 5 with the development of an analytic model of an antenna array that operates in real-time. This array mitigates interference using a PN sequence generated by a transmitter such as a Universal Software Radio Protocol (USRP) device to identify the SOI as shown in

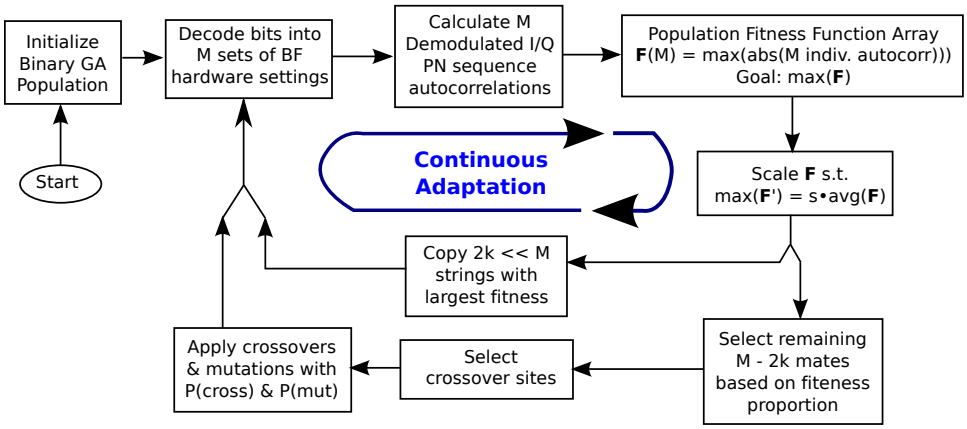


Figure 7.2: Genetic Algorithm (GA) flowchart adapted to operated as part of a USRP wireless link.

Figure 7.1. The beamformer is adapted with a TDGA. In the model, the transmitter sends a pseudo-random noise (PN) sequence I/Q modulated onto a carrier frequency (i.e., 2.4 GHz). The receiver following the beamformer demodulates the received signal and correlates it with a reference PN sequence. Because jammers are present, the demodulated signal resembles noise, and the correlation fails. As the beamforming system adapts to better solutions, the demodulated signal resembles the transmitted PN sequence, and the correlation produces positive results. The system adaptively forms and aims nulls in jammer directions to maximize the correlation, and it is agile because it adapts to time-varying wireless channels and mobile jammers.

The GA applied to this problem uses the received cross-correlation function as its fitness function to be maximized (see Figure 7.2). Because the received correlation function is a high-level measure that uses a demodulated signal after the RF divider/summer, antenna element to receiver path amplitude and phase differences plays a minor role in the overall system performance. This is unlike an interference

cancellation system (ICS) where all internal RF paths need to be amplitude and phase matched to obtain best performance.

If the receiver is jammed, the receiver's demodulated I/Q output in Figure 7.1 resembles noise because the interference exceeds the receiver's spreading gain, and the receiver cannot properly demodulate the SOI's PN sequence. The receiver is unjammed when the receiver's output resembles a noisy copy of the transmitted PN sequence. The beamformer hardware settings are optimized to mitigate interference when this occurs.

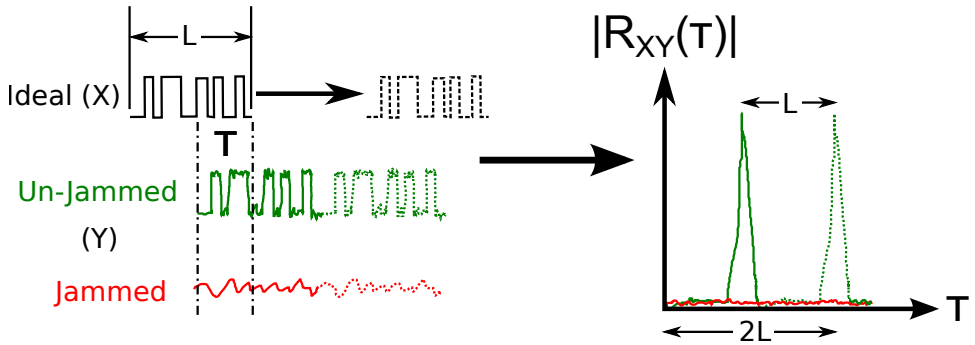


Figure 7.3: A graphical example showing concept of using $\max|R_{XY}(\tau)|$ as a fitness function: The unjammed system created an cross-correlation with series of impulses having spacing L whereas the jammed system resulted in an cross-correlation resembling noise.

The concept of using the maximum of the cross-correlation magnitude ($\max|R_{XY}(\tau)|$) is shown in Figure 7.3. Assuming that the SOI transmits the PN sequence repeatedly, $|R_{XY}(\tau)|$ consists of a series of impulses (i.e., a comb function) when the receiver is unjammed. The length of one cross-correlation sequence is $2L$, but because the PN sequence repeated itself, $\max|R_{XY}(\tau)|$ becomes maximum since the ideal PN sequence aligns with another unjammed sequence when $\tau = 2L$. It is assumed that

Table 7.1: Comparison of SINR vs. PN sequence maximum cross-correlation as fitness functions

SINR		**PN Sequence Max. Cross-correlation**	
Advantages	Disadvantages	Advantages	Disadvantages
Relatively simple implementation	Requires signal classification	Only needs SOI information	Calculation time dependent of PN sequence length
Commonly used metric in the literature	Cannot calculate SINR if SOI and a jammer have similar frequency spectra	Many digital wireless protocols (such as 802.11b/g [83, 84], CDMA [85], RFID [86]) use PN sequences as identifiers	New metric for anti-jamming fitness
SINR $\rightarrow$ SNR as I $\rightarrow 0$	SNR limited	I/Q demodulated sequence $\rightarrow$ TX sequence + noise as I $\rightarrow 0$	SNR limited
$\mathcal{O}(N)$ SINR calculation	$\mathcal{O}(N^3)$ spectral estimation	$\mathcal{O}(N)$ cross-correlation with $\mathcal{O}(\log 2N)$ max search	Assumes transmitted sequence known by receiver

$\max|R_{XY}(\tau)| = 0 \forall \tau < 0$ since the system turns at at time $t = \tau = 0$.

A comparison of SINR versus the maximum of the absolute valued PN sequence cross-correlation is presented in Table 7.1. An issue with SINR fitness is that it requires spectral estimation to estimate SOI and interference power levels. If the SOI and at least one interfering signal have similar spectra, the signals cannot be classified as SOI or interference, and SINR cannot be computed. In addition, spectral computation is computationally intensive. For example, the MUSIC algorithm uses singular value decomposition (SVD) which is $\mathcal{O}(N^3)$ per Golub *et al.* [87].

This method has disadvantages in the sense that it includes a new metric for anti-

jamming fitness, and it assumes that the receiver knows the transmitted PN sequence *a priori*. Neither disadvantages prevent this method from being implemented for the reason that PN sequences are used in many digital wireless standards [83–85] as noted in Table 7.1.

7.2.2 Modular Eight Element Array

The analytical model described in [34] assumes that the beamforming system operates in a Rayleigh fading channel, and algorithms exist that can separate SOI from interference and calculate their power levels at the array's output. It calculates the array's output Signal to Interference and Noise Ratio (SINR) as the GA's fitness function. Although our simulations and subsequent experiments in [34,77–80,82] show that the GA can thwart interference once their power levels have been determined, it may not be feasible to implement a spectral estimation algorithm prior to optimizing the array's hardware settings.

Future work includes developing a real-time, anti-jamming adaptive beamforming system based on the analytical model of Figure 7.1 that can be used defensively to thwart jammers in a lab environment. The system will be built with Printed Circuit Boards (PCBs) using surface mount phase shifters and other electronic components such as microcontrollers. The system will be modular with one antenna and accompanying phase shifters and step-attenuators on each module PCB, as this will allow multiple layout configurations of the beamforming system. It will have eight antenna modules, and the system will have a separate microcontroller PCB module that al-

lows control of different beamformer configurations via a main computer. While the research will focus on WiFi (802.11n, 2.4 GHz and 5.8 GHz) frequency bands, the results will be applicable to other frequency bands from 2.3 GHz to 6.0 GHz.

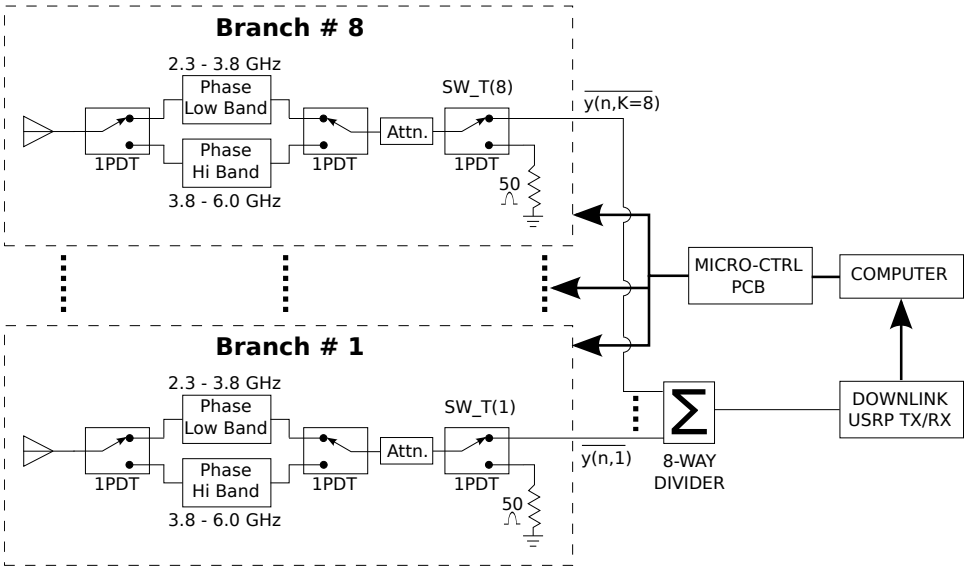


Figure 7.4: Block diagram of a wideband, modular beamforming system.

A block diagram of a proof-of-concept (POC) hardware system is shown in Figure 7.4. Each branch is a PCB module with a wideband antenna. Each module has multiple one-pole-double-throw (1PDT) switches that allow the system to switch between a low and high band based on commercially available surface mount microwave components. The beamforming system can be expanded to more than eight antennas with multiple RF power dividers and microcontroller PCBs.

7.2.3 Direction Finding and Adversarial Capabilities

The eight antenna modular POC hardware system (shown in Figure 7.4) has a natural potential to provide direction finding and offensive active jamming capabilities with the development of additional software. For offensive capabilities, a beamforming

system must detect directions in which it needs to transmit signals to jam advisories. A flowchart showing how the algorithm adds signal direction tracking and adversary jamming is shown in Figure 7.5. The receive (i.e., RCV) mode GA of Figure 7.2 is run until convergence, and the best found GA solution is held in active memory. The SOI is the only signal present at the array's output. The algorithm connects one antenna branch to the power divider at a time and collects multiple time-indexed samples after each branch. The MUSIC algorithm calculates the SOI's direction using these samples. MUSIC can be used with any antenna array layout with antenna positions incorporated into the algorithm [1].

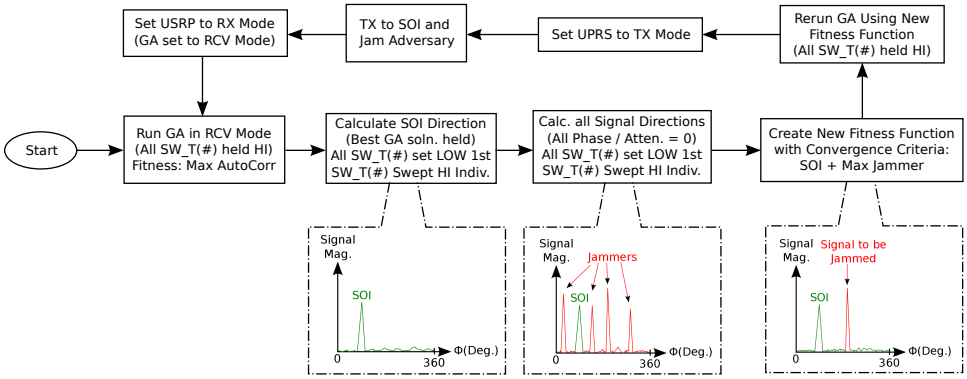


Figure 7.5: Algorithm flowchart for potential direction finding and active jamming capabilities.

The MUSIC algorithm is run again with the phase shifters and step attenuators set to 0. All signals are present at the array's output with these settings, but the SOI's direction is known from the previous step and can be isolated from interference. The new fitness function tracks MUSIC signal magnitudes in SOI and maximum strength jammer directions. It assumes that the jammer with the maximum signal strength is the closest adversary and is therefore the most desirable candidate to be jammed.

The convergence criteria are shown graphically in Figure 7.5 (rightmost MUSIC signal diagram).

The offensive jamming mode GA converges when the MUSIC output containes only the SOI and maximum jammer directions. When the system returns to RX mode, it is assumed that the previous RCV mode GA solution is no longer valid, as the adversary would likely change its jammer characteristics (i.e., power, frequency, and/or location), or the wireless channel would have changed. The algorithm in Figure 7.5 cannot be effectively run prior to running the receive mode GA because signal directions are not known *a priori*, nor can the SOI be isolated from the jammers before anti-jamming.

7.2.4 Fault Detection and Localization Algorithms

Fault localization theory discussed in Chapter 4 localizes faults in an antenna array by correlating the radiation patterns measured *in-situ* (either in an anechoic chamber or in the field with test equipment) with calculated patterns that have a candidate antenna removed at various locations. This generates a probability that a fault occurs at that location. This theory is easily extended to multiple faults with coupling between antennas considered, and the fault position(s) that generate the highest cross-correlation values are the best candidate positions for fault locations.

This method requires that the array be removed from wireless system operation and connected to test equipment to measure its radiation patterns. A more ideal method allows fault detection and localization to be performed *in-situ* with connected

wireless sub-systems. However, the current POC system investigated in this thesis (see Chapters 5 and 4) does not lend itself to performing these functions. To localize faults when the beamforming array operates *in-situ* with other wireless subsystems, an array structure with multiple receivers or a reverse commutated single-point connection (see Figure 7.4) is necessary. The real-time localization algorithm performs cross-correlations like the method described in Section 4.4, but the cross-correlations calculated using the architecture shown in Figure 7.4 are of time-sampled power measurements collected for all N branches, one branch at a time. Our advancement in hardware fault recovery via stochastic algorithms helps integration in the military field because our algorithms make the array more reliable without adding additional hardware. Interference in wireless communications has long been a problem and will continue being a problem as wireless devices become more and more common everyday.

Bibliography

[1] F. B. Gross, *Frontiers in Antennas: Next Generation Design & Engineering*. New York, NY: McGraw-Hill Professional, 2011.

[2] H. Van Trees, *Detection, Estimation, and Modulation Theory, Optimum Array Processing*, ser. Detection, Estimation, and Modulation Theory. Wiley, 2004.

[3] C. Balanis, *Antenna Theory: Analysis and Design*, 3rd ed. Hoboken, New Jersey: Wiley, 2005.

[4] R. J. Mailloux, *Phased Array Antenna Handbook*, 2nd ed. Boston, MA: Artech House, 2005.

[5] M. Manteghi and R. Blanco, "A novel technique for a low-cost digital phased array design," *IEEE Transactions on Antennas and Propagation*, vol. 61, no. 7, pp. 3495–3501, 2013.

[6] P. J. Bevelacqua, "Antenna arrays: Performance limits and geometry optimization," Ph.D. dissertation, Arizona State University, May 2008.

[7] C. Balanis, *Advanced Engineering Electromagnetics*, 1st ed. Hoboken, New Jersey: Wiley, 1989.

[8] Y. Zhang and M. Amin, "Anti-jamming GPS receiver with reduced phase distortions," *IEEE Signal Processing Letters*, vol. 19, no. 10, pp. 635–638, 2012.

[9] F. B. Gross, "Smart Antennas," in *Antenna Engineering Handbook*, 4th ed., J. L. Volakis, Ed. New York, NY: McGraw-Hill, 2007, ch. 25.

[10] F. Chiti, R. Fantacci, L. Maccari, D. Marabissi, and D. Tarchi, "A broadband wireless communications system for emergency management," *IEEE Wireless Communications*, vol. 15, no. 3, pp. 8–14, 2008.

[11] H. Khatib, "Theater wideband communications," in *MILCOM 97 Proceedings*, vol. 1, 1997, pp. 378–382 vol.1.

[12] C. Rabbath and N. Léchevin, *Safety and Reliability in Cooperating Unmanned Aerial Systems*. Hackensack, NJ: World Scientific, 2010.

[13] R. Klemm, "Special Aspects of Airborne MTI Radar," in *Principles of Space-Time Adaptive Processing*, 2nd ed. London, United Kingdom: The Instituition of Electrical Engineers, 2002, ch. 15.

[14] G. Oliveri, P. Rocca, and A. Massa, "Reliable diagnosis of large linear arrays: A bayesian compressive sensing approach," *IEEE Transactions on Antennas and Propagation*, vol. 60, no. 10, pp. 4627–4636, 2012.

[15] A. K. Agrawal and E. Holzman, "Active phased array design for high reliability," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 35, no. 4, pp. 1204–1211, 1999.

[16] E. Dubrova, *Fault-Tolerant Design*. New York, NY: Springer Science+Business Media, 2013.

[17] G. S. Hornby, A. Globus, D. S. Linden, and J. D. Lohn, "Automated antenna design with evolutionary algorithms," American Institute of Aeronautics & Astronautics, Tech. Rep., 2006.

[18] J. Lohn, W. Kraus, and D. Linden, "Evolutionary optimization of a quadrifilar helical antenna," in *IEEE Antennas and Propagation Society International Symposium*, vol. 3, 2002, pp. 814–817.

[19] X. Chen, K. Huang, and X.-B. Xu, "Automated design of a three-dimensional fishbone antenna using parallel genetic algorithm and NEC," *IEEE Antennas and Wireless Propagation Letters*, vol. 4, pp. 425–428, 2005.

[20] S. Koulouridis, D. Psychoudakis, and J. Volakis, "Multiobjective optimal antenna design based on volumetric material optimization," *IEEE Transactions on Antennas and Propagation*, vol. 55, no. 3, pp. 594–603, March 2007.

[21] D. W. Boeringer, D. H. Werner, and D. W. Machuga, "A simultaneous parameter adaptation scheme for genetic algorithms with application to phased array synthesis," *IEEE Transactions on Antennas and Propagation*, vol. 53, no. 1, pp. 356–371, January 2005.

[22] B.-K. Yeo and Y. Lu, "Array failure correction with a genetic algorithm," *IEEE Transactions on Antennas and Propagation*, vol. 47, no. 5, pp. 823–828, 1999.

[23] J.-H. Han, S.-H. Lim, and N.-H. Myung, "Array antenna TRM failure compensation using adaptively weighted beam pattern mask based on genetic algorithm," *IEEE Antennas and Wireless Propagation Letters*, vol. 11, pp. 18–21, 2012.

[24] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Adison-Wesley, 1989.

[25] J. H. Holland, *Adaptation in Natural and Artificial Systems*, First MIT press ed. Cambridge, MA: MIT Press, 1992.

[26] K. A. De Jong, "Analysis of the behavior of a class of genetic adaptive systems," Ph.D. dissertation, University of Michigan, Ann Arbor, MI, 1975.

[27] K. A. De Jong, *Evolutionary Computation: A Unified Approach.* Cambridge, MA: The MIT Press, 2006.

[28] D. S. Weile and E. Michielssen, "The control of adaptive antenna arrays with genetic algorithms using dominance and diploidy," *IEEE Transactions on Antennas and Propagation*, vol. 49, no. 10, pp. 1424–1433, October 2001.

[29] P. van Laarhoven and E. Aarts, *Simulated Annealing: Theory and Applications*, ser. Mathematics and Its Applications. Norwell, MA: Kluwer Academic Publishers, 1987.

[30] H. Evans, P. Gale, B. Aljibouri, E. Lim, E. Korolkwiwicz, and A. Sambell, "Application of simulated annealing to design of serial feed sequentially rotated 2 times 2 antenna array," *Electronics Letters*, vol. 36, no. 24, pp. 1987–1988, 2000.

[31] J. Dong, J. Yang, W. Lei, R. Shi, and Y. Guo, "Antenna array design in MIMO radar using cyclic difference sets and simulated annealing," in *2012 International Conference on Microwave and Millimeter Wave Technology (ICMMT)*, vol. 1, 2012, pp. 1–4.

[32] Y. Yang, Y. Tan, R. Liang, Q. Wang, and N. Yuan, "A calibration algorithm of millimeter-wave sparse arrays based on simulated annealing," in *2010 International Conference on Microwave and Millimeter Wave Technology (ICMMT)*, 2010, pp. 1702–1705.

[33] R. L. Haupt, *Antenna Arrays: A Computational Approach.* Hoboken, New Jersey: John Wiley & Sons, Inc., 2010.

[34] J. Lohn, J. M. Becker, and D. Linden, "An evolved anti-jamming adaptive beamforming network," *Genetic Programming and Evolvable Machines*, vol. 12, no. 3, pp. 217–234, 2011.

[35] D. Krusienski and W. Jenkins, "A particle swarm optimization-least mean squares algorithm for adaptive filtering," in *Conference Record of the Thirty-Eighth Asilomar Conference on Signals, Systems and Computers*, vol. 1, Nov 2004, pp. 241–245 Vol.1.

[36] K.-M. Lee, R.-S. Chu, and S.-C. Liu, "A built-in performance-monitoring/fault isolation and correction (pm/fic) system for active phased-array antennas," *IEEE Transactions on Antennas and Propagation*, vol. 41, no. 11, pp. 1530–1540, 1993.

[37] J. Becker, J. Bain, and J. Hoburg, "Dipole arrays, electronically steered arrays, and anti-jamming adaptive beamforming arrays," April 2011, class Lecture.

[38] G. T. F. De Abreu and R. Kohno, "A modified Dolph-Chebyshev approach for the synthesis of low sidelobe beampatterns with adjustable beamwidth," *IEEE Transactions on Antennas and Propagation*, vol. 51, no. 10, pp. 3014–3017, 2003.

[39] B. K. Lau and Y. Leung, "A Dolph-Chebyshev approach to the synthesis of array patterns for uniform circular arrays," in *2000 IEEE International Symposium on Circuits and Systems (ISCAS) Geneva*, vol. 1, 2000, pp. 124–127 vol.1.

[40] N. Goto and Y. Tsunoda, "Sidelobe reduction of circular arrays with a constant excitation amplitude," *IEEE Transactions on Antennas and Propagation*, vol. 25, no. 6, pp. 896–898, 1977.

[41] H. Steyskal, "Circular array with frequency-invariant pattern," in *Antennas and Propagation Society International Symposium, 1989. AP-S. Digest*, 1989, pp. 1477–1480 vol.3.

[42] C. Stearns and A. Stewart, "An investigation of concentric ring antennas with low sidelobes," *IEEE Transactions on Antennas and Propagation*, vol. 13, no. 6, pp. 856–863, 1965.

[43] T. Zhang and W. Ser, "Robust beampattern synthesis for antenna arrays with mutual coupling effect," *IEEE Transactions on Antennas and Propagation*, vol. 59, no. 8, pp. 2889–2895, 2011.

[44] S. Applebaum, "Adaptive arrays," *IEEE Transactions on Antennas and Propagation*, vol. 24, no. 5, pp. 585–598, 1976.

[45] L. Lei, X. Rongqing, and L. Gaopeng, "Robust adaptive beamforming based on generalized sidelobe cancellation," in *International Conference on Radar, 2006. CIE '06*, 2006, pp. 1–4.

[46] Y. Xu and Z. Liu, "Noncircularity-rate maximization: A new approach to adaptive blind beamforming," in *5th International Conference on Wireless Communications, Networking and Mobile Computing, 2009. WiCom '09.*, 2009, pp. 1–4.

[47] S. Chen, L. Hanzo, N. N. Ahmad, and A. Wolfgang, "Adaptive minimum bit error rate beamforming assisted receiver for QPSK wireless communication," *Digit. Signal Process.*, vol. 15, no. 6, pp. 545–567, Nov. 2005.

[48] H.-C. Lee, D.-C. Oh, and Y.-H. Lee, "Coordinated user scheduling with transmit beamforming in the presence of inter-femtocell interference," in *2011 IEEE International Conference on Communications (ICC)*, 2011, pp. 1–5.

[49] R. Haupt and H. Southall, "Experimental adaptive nulling with a genetic algorithm," *Microwave Journal*, vol. 42, no. 1, pp. 78–89, 1999.

[50] A. Massa, M. Donelli, F. G. B. D. Natale, S. Caorsi, and A. Lommi, "Planar antenna array control with genetic algorithms and adaptive array theory," *IEEE Transactions on Antennas and Propagation*, vol. 52, no. 11, pp. 2919–2924, November 2004.

[51] Y.-J. Lee, J.-W. Seo, J.-K. Ha, and D.-C. Park, "Null steering of linear phased array antenna using genetic algorithm," in *2009 Asia Pacific Microwave Conference (APMW)*, 2009, pp. 2726–2729.

[52] F. Ares, S. Rengarajan, E. Villaneuva, E. Skochinski, and E. Moreno, "Application of genetic algorithms and simulated annealing technique in optimising the aperture distributions of antenna array patterns," *Electronics Letters*, vol. 32, no. 3, pp. 148–149, 1996.

[53] F. Ares, S. Rengarajan, E. Villanueva, E. Skochinski, and E. Moreno, "Application of genetic algorithms and simulated annealing technique in optimizing the aperture distributions of antenna arrays," in *Antennas and Propagation Society International Symposium, 1996. AP-S. Digest*, vol. 2, 1996, pp. 806–809 vol.2.

[54] D. Sadler, "Planar array design for low ambiguity," in *2009 Loughborough Antennas Propagation Conference (LAPC)*, 2009, pp. 713–716.

[55] A. Pascual-Iserte, A. I. Pérez-Neira, and M. A. Lagunas, "An approach to optimum joint beamforming design in a mimo-ofdm multiuser system," *EURASIP J. Wirel. Commun. Netw.*, vol. 2004, no. 2, pp. 210–221, Dec. 2004.

[56] D. Migliore, "A compressed sensing approach for array diagnosis from a small set of near-field measurements," *IEEE Transactions on Antennas and Propagation*, vol. 59, no. 6, pp. 2127–2133, 2011.

[57] N. Chamberlain, "Impulse testing of corporate-fed patch array antennas," in *2011 IEEE Aerospace Conference*, 2011, pp. 1–15.

[58] S. Liu, "A fault correction technique for phased array antennas," in *Antennas and Propagation Society International Symposium, 1992. AP-S. 1992 Digest. Held in Conjuction with: URSI Radio Science Meeting and Nuclear EMP Meeting., IEEE*, 1992, pp. 1612–1615 vol.3.

[59] M. Joler, "Self-recoverable antenna arrays," *IET Microwaves Antennas Propagation*, vol. 6, no. 14, pp. 1608–1615, 2012.

[60] S. Mitilineos, S. C. A. Thomopoulos, and C. Capsalis, "On array failure mitigation with respect to probability of failure, using constant excitation coefficients and a genetic algorithm," *IEEE Antennas and Wireless Propagation Letters*, vol. 5, no. 1, pp. 187–190, 2006.

[61] M. J. Rivera, J. Costantine, Y. Tawk, and C. G. Christodoulou, "Detection of failures in switch reconfigurable antenna arrays using embedded sensing lines," in *2012 IEEE Antennas and Propagation Society International Symposium (APSURSI)*, 2012, pp. 1–2.

[62] R. Iglesias, F. Ares, M. Fernandez-Delgado, J. Rodriguez, J. Bregains, and S. Barro, "Element failure detection in linear antenna arrays using case-based reasoning," *IEEE Antennas and Propagation Magazine*, vol. 50, no. 4, pp. 198–204, 2008.

[63] B.-K. Yeo and Y. Lu, "Fast detection and location of failed array elements using the fast SVM algorithm," in *2010 14th International Symposium on Antenna Technology and Applied Electromagnetics the American Electromagnetics Conference (ANTEM-AMEREM)*, 2010, pp. 1–4.

[64] J. Rodríguez-González, F. Ares-Pena, M. Fernández-Delgado, R. Iglesias, and S. Barro, "Rapid method for finding faulty elements in antenna arrays using far field pattern samples," in *3rd European Conference on Antennas and Propagation (EuCAP)*, 2009, pp. 3380–3384.

[65] A. Alexiou and A. Manikas, "Array robustness to sensor failure," in *2000 IEEE International Conference on Phased Array Systems and Technology*, 2000, pp. 177–180.

[66] A. Sleiman and A. Manikas, "The impact of sensor positioning on the array manifold," *IEEE Transactions on Antennas and Propagation*, vol. 51, no. 9, pp. 2227–2237, 2003.

[67] D. Cheng, *Field and Wave Electromagnetics*, 2nd ed., ser. Addison-Wesley Series in Electrical Engineering. Addison-Wesley, 2004.

[68] M. D. Greenberg, *Advanced Engineering Mathematics*, 2nd ed. Upper Saddle River, NJ: Prentice Hall, 1998.

[69] J. Proakis, *Digital Communications*, ser. McGraw-Hill series in electrical and computer engineering. McGraw-Hill Higher Education, 2001.

[70] A. Papoulis and S. U. Pillai, *Probability, Random Variables, and Stochastic Processes*, 4th ed. New York, NY: McGraw-Hill Companies Inc., 2002.

[71] R. D. Yates and D. J. Goodman, *Probability and Stochastic Processes: A Friendly Introduction for Electrical and Computer Engineers*, 2nd ed. Hoboken, NJ: John Wiley & Sons, Inc., 2005.

[72] D. S. Weile and D. S. Linden, "AntNet: A fast network analysis add-on for WIPL-D," in *27th International Review of Progress in Applied Computational Electromagnetics*, March 2011, pp. 1–5.

[73] E. Newman, J. Richmond, and C. H. Walter, "Superdirective receiving arrays," *IEEE Transactions on Antennas and Propagation*, vol. 26, no. 5, pp. 629–635, 1978.

[74] E. Gilbert and S. Morgan, "Optimum design of directive antenna arrays subject to random variations," *Bell System Technical Journal*, vol. 34, no. 3, pp. 637–663, 1955.

[75] M. Gustafsson and S. Nordebo, "Optimal antenna currents for Q, superdirectivity, and radiation patterns using convex optimization," *IEEE Transactions on Antennas and Propagation*, vol. 61, no. 3, pp. 1109–1118, 2013.

[76] Y. Ma, Y. Yang, Z. He, K. Yang, C. Sun, and Y. Wang, "Theoretical and practical solutions for high-order superdirectivity of circular sensor arrays," *IEEE Transactions on Industrial Electronics*, vol. 60, no. 1, pp. 203–209, 2013.

[77] J. Becker, J. Lohn, and D. Linden, "An anti-jamming beamformer optimized using evolvable hardware," in *2011 IEEE International Conference on Microwaves, Communications, Antennas, and Electronic Systems (COMCAS)*, November 2011, pp. 1–5.

[78] J. Becker, J. D. Lohn, and D. Linden, "An in-situ optimized anti-jamming beamformer for mobile signals," in *2012 IEEE International Symposium on Antennas and Propagation, IEEE APS*, July 2012, pp. 1–2.

[79] J. Becker, J. Lohn, and D. Linden, "Evaluation of genetic algorithms in mitigating wireless interference in situ at 2.4 GHz," in *WiOpt 2013 Indoor and Outdoor Small Cells Workshop*, May 2013, pp. 1–8.

[80] J. Becker, J. D. Lohn, and D. Linden, "Algorithm comparison for in-situ beamforming," in *2013 IEEE International Symposium on Antennas and Propagation, IEEE APS*, July 2013, pp. 1–2.

[81] S. J. Louis and G. J. Rawlins, "Predicting convergence time for genetic algorithms," *Foundations of Genetic Algorithms*, vol. 2, pp. 141–161, 1993.

[82] J. Becker, J. D. Lohn, and D. Linden, "Towards a self-healing, anti-jamming adaptive beamforming array," in *2013 IEEE-APS Topical Conference on Antennas and Propagation in Wireless Communications (APWC)*, September 2013, pp. 1–4.

[83] B. O'Hara and A. Petrick, *IEEE 802.11 Handbook: A Designer's Companion*, 2nd ed., ser. IEEE Standards Wireless Networks Series. New York, NY: Wiley, 2005.

[84] "IEEE Standard for Information Technology- Telecommunications and Information Exchange Between Systems- Local and Metropolitan Area networks- Specific Requirements Part Ii: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," *IEEE Std 802.11g-2003 (Amendment to IEEE Std 802.11, 1999 Edn. (Reaff 2003) as amended by IEEE Stds 802.11a-1999, 802.11b-1999, 802.11b-1999/Cor 1-2001, and 802.11d-2001)*, pp. i–67, 2003.

[85] T. S. Rappaport, *Wireless Communications: Principles and Practice*, 2nd ed., ser. Prentice Hall Communications Engineering and Emerging Technologies. Upper Saddle River, N.J.: Prentice Hall, 2002.

[86] J. Becker, M. Trotter, and J. Griffin, "Passive displacement sensing using backscatter RFID with multiple loads," in *2013 IEEE Sensors*, November 2013, pp. 1–4.

[87] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed. Baltimore, Maryland: The Johns Hopkins University Press, 1996.