Multiple Objective Fitness Functions for Cognitive Radio Adaptation

Timothy R. Newman

Submitted to the Department of Electrical Engineering & Computer Science and the Faculty of the Graduate School of the University of Kansas in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Thesis Committee:

Dr. Joseph B. Evans, Advisor
Professor, EECS

Dr. Gary J. Minden
Professor, EECS

Dr. Perry W. Alexander
Professor, EECS

Dr. Alexander M. Wyglinski
Assistant Professor, ECE (WPI)

Dr. Tyrone Duncan
Professor, Mathematics

Date Approved

The Dissertation Committee for Timothy R. Newman certifies
That this is the approved version of the following dissertation:

Multi-Objective Fitness Functions for Multi-Carrier Cognitive Radio Systems

Committee:

Dr. Joseph B. Evans, Advisor
Professor, EECS

Dr. Gary J. Minden
Professor, EECS

Dr. Perry W. Alexander
Professor, EECS

Dr. Alexander M. Wyglinski
Assistant Professor, ECE (WPI)

Dr. Tyrone Duncan
Professor, Mathematics

Date Approved

Abstract

This thesis explores genetic algorithm and rule-based optimization techniques used by cognitive radios to make operating parameter decisions. Cognitive radios take advantage of intelligent control methods by using sensed information to determine the optimal set of transmission parameters for a given situation. We have chosen to explore and compare two control methods. A biologically-inspired genetic algorithm (GA) and a rule-based expert system are proposed, analyzed and tested using simulations. We define a common set of eight transmission parameters and six environment parameters used by cognitive radios, and develop a set of preliminary fitness functions that encompass the relationships between a small set of these input and output parameters. Five primary communication objectives are also defined and used in conjunction with the fitness functions to direct the cognitive radio to a solution. These fitness functions are used to implement the two cognitive control methods selected. The hardware resources needed to practically implement each technique are studied. It is observed, through simulations, that several trade offs exist between both the accuracy and speed of the final decision and the size of the parameter sets used to determine the decision. Sensitivity analysis is done on each parameter in order to determine the impact on the decision making process each parameter has on the cognitive engine. This analysis quantifies the usefulness of each parameter.

To my beautiful and patient wife and my two lovely daughters.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Research Motivation

Cognitive radio technology is receiving significant attention as an approach to alleviate the FCC identified problem of the scarcity of available radio spectrum [1–4]. Cognitive radios take advantage of the reconfigurable attributes of a conventional software-defined radio (SDR) by using an "'intelligent"' control method to automatically adapt operating parameters based on learning from previous events and current inputs to the system. The momentum of research efforts, due in part to the current spectrum scarcity problem, as well as a Department of Defense initiative [1] to develop a flexible software radio approach for war-fighter communications, has yielded numerous initiatives and programs by researchers in academia [5] and industry [6]. The resulting plethora of cognitive radio solutions range from cognitive radio components and radio network testbeds [5] to complete radio systems [2]. However, there is still no common consensus on how to implement a cognitive radio, most importantly no agreement on the best method used as the "'intelligent"' control. This research investigates two possible methods for "'intelligent"' control and derives an analytical relationship between the radio environment and the radio transmission parameters that drive the control method

to a solution. From this relationship, sensitivity analysis is performed on the communication parameters commonly used in wireless communications in order to determine the performance impact for each parameter.

The initial focus of this thesis is to investigate the question of what technology could be used to implement a cognitive radio decision making engine. However, before this question can be examined, several other questions must be answered in order to piece together a conclusion. In the most general sense, a cognitive radio uses information about the environment, and determines the best possible set of transmission parameters to use given some set of service performance objectives. Defining the environmental inputs used to make accurate decisions has a major impact on the accuracy of the cognitive radio decisions. These measurements are the basis of the decisions being made in the system. Similarly, defining the set of transmission parameters that are controlled by the cognitive radio also dramatically affects the efficiency of the radio. Using several references and experience in the communications field, we propose a list of six transmission parameters and six environmental measurements that are used to investigate the implementation of the cognitive decision methods. The transmission parameters chosen represent common transmission parameters that span the physical, MAC, and network layers. The environment measurements were selected based upon common attributes that can be measured and that affect the operation of the radio.

With a properly defined list of transmission parameters and measurement inputs, a cognitive radio engine uses the relationships between the parameters and measurements to select the optimal set of transmission parameters. However, a function that represents the relationship between the set of environmentally sensed parameters and the set of controllable transmission parameters does not yet exist. For example, this function is needed by the cognitive radio engine to understand that by modifying transmission parameter A, theoretically the environmentally sensed parameters B, C, and D,

should be affected in a certain way. This requires an analytical relationship to be found that encompasses both the input and output radio parameters. The solution to this problem is to create an analytical environment model of the communication environment. In this work, we derive relationships between the transmission and environmental parameters and present a function that is used as the adaptive engine for a cognitive radio implementation. Once the function is defined, determining the appropriate transmission parameters becomes an optimization problem.

Determining the proper method to solve this optimization problem is the primary research goal of this dissertation. First, we classify this optimization problem as non-linear because of the nature of the wireless environment. Variables such as the received power of a signal, noise power, or path loss can vary widely depending on the state of the current wireless system. Between the many different fading environments and multipath effects, the measured value of environment variables can change based on temporal or spatial differences. Several non-linear models exist simply to model the fading characteristics in an AWGN channel [7].

Several methods exist to solve optimization problems. The complexity of non-linear and dynamically changing wireless communication environments make using traditional non-linear programming (NLP) optimization methods problematic with respect to the convergence to a local optimum, or in some situations the inability to find a feasible solution. Typically, non-linear optimization techniques use some form of a gradient search technique to move along the slope until the maximum point it reached. This problem is illustrated in Figure 1.1. To solve this type of optimization problem, we propose using techniques from the artificial intelligence (AI) domain. AI methods such as genetic algorithms and simulated annealing take advantage of random mutations in order to avoid the local optima problem inherent in gradient techniques.

AI can be divided into roughly two schools of thought: Conventional AI and Com-

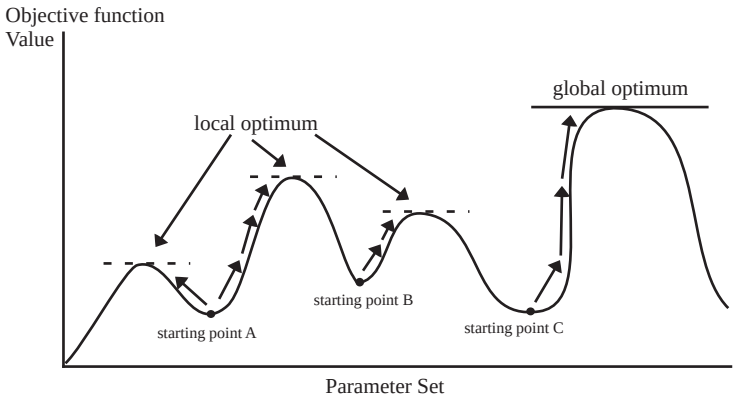

Figure 1.1 Gradient search technique

putational AI. Conventional AI, also commonly referred to as *machine learning*, encompasses a wide variety of technologies including, *expert systems* [8, 9], *case-based learning* [10], and *reinforcement learning* [11]. The machine learning techniques follow the simple process flow illustrated in Figure 1.2. With parameters, sensor data, and objectives as input to the system, machine learning uses the objectives and sensor data to determine a possible action. Once this action is applied to the system, a teacher provides feedback to the machine learning component, providing a quantitative measurement that represents the effectiveness of the action.

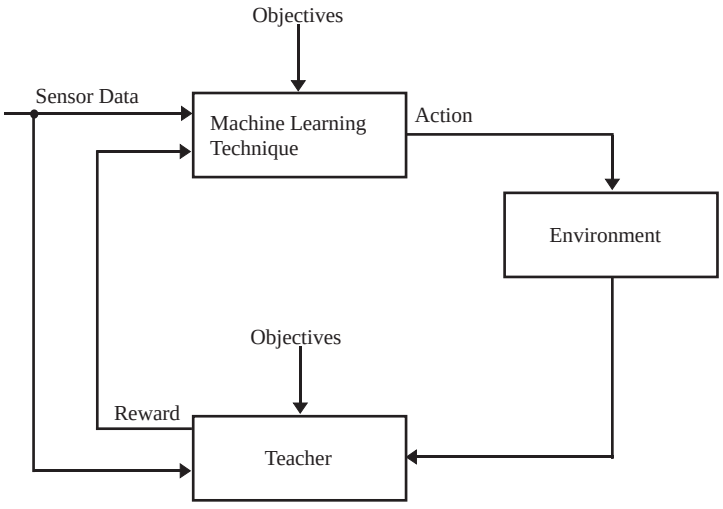

Figure 1.2 Machine Learning Flow

Machine learning techniques must make an approximation, based upon the environmental measurements, of how well a given action will achieve the goal. This specific approximation stage will be called the *criteria selection* stage. Within this stage the actions are given scores based on how well they achieve the goal. Using this knowledge, the technique either continues to evolve and try to create better actions, or outputs the current action to be applied to the system. Typically, the function used to determine the optimality of the output is called the *utility function* or *fitness function* [12]. The optimization world tends to use *utility function*, while *fitness function* is typically used for genetic algorithms. We will use the term *fitness function* in this research which will be considered analageous to *utility function*. The inputs to the fitness function include the set of objectives for the system, the current environmental parameters or sensor data, and the possible action to be scored. Translating into radio terms, the parameters become: the intended performance goals for the radio, the measured channel parameters, and a set of controllable transmission parameters.

We investigate the applicability of a non-linear heuristic approach using genetic algorithms as the cognitive radio engine. A wireless transmission environment is a non-linear environment, especially when the wireless devices are mobile. For example, this can be seen from any of the theoretical bit-error-rate models given in [7]. Signal strengths are constantly changing, other wireless nodes are appearing and disappearing randomly, and physical obstacles create multipath effects that can be model using the Rayleigh distribution [7]. The implicit parallelisms of genetic algorithms allow it to exploit large numbers of regions in the search space while working on relatively few strings. This parallelism also helps genetic algorithm overcome nonlinear problems [13]. Genetic algorithms (GAs), like simulated annealing (another popular non-linear search method), use a random walk method and a mutation probability to help guide their search and avoid local optima. However, unlike simulated annealing,

GA's use a randomly chosen set of models and use binary strings for the representation of these models. Unlike a random search, such as the "Monte Carlo" method, the search used in the GA is not directionless. However, GAs typically do not converge to the absolute optimum value. The accuracy of the GA decision is directly related to the number of iterations, or generations, that the GA uses to process the information. Thus, a secondary research question is: "What are the optimal number of generations needed for the GA to provide an acceptable solution?".

As an alternative to the genetic algorithm, we use the fitness function to explore the entire possible search space offline. This information is used to create a rule base for an expert system. To model the wireless environment accurately, several parameters are needed, each having a wide range of possible values. This creates a large search space for our problem. The expert system must take into account a large amount of information and the memory resources needed to represent every possible environment state may become an issue. In our work, the total number of possible environments reaches 100 million combinations taking into account all possible scenarios. Representing the real-word environmental state requires us to approximate the sensed state to a certain degree of precision, and therefore introduces some error. This error is directly related to the degree of precision used to create the rules. Thus, we examine the expert system specific question: "What are the trade offs between the degree of precision of the rule base and the accuracy of the expert system?".

We foresee both genetic algorithms and expert systems having their advantages and disadvantages. GAs require little memory. However, they can typically only achieve near-optimal solutions. Expert systems have the advantage of being able to produce a solution fast due to the offline generated rules, but typically require a large amount of memory to be implemented and determining the degree of precision of the rules can become a problem. This thesis investigates the applicability of these two methods to the

cognitive radio engine domain. Each of these methods uses the relationships between the environmentally sensed parameters and the controllable transmission parameters to determine the solution, thus deriving this relationship is an important task.

In multiple objective problems, such as the cognitive radio environment addressed in this research, generating fitness functions becomes difficult. In the case of multiple objectives, where several objectives are often competing, there is not a single optimal solution, but rather a set of possible solutions. These solutions are optimal in the sense that no other solutions are superior to them when all objectives are being considered equally. This set of possible solutions is known as the *Pareto-optimal* solution set [14–16]. Although, many different solutions have been proposed to alleviate this problem [14], this disseration focuses on using preference information through a weighted sum approach [17–20]. Preference information is used to rank the objectives in order of importance. This importance ranking can then be used to single out a solution that represents the optimal solution for a specific ranking of objectives.

A comprehensive study of multiple objective optimization problems for evolutionary algorithms (EA) was published by Fonseca and Fleming [14]. This study categorized different approaches in formulating fitness functions, including aggregation methods, population based non-Pareto approaches, Pareto-based approaches, and methods using the niched induction technique. Aggregation methods combine objectives to form a single fitness function. Formulating this single equation requires a large amount of domain knowledge to form complete relationships between the objectives and the parameters. Pareto-based, non-Pareto based, and the niched techniques have the advantage of being able to solve for a family of solutions if preference information is not available. However, this work uses preference information, such as objective weights, to rank objectives by their importance pertaining to the goal state. Consequently, using an aggregation method creates a single fitness function, instead of a family of equa-

tions that would be more difficult to implement and much less tractable of a solution. The domain knowledge needed to create the aggregate fitness function is sought using analytical relationships between the parameters and the objectives. However, some parameters may not have closed form solutions that relate them to the objectives. These relationships must be found through extensive computer simulations. A goal of this dissertation is combine these relationships into an aggregate function that can be used to relate a general set of transmission parameters to a set of environmental parameters and performance objectives.

Research conducted at Virginia Tech has also developed a genetic algorithm engine for cognitive radios [2, 21]. Their simulation results validate that their genetic algorithm implementation does in fact change the transmission parameters to different settings, based upon a set of objectives. However, more research is needed in the area of fitness functions, which are not analyzed or presented in their work. The focus of the research thus far has been showing that the genetic algorithms can converge to a suitable set of transmission parameters. There has been no mention of the time requirements or memory resources needed to perform this task using genetic algorithms. The work presented in this paper goes beyond just demonstrating that the genetic algorithm outputs a selection, but also provides the numerical analysis for the fitness functions that drive the GA's and present simulations and analysis showing the practical resource usage for both genetic algorithms and the alternative expert systems implementation.

1.2 Research Objectives

The primary research question we answer in this dissertation is: What is the best adaptation technique that can be used to implement a cognitive radio? Before we begin to answer this question, we explore the characteristics of a cognitive engine in order

to determine what is needed for a cognitive engine to operate. We have identified the need for a set of common parameters that must be used by the cognitive engine to make decisions. These parameters represent both, transmission level control parameters and environmentally sensed parameters. It is important that this list consist of common parameters so that the cognitive engine analyzed in this work can be related to the majority of other radios being developed. In addition to needing a list of parameters, radio operating goals must also be defined. These goals are needed to guide the system to a specific output, and may change depending upon the wireless environment, operating scenario, or hardware conditions of the radio.

The ideal cognitive radio observes the wireless environment and make transmission parameter modifications based upon those observations and the radio operating goals. Based upon this definition, the implementation of the cognitive engine is an optimization technique. The primary research question we address in this dissertation is now more specific: What is the best optimization technique or adaptation technique to use within the cognitive radio engine? A choice must now be made between several possible techniques. To determine the best technique, we first analyze our problem domain to find important characteristics that let us weed out possible optimization techniques. Thus, characterizing our problem domain is an important task in order to answer the primary research question.

The secondary question that is answered is how each of these parameters impact the performance of the communication. This analysis is done using the fitness function relationships that we have derived. Using the output of the fitness functions, we can determine quantitively how the communications is affected if the cognitive radio system does not take into affect a certain parameter. For example, a cognitive radio that does not use the signal-to-noise (SNR) ratio will potentially not be able to make an accurate decision regarding anything to do with channel performance. However, with other

parameters such as coding rate, this may not be as clear. Some parameters may require a large amount of processing in order for the cognitive radio to use it for decision making. However, it may turn out that these same parameters have little to no affect on the communication system and are not needed. This sensitivity analysis can save cognitive radio system developers time and cost.

We have selected two different techniques that can be used as a cognitive radio engine. After characterizing the problem domain, we can select these two methods and focus on their specific implementations presented in Chapter 4 and Section 4.3. Independent of the method chosen, we have derived the analytical relationship that relates all of our parameters in Chapter 5. This relationship is used by the selected methods for the optimization. This brings us to the second research question that this dissertation answers: How can the transmission parameters, environmental measurements and radio operating goals be related and represented numerically? Once the optimization techniques are determined, we turn our focus to how this relationship is formed and represented. Using published equations and simulation results we can create these relationships and put them into a form that is usuable by the optimization techniques.

Research Questions and Tasks To summarize the following research questions and their tasks that this dissertation addresses is presented:

1. What is the best way to implement a cognitive radio engine?
    (a) Explore the characteristics and attributes of a cognitive engine to determine what is needed for operation.
2. What is the best set of transmission parameters and environmentally sensed parameters that can be used by a cognitive radio engine?
    (a) Determine a common list of parameters that the majority of wireless radios will be using.

(b) Define a set of radio operating objectives that can be related to the parameters.

3. How can the transmission parameters, environmental parameters and radio operating parameters be related and represented analytically?

   (a) Find published relationships between the defined parameters and radio operating goals.

   (b) Gather data using simulations for radio parameters that cannot be analytically related to the objectives.

4. How much impact does each parameter have in the decision making process of the cognitive engine?

   (a) Explore the affect on the communication system when not using certain parameters.

5. What implementation specific trade offs are present for the two selected cognitive methods?

6. What metrics should be used in order to accurately compare different cognitive methods?

   (a) Explore the characteristics and ideal attributes of a cognitive engine to determine what is needed for operation.

1.3 Research Contributions

This research investigates different methods for implementing cognitive radio engines in a multi-carrier wireless environment. We are focusing on answering the questions of whether AI algorithms such as genetic algorithms or expert systems can be

practically used as the cognitive engine. We begin by defining a standard list of input and output parameters used by the cognitive engine. These parameters represent the controllable transmission parameters and the environmentally sensed parameters use to make decisions. There is no solid consensus of which input and output parameters should be used when developing a cognitive radio. However, the parameters we choose to use several parameters that have been commonly defined in multiple publications.

For cognitive radios to make decisions in any implementation, there must be a relationship showing how the environment is affected as the controllable parameters are modified. We derive this relationship between our defined list of parameters using commonly published equations and simulations that characterize the parameters. We present a relationship represented as a scalar fitness function that is used to score how well a set of transmission parameters affects a specific environment given a set of performance objectives. This fitness function represents the main contribution of this work and is used to implement two cognitive engines, each using a different AI method. The GA was selected because of its ability overcome non-linear problems and adapt to the constantly changing environment with no interaction. The GA is also desirable because of the relatively small amount of memory needed for the processing, but the time needed to produce the output needs to be explored. Expert systems were also chosen as an alternative technique to the GA. The expert system produces a decision very quickly, although the question of how much memory is required to hold the rule base needs to be explored. However, this function is not restricted to be used by only the methods presented here. In general, it can be used to determine if a set of transmission parameters is well matched to a given wireless environment.

We examine the resource usage of both the GA implementation and the expert system implementation, highlighting the trade offs for each method. We show that interesting trade offs exist for the GA implementation between the size of the parameter list

and the time required to determine a solution. We also show that by decreasing the discrete range values used to generate the rule base for the expert system, the accuracy of the decisions is increased. However, this comes at the price of increased memory usage. Uncovering these trade offs allows us to discover the proper parameter settings for each method in order to make practically implementing these methods as a cognitive radio engine a reality.

1.4 Dissertation Organization

The dissertation is organized as follows: In Chapter 2, a brief history of cognitive methods and the role fitness functions play is presented. A detailed literature survey is provided that focuses on the origin of cognitive methods, machine learning, and several popular cognitive methods. Also in Chapter 2, the cognitive radio parameters are introduced, including the transmission parameters, performance objectives, and the environmental channel measurements. In Chapter 3, the genetic algorithm-based cognitive engine is describe in detail. A population adaptation technique used to improve the performance of the GA-based engine is introduced and the simulation results of this engine are presented. Also in Chapter 3, the rule-based system cognitive engine is describe in detail. and the simulation results of this engine are presented. Chapter 4 describes the derivation of the dynamic multi-objective fitness functions. In Chapter 5, simulation results and the analysis on the simulation performance results are presented. Chapter 5 also contains the sensitivity analysis results detailing how transmission parameters can affect the optimality of the system if they are not allowed to adapt.

Chapter 2

Cognitive Radio Overview

2.1 Cognitive Radio

Cognitive radios have received much attention and funding recently as a proposed solution to the spectrum scarcity problem identified by the FCC. The problem being that, although there is a shortage of available frequency bands to license out, the current licensed bands are severely underused in both a time and space sense [4].

Mitola proposed that cognitive radios solve this problem by sensing the environment and autonomously adapting to take advantage of the underused spectrum, while staying clear of the incumbent user's signals [22]. Mitola proposed that the integration of machine learning techniques in radio will allow the technical operation of wireless networks to operate more efficiently. He analyzes the need for a cognitive control system that translates the user needs into commands to the underlying radio functions.

In March of 2005, Vanu, Inc. introduced the first (and presently only), FCC-certified software radio product [23]. The AnyWave^{TM} Base Station is the first base station system that fully implements the base transceiver and base station controller entirely in software, running on a general purpose server and RF front-end. With software-based signal processing and a standards-based architecture, their base station is also capable

of becoming the first commercial cognitive radio on the market.

Currently several research efforts exist to explore the abilities that cognitive radios can provide for both commericial and military use. Open source efforts, such as the GNU Radio [24] and the Virginia Tech OSSIE program [25], allow for wide spread experimentation on software defined radios which in turn provide the ground work for cognitive radios. In order for these efforts to be expanded to cover cognitive radio research, a connection needs to be made between the SDR community and the artificial intelligence (AI) community. AI algorithms such as genetic algorithms, neural networks, expert systems, or case-based reasoning systems can be essentially layers on top of the SDR system to provide that extra layer of intelligence that defines a cogntive radio. However, simply adding the AI layer is not enough to develop the radio. The expertise of a radio engineer needs to be used by the AI architect in order to create the complex relationships between the transmission parameters that define how the cognitive radio engine will operate. This is the primary challenge addressed in this dissertation.

2.2 Adaptive Parameters

Cognitive radios adapt the available transmission parameters in order to achieve a specific performance goal. They do this by combining several adaptation techniques to form a decision making engine with several dimensions of transmission control. Adaptive parameters for wireless systems is not a new research topic. Dynamic power adjustment schemes for wireless systems have been proposed [26, 27]. Adaptive modulation has also become a popular way to adapt a wireless system to a channel to achieve near-optimal throughput [28–31]. Even between transmit power and modulation there exists trade-offs between the system throughput and the system bit error rate (BER) [29].

Combining these different techniques such as adaptive modulation, dynamic power adjustment, dynamic spectral allocation, and even adaptive frame length techniques will provide for efficient communication because all of these advanced techniques are being used. However, the difficulty arises when a system tries to make use and optimize all these techniques that share operating parameters. For example, adaptive modulation is common place even within the IEEE 802.11 wireless networks common to most households. The adaptive modulation techniques implemented in the PHY layer of the hardware monitor the signal-to-noise ratio (SNR) of the communications signal and adjust the power and modulation accordingly in order to achieve the best throughput while still maintaining a useable bit-error-rate (BER). Research on frame length adaptation has also been used to change the value of the framelength in order to achieve a higher overall throughput in low SNR environments. Individually, these techniques can optimize performance for their specific goal. By using multiple techniques at the same time, a single parameter affects different techniques in different ways and a tradeoff is created. Combining these adaptive techniques is the job of a cognitive engine that can input the environment scenario and output the adaptable scenario based upon specific objectives of the system. Figure 2.1 gives a visual representation on how the parameters interact and are used in a cognitive radio.

2.3 Cognitive Engine Techniques

Several methods are available to implement the cognition engine for a cognitive radio. As mentioned earlier, in addition to traditional global optimization techniques, a wide variety of AI technologies including, neural networks, genetic algorithms, case-based learning, reinforcement learning, fuzzy system, expert systems, and pattern recognition exist that can be used as the control for the transmission parameters in a wireless

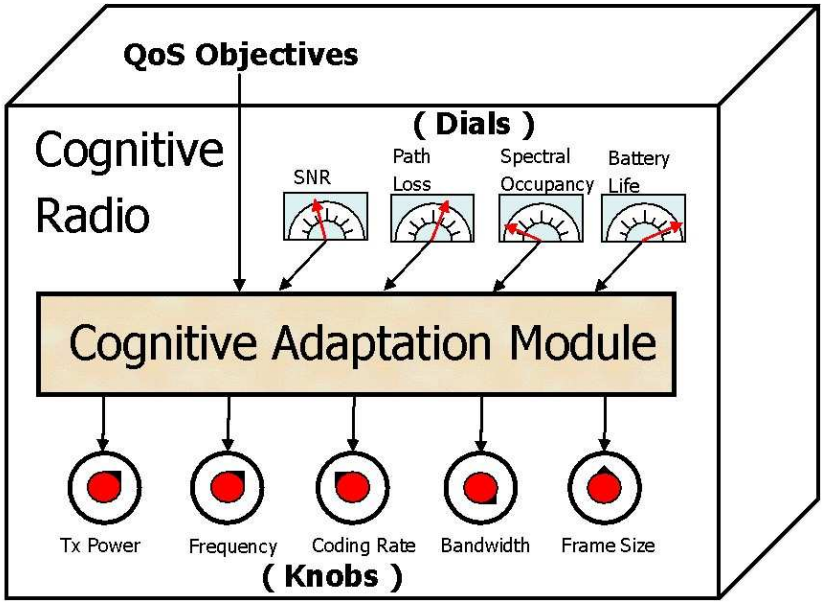

Figure 2.1 Visual representation of cognitive radio knobs and dials

system. Traditional optimization problems typically consist of derived algorithms that determine the exact optimal parameters for a given problem. Several optimization techniques exist for both linear and non-linear problems.

In addition to the non-linear attribute of the wireless communication environment, we also must deal with a very dynamic and constantly changing environment. The cognitive system will be periodically sensing the environment, providing a dynamic and constantly changing the set of inputs to the system. The actual system components may also be changing periodically, in order to compensate for large changes in the environment. With system component changes come changes to the constraints of the problem. For example, changing from one modulation to another may require a changing in the possible coding types, or even a change in the maximum transmit power available. These constraint changes and periodic changes of the input parameters make traditional optimization techniques not suitable for use in this problem domain. Keeping an updated optimization algorithm current to the dynamic parameters and constraints is not

possible. Thus, we need a flexible solution that adapts to the dynamic environment. For this reason, we focus on using AI based techniques.

We have implemented two techniques that can be used as the adapative engines for cognitive radios. Expert systems are built upon a set of rules that define how the system operates. All possible environments are represented in the rule database along with the optimal decision for each environment. This requires an offline analysis of the problem in order to generate the proper rule set. The advantage of this method is the speed at which a solution is found. However, the quickness of the output comes at the price of the large database required to represent the rules. We explore the relationship between the size of the rule base, the correctness of the output, and the time it takes to determine a solution. This technique was chosen as a benchmark against a machine learning technique that does not require a complete database all possible scenarios. We explore the possibility of this complete search space technique being more efficient then the machine learning technique.

As the primary adaptive engine, we have implemented a machine learning technique to determine the optimal solution. For our adaptive engine, we needed a machine learning technique with the following primary characteristics:

- Ability to overcome the local minima in non-linear problems.
- Efficient search with problems containing large search space.
- High amount of system configuration flexibility.
- Low requirement on storage memory usage.
- Little initial setup required to operate.

There are several different surveys of machine learning techniques available [32–34]. We looked at several different choices as to the selection of the adaptation engine

technique. The most commonly referenced techniques being neural networks, genetic algorithms, and symbolic learning techniques such as case-based reasoning systems and rule-based systems. Each technique has its own advantages and disadvantages and we need to find the technique that would fit our needs the best.

The ability to overcome the local minima problem is inherent in all the machine learning techniques we looked at. This problem is primarily present in simple gradient search techniques such as hill climbing. To distinguish between the different techniques we focus on the ability to work with problems with large search spaces and techniques that are flexibly and can be reconfigured on the fly with little code changes. Placing a small amount of focus on the amount of memory usage the technique uses is feasible otherwise a full search space solution would suit our needs. Ideally our solution will be implemented on mobile radios with limited processing and memory resources.

Knowledge in neural networks is learned and stored using a network of connected neurons, weighted synapses, and threshold logic units [33]. Learning algorithms are used to adjust the weights on the synapses in order to classify unknown samples correctly. However, the output of the inductive techniques such as neural networks rely fully on the examples provided during the training phases. These examples may include user feedback or information collected during a relevant feedback stage. In reality, user feedback for the situation addressed here may be limited or noisy. Gathering training examples would require a user to perform tests in the wireless communication area before the cognitive system would ever be deployed and have the user set the optimal parameters based upon the measurements collected. This information would be feed into the neural network in order to train it appropriately. The question of the sample size used to train the neural network is also an on going research question.

Case based reasoning (CBR) systems observe the environment and alter previous solutions to similiar environments [35]. The difficulty in CBR systems is determining

the similarity of cases especially in non-linear environments and how to adapt these case to the current situation. We may observe an environment and have an existing case that is similiar but not close. How can we adapt multiple parameters of this case to better match the current non-linear wireless environment? The effectiveness of CBR systems rely on the case base. Having a larger case database increases the probability of a similiar case, thus increasing the probability of a better final solution. However, holding cases requires storage memory, and in problems with large search spaces the case database is large in order to provide a sufficient amount of similar cases. Similiar to neural networks, a CBR system needs to being with examples from training. Again, practically these inital cases can be difficult to produce. CBR systems are similiar to rule based systems in that they start with initial cases or rules and make decisions based on them. However, rule based systems hold every possible combination that is expected to be seen.

We desire a system that ideally needs no inital setup to operate. Evolutionary techniques posess this characteristic by randomly generating an inital set of solutions and evolving these solutions over time in order to eventually generate an optimal solution [12, 13]. Through the use of mutations and solution combinations, evolutionary techniques are well known for their resilience to convergence at local extrema, unlike traditional optimization techniques. More specifically, a genetic algorithm was choosen over a simulated annealing technique which is very similiar to the genetic algorithm [36]. The primary difference being that GAs work with a population of possible solutions, not just a single point as simulated annealing does. This parallel processing allows the GA to explore different portions of the search space simultaneously. Genetic algorithms operate on the original cost functions, not the derivatives of the cost functions like traditional optimization solutions. This provides a higher level of flexibility when dealing with a highly dynamic environment. For example, changing the com-

plete operation of the system is achieved by subsituting the fitness function used by the genetic algorithm out for another, making this technique very flexibly with respect to code changes in the event of a large change to the environment model. In addition, as explained in Section 2.3.2, genetic algorithms does not require a database of solutions to be stored or any type of storage that grows as the system observes more scenarios. The primary disadvantage of genetic algorithms is the amount of computation needed to determine a solution. We have selected genetic algortihms as the adaptation technique to explore because it matches our needs well, and explore the processing requirements needed to implement this technique within a cognitive radio engine.

The following sub-sections provide background information for each of the selected methods and gives a general overview of how each determines a solution. Sections 6.3.1 and 6.4.1 provide the implementation details for each method.

2.3.1 Expert Systems

An expert system uses non-algorithmic expertise to solve certain problems [8, 9]. Expert systems have a number of primary system components that must interface with people in various ways. Figure 2.2 shows major components and the individuals who interact with the system. The *knowledge base* is the representation of the expertise. Each piece of expertise is typically termed a *rule*, and is represented using an IF THEN format. For example, one rule in our system may be: IF frequency band of interest is currently in use THEN alter frequency. The expertise or the rules are created by the *domain expert*. The domain expert would say specifically what frequency is the optimal to use. Typically, as shown in Figure 2.2, a knowledge engineer is used to encode the experts knowledge into a form that can be used by the expert system. For some projects the domain expert and the knowledge engineer may be the same person, or in our case the information produced by the domain expert actually comes from an equation.

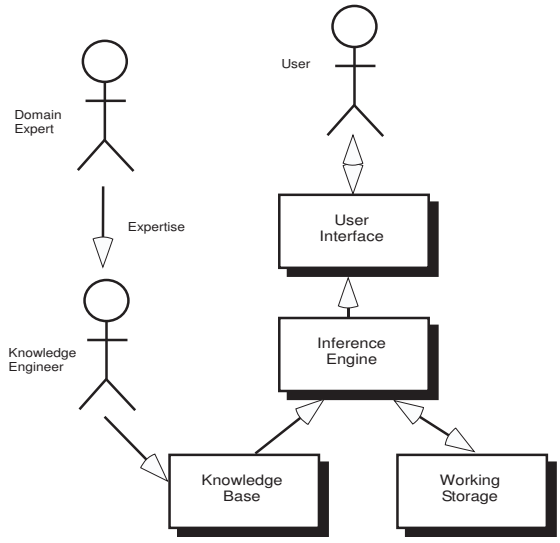

Figure 2.2 Expert System Diagram

The second major component in the system is the *working storage*. This component holds the problem specific information for the problem currently being solved. This information is typically termed the *facts*. For example, a fact may exist that states: frequency band 5.325 GHz is currently in use. This information can be added anytime the system is running. In our case, the facts are brought in from sensors on the wireless system. Information such as the battery life, channel noise figure, and received signal-to-noise ratio can all be represented as facts.

The inference engine is also a primary system component in an expert system. This component includes the code which combines the information from the working storage and the knowledge base to find the solution. This component is accessed by the user through the *user interface*. The user interface is simply the code that controls the dialog between the user and the system.

Many expert systems are implemented as products called expert system *shells*. The shell is the software that contains the user interface, a format for the rules and facts, and an inference engine. A commonly used shell is the CLIPS expert system tool [37].

CLIPS provides a complete environment for the construction of rule based expert systems. The primary reasons CLIPS is used in this research include: knowledge representation, the rule based environment is already implemented, portability, CLIPS is written in C and runs on several different operating platforms, interactive environment, and most importantly, full documentation.

Currently the layout of our rule base uses a one layer approach. This means that the decision is determined after one rule is fired. An example of a one layer rule is as follows: IF sensed parameter THEN set transmission parameters. This approach could also be implemented as a table lookup method because of it's one layer simplicity. Using the table lookup method, the processing time should be relatively faster than the CLIPS system because it is a simple table lookup instead of a fully implemented expert system. However, as this research progresses, the rule base may become more complex, involving several layers of rule dependencies. An example set of two layer rules is as follows: IF sensed parameter1 THEN set temporary variable1, IF sensed parameter2 THEN set temporary variable2, IF temporary variable1 AND temporary variable2 THEN set transmission parameters. In addition to the possible changes in the future, the primary analysis of the expert system will be on the storage resources required for it to represent the complete expertise of the system, or equivalently the total number of rules needed. The total number of rules is equivalent to the total number of entries used in a table lookup method. Given the primary goal of the analysis and the possibility of more complex rules in the future, CLIPS was determined to be an appropriate method to be analyzed as a feasible cognition engine technology.

2.3.2 Evolutionary Algorithms

Evolutionary algorithms, such as genetic algorithms, are biologically inspired search technique to find the solution to optimization problems. Genetic algorithms originated

from the research of Dr. Holland on cellular automata at the University of Michigan [13]. Initially remaining largely theoretical, academic interest started growing in the mid 1980's as the power of processors grew to match the needed power required by GA's. Currently, GA's are used to solve difficult scheduling, data fitting, trend spotting and budgeting problems for several Fortune 500 companies [38]. A primary advantage of genetic algorithms is the fact that they work on a population of solutions. This way, the GA population can explore several parts of the solution space in parallel.

Genetic algorithms have been applied to wireless communications research in several different aspects. In [39], an algorithm for controlling mobile users transmitter power and information bit rate cooperatively in CDMA networks is proposed. A significant enhancement in signal quality and power level was noticed through several of their experiments. Genetic algorithms are also invoked in [40] for finding the optimal weight vectors for the minimum BER of multiuser detector (MUD) for multiple-antenna orthogonal frequency division multiplexing (OFDM) systems. Their results show that the GA-assisted method provides a lower complexity approach than the traditional conjugate algorithm (CG) approach in which they used as a comparison. The processing power made available today allows wireless systems to outperform traditional gradient methods [40]. It has been shown in [41] that the time required by a GA to converge is O(nlogn) function evaluations, where n is the population size. With GA's being used more frequently as optimization methods for wireless communications, the assumption that GA's are computationally infeasible for real-time communication systems is fading.

The basic idea of genetic algorithms is as follows: the genetic pool of a given population of possible solutions or chromosomes, potentially contains the optimal solution to a given adaptive problem. The optimal solution is not active in the current population because its genetic combination is split between several other possible solutions. Split-

ting and combining multiple chromosomes in the population several times can lead to the optimal solution. To perform this procedure, we must define two things. The first is the genetic representation of the solutions, and the second is the fitness function to score the possible solutions. For the genetic representation the standard representation is an array of bits. We explore the fitness function that will be used by the genetic algorithm to determine how well the possible solutions are for a given objective.

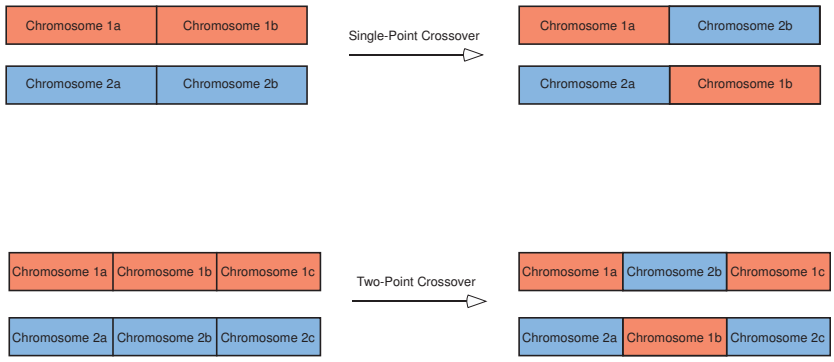

Figure 2.3 Chromosome crossover example

Once the evaluation of the current population is finished, the top ranking pairs of chromosomes are combined to create a new generation of possible chromosomes. Several techniques exist to combine the chromosomes. Figure 2.3 depicts two crossover techniques that are most commonly used. The *one-point crossover* selects a random bit position in the chromosome and all data beyond that point is swapped between the two parent chromosomes, thus creating two new chromosomes. This research uses the slightly more complicated *two-point crossover* technique. This method calls for two points to be selected on the parent chromosomes. Everything between these two points is swapped between the parents, also creating two new chromosomes.

In addition to creating a new generation through the combination of chromosomes, mutation is also a possibility. The purpose of mutation is to allow the algorithm to avoid local minima by allowing the population to randomly mutate and avoid becoming

to similar. This can prevent the slowing, or the complete halting of evolution. A more detailed look at the genetic algorithm methodology is presented in Section 4.2. The full details of the implementation of the genetic algorithm used in this research, along with the genetic algorithm parameter settings used are presented in Section 6.3

Chapter 3

Cognitive Radio Operating Parameters

3.1 Introduction

In developing a cognitive radio control system, several inputs must be defined. The accuracy of the decisions made by an AI method are based upon the quality and quantity of inputs to the system. More inputs to the system make the radio more informed, thus allowing the decision making process to generate decisions that are more accurate. This brings us to the first set of inputs to the system. Environmental parameters are information about the current wireless environment that are used as inputs to the cognitive system. In order for the cognitive engine to make decisions about a certain output, the current wireless environment must be modeled internally. This model is created using environmentally-sensed data received by the system using an external sensor.

Several devices exist to detect characteristics of the wireless environment. The DARPA XG program has hardware for sensing environment characteristics, including spectrum usage [42]. This information is useful if the radio is trying to maximize spectral efficiency. Other sensors detect important characteristics such as: the current noise floor, signal-to-noise ratio (SNR), or determine the BER of the current running configuration. In the following sections, we will propose a list of environmentally-sensed

parameters that will be used to aide in the decision making process of the cognitive controller. Section 3.3 covers the selection of the environment variables used in this dissertation.

Decision variables are also another important input to AI methods. In the cognitive radio case, these variables represent the transmission parameters that can be controlled by the system. Once the virtual wireless environment is created, a set of decision variables is applied to the fitness function and an approximation of how well they meet a set of operation goals is returned based upon the virtual environment. The end result is a quantification of how well a sample set of transmission parameters achieves the set of operation goals. The AI uses this scalar approximation to evolve the system to an optimal set of transmission parameters.

In addition to the environmental data used to model the wireless channel and the transmission parameters, performance objectives must also be determined to define how the system should operate. The objectives of the system are the road map for determining the fate of the system. They provide the means for the controller to steer the system to a specific state. For example, one basic objective is to minimize the bit-error-rate (BER) of the system. This can be done by manipulating the transmission parameters in a certain way as to provide the lowest possible BER given the current environment. This dissertation defines five objectives that represent common wireless radio goals. Section 3.4 covers the selection process of these five objectives.

The following sections detail the selection process of the decision variables used for generating the multi-objective fitness functions. Section 3.4 also covers the selection of the multiple objectives that are used to inform the fitness functions of the optimal direction for scoring.

3.2 Transmission Parameters

Cognitive radios takes advantage of the control parameters made available by the underlying software-defined radio system. These control parameters are input to a fitness function along with the environmental parameters and objectives. This fitness function provides a scalar score that represents how well the control parameters achieve the given objectives. Generating fitness functions to be used by cognitive radio methods require defining a specific list of transmission parameters that must be available to the system. These transmission parameters are equivalent to the control parameters made available by the software radio components. The term *transmission parameters* will be used in this dissertation to refer to the list of parameters that are used to control the individual radio components.

Defining a complete list of transmission parameters and generating a generic fitness function usable by all radios is not possible. Radios are developed for many different reasons and depending upon the application of the radio, each will possess a unique list of parameters. A goal of this dissertation was to define a transmission parameter list large enough to accommodate a large percentage of software radios.

The transmission parameters selected for this research are parameters that would commonly be adjusted to adapt to the channel environment. We have chosen the eight parameters based on an extensive literature survey. The parameters we have chosen have been commonly cited in research literature as as transmission parameters that can be used by cognitive radios to control communication characteristics [21, 43–46].

This work intentionally does not focus on parameters that change on the order of hours, such as transmission formats (e.g. OFDM or CDMA), encryption (e.g. WEP or PGP), or error control techniques (e.g. Turbo coding or convolutional coding). As shown in the results section in Chapter 6, our system can update on the order of 100 ms.

The complete list of parameters used in this dissertation to generate a fitness function is shown in Table 3.1.

Table 3.1 Transmission Parameter List

Parameter Name	**Description**
Transmit Power	Raw transmission power
Modulation Type	Type of modulation format
Modulation Index	Number of symbols for given modulation scheme
Bandwidth	Bandwidth of transmission signal in Hertz
Channel Coding Rate	Specific rate of coding scheme
Frame Size	Size of transmission frame in bytes
Time Division Duplexing	Percentage of transmit time
Symbol Rate	Number of symbols per second

We assume the time scale for modifying the values of these parameters begins at 100 ms, due to our system being able to find a solution in this amount of time. The value of 100 ms is approximately the fastest the genetic algorithm is able to adapt the parameters. This means that the genetic algorithm adaptive technique is not well suited for environments that change at speeds faster than 100 ms because at the time the solution is generated, the environment will have already changed and need new settings.

Although the focus of this dissertation is on the transmission-level parameters listed above and not system-level parameters such as using CDMA or FDMA, those higher order system parameters may still be passed to the cognitive system to allow the filtering out of several possible parameter values. For example, if the cognitive component is informed that the system should be using iterative coding and OFDM modulation, this restricts the modulation type and the channel coding rate possibilities. Chapter 5 goes into detail about how the parameters are represented within the cognitive engine.

3.3 Environment Measurements

Environmental measurements inform the system of the surrounding environment characteristics. These characteristics include: internal information about the radio operating state, and external information representing the wireless channel environment. Both types of information can be used to aide the cognitive controller in making decisions. The environmental variables can be classified into two categories. The first being environment variables that are directly used by the fitness function as primary parameters to the function. An example of this type of parameter is the noise power of the channel which is used in the minimize BER objective function. These parameters directly impact the fitness score of the specific objective. The second class of environment parameters are trigger parameters. These parameters are monitored by the system, and decisions about the objective function are made based upon their values. A good example of this is in regard to the battery life parameter. The system may be monitoring this parameter while is decreases below a specified threshold. In this case, the system may alter the weighting on the objective functions so as to provide a higher weighting on the minimize power consumption objective.

The complete list of environmental parameters used in this dissertation is shown in Table 3.2:

Table 3.2 Environmentally Sensed Parameter List

Parameter Name	**Description**
Path Loss	Amount of signal degradation lost due to the channel path characteristics.
Noise Power	Size in decibels of the noise power.
Battery Life	Estimated energy left in batteries.
Power Consumption	Power consumption of current configuration.
Spectrum Information	Spectrum occupancy information.

The path loss is the reduction in power density of the signal as it travels through space. Path loss may be due to effects such as free-space loss, refraction, diffraction, reflection, and absorption. Path loss can also be influenced by the terrain and environment. The noise power parameter informs the system of the approximate power of the noise in decibels of the measured power referenced to one milliwatt (mW). Battery life and power consumption are both internal parameters. These parameters are used to determine when the system should place more emphasis on minimizing the power. The primary external parameter is spectrum occupancy information. This parameter consists of information from cognitive radios within the local network identifying the spectral location of other signals in frequency bands of interest. This information is used to improve the spectral efficiency of the transmission and the spectral occupancy of the frequency band [47, 48].

The trigger parameters represent an important characteristic of cognitive radio systems. Much research focuses on the active parameters that are used in making transmission parameter decisions, and the objective steering trigger parameters are often overlooked. This work defines these important parameters and shows how they can be integrated into cognitive radio systems using objective weights to control the instantaneous goals of the communication system.

3.4 Performance Objectives

In a wireless communications environment, there are several desirable objectives that the radio system may want to achieve. We define five objectives for the fitness function in order to guide the system to an optimal state. The five objectives are given in Table 3.3.

Minimizing the BER is a common communications goals in todays wireless world.

Table 3.3 Cognitive Radio Objectives

Objective Name	Description
Minimize Bit-Error-Rate	Improve the overall BER of the transmission environment.
Maximize Throughput	Increase the overall data throughput transmitted by the radio.
Minimize Power Consumption	Decrease the amount of power consumed by the system.
Minimize Interference	Reduce the radios interference contributions.
Maximize Spectral Efficiency	Maximize the efficient use of the frequency spectrum.

This objective represents minimizing the amount of errors in relation to the amount of bits being sent. In general, this objective represents improving the communications signal of the radio. Maximizing the throughput deals with the data throughput rate of the system. Emphasis on this objective improves system throughput. Minimize power consumption is self explanatory and is used to direct the system to a state of minimal power consumption. Trade-off analysis between minimizing BER, maximizing throughput, and minimizing power consumption are shown as the preliminary results in Section 5.4.

The last two objectives focus on the spectral domain of wireless communications. Minimizing interference encompasses avoiding areas of the spectrum with a high noise floor, or areas with high possibility of interference being present. Similarly, emphasis on the maximize spectral efficiency objective would reduce the spectral space used by the transmitted signal.

In order to direct the system to a specific solution, we must attach preference information to each objective. Otherwise, simply minimizing both BER and power will result in a set of solutions instead of a single solution. This is because minimizing BER and minimizing power will have different solutions. Thus, the objectives must also contain a quantifiable rank representing the importance of each. This will allow the fitness

function to characterize the trade-offs between each objective by ranking the objectives in order of importance. Several approaches exists for determining the preference information of a set of objectives [14]. We have decided to use a weighted, aggregate sum approach where each objective receives a weight representing its importance. We selected this method primarily because of the simplicity of implementation within the genetic algorithm technique and the ability of control that this method provides to the system. This method is detailed in Section 5.3, along with other similar methods and more detailed reasoning behind its selection.

3.5 Summary

This chapter presented a well-defined list of common parameters for cognitive radio systems. These parameters included environmentally measured parameters from sensors within the system, and internal operating information providing measurements about the internal state of the radio. This information is used in conjunction with the radio objectives to determine the appropriate transmission parameters to use for communication. Multiple objective problems have difficulties when all objectives are trying to be achieved at once. The primary difficultly being determining the complex relationships and trade-offs between the parameters the the multiple objectives. Determining a single search direction for the system is also a common difficulty in multiple objective problems. We defined five performance objectives for a wireless communication in which several objectives conflict with regards to maximizing their performance. These conflicts create trade-offs between the transmission parameters to meet a given performance constraint. In order to quantify these trade-offs, the objectives must have some type of preference ranking. The fitness function must also input objective preference information to solve the problems of parameter conflict.

Chapter 4

Adaptive Cognitive Radio Engine Techniques

4.1 Introduction

Genetic algorithms are a class of artificial reasoning whereby the search is performed in a manner similar to genetic evolution. In general, solutions to a problem set are represented by binary strings. These strings then are allowed to act in a manner similar to genetic growth; strings which are considered 'good' split and recombine with other good strings to form new solutions, while 'poorer' strings are allowed to 'die' out of the solution set. This decision is made by the fitness function which inputs the parameters and outputs a score based on the specific goals of the radio. Strings undergo a process called mutation, i.e., a random flipping of bits, to help prevent local minimization from occurring. Genetic algorithms are typically used as a method of problem optimization [12, 49]. However, given its random nature, fast computation time, and ability to spontaneously generate unique solutions, genetic algorithms are an appealing candidate for cognitive radios. Input and output parameters can easily be mapped to a binary form and the size of the genetic population is customizable to space avail-

able within any given configuration [12]. Genetic algorithms are used mainly when the search space is too large to be simply brute force search to determine the optimal parameter set. In this disseration the binary chromomsomes used in the genetic algorithm consists of the eight parameters defined in Table 3.1.

As an alternative technique to genetic algorithms, we choose to implement a drastically different approach to selecting the optimal transmission parameters in a wireless device. Genetic algorithms rely on the ability to use a small amount of memory and large amounts of processing to evolve to a solution. Rule-based systems, in contrast, use large amounts of memory and a small amount of processing to make decisions.

A rule-based system (RBS) uses a simplistic model based upon a set of if-then statements to implement an expert system. Expert systems are widely used in many fields with the primary concept being that the knowledge of an expert is coded into the rule set. When the expert system comes across a data set, it should behave the same way as the expert who populated the database. We want to explore the feasibility of using such a RBS in the context of cognitive radio decision making. Rule-based systems are at a disadvantage when the rule base becomes large because it becomes hard to manage and the rule base itself may consume too much memory. Rule-based systems are typically also constrained by discrete values. This can be overcome by using fuzzy sets and defined ranges for the rules to match on. However, this then requires a specific range to be determined for each rule. In the following sections, we explore the implementation of both of these techniques in the context of cognitive radios.

4.2 Genetic Algorithms

The methodology of a genetic algorithm can be broken up into four seperate stages. The first stage is the initialization of the population. Initially the population is randomly generated to form the first generation of possible solutions. The choice of population size is based loosely on the specific problem we are dealing with, however a common set of genetic algorithm settings has been defined and used in several GA implementations with slight variations [50, 51]. In our case we use a population size of 100 chromosomes. Traditionally the initial generations is selected at random. However, as we show in Section 4.2.1, we can take advantage of previous runs and seed the initial generations in order to achieve better performance when compared to randomly generated inital populations.

The second stage is the selection stage. During each generation a proportion of the population is selected to breed a new generation. Individual solutions are ran through a fitness function that assigns a fitness score to the solution representing its value. Several selection methods exist, such as tournament selection, stochastic remainder selection and roulette wheel selection that use the fitness scores to select the solutions that are too be used to form the next generation of solutions. We chose to use the stochastic remainder selection method. We chose this method mainly because of its popularity and the large amount of research associated with it [12, 52, 53]. This method uses the ratio between the fitness of an individual solution and the average fitness of the population to determine the probability of the solution moving onto the reproduction stage.

The third stage is the reproduction stage. In this stage, the next generation of solutions is generated from the previously selected group of solutions. This process is completed through genetic operators such as *crossover* and *mutation*. Crossover is a

method of combining two possible solutions to create a new solution. Mutation is the process of randomly mutating a solution, typically randomly flipping a bit in the genetic sequence. Mutation provides the means for the GA to avoid local minima by preventing the solutions from becoming too similar to each other, which can slow or even stop evolution. For each new solution in the next generation a pair of solutions is selected to be the "parents." One point and two point crossover are two possible approaches of combining the "parents." One point crossover selects a random point in the genetic sequence in which the "parents" swap all data beyond the selected points Two point crossover is similiar except that two points are selected and all data between the two points are swapped. Dejong shows in [50] that two point crossover provides a better mechanism for combining and mixing the chromosomes and produces better results then the single point crossover technique. Along with the crossover function, each new solution has a typically small chance to have a bit mutated. These processes result in the next generation of solutions. Generally, the average fitness has increased since mostly the higher scoring solutions are selected to breed, along with a small proportion of lesser fit solutions to provide for a more diverse search.

The final stage is the termination stage. The genetic algorithm process detailed previously continues until a termination condition has been reached. Common termination conditions include:

- A solution that satifies a minimum criteria is found.
- A fixed number of generations is reached.
- A specified computation time is reached.
- The fitness scores have plateaued such that succesive generations show no improvement.

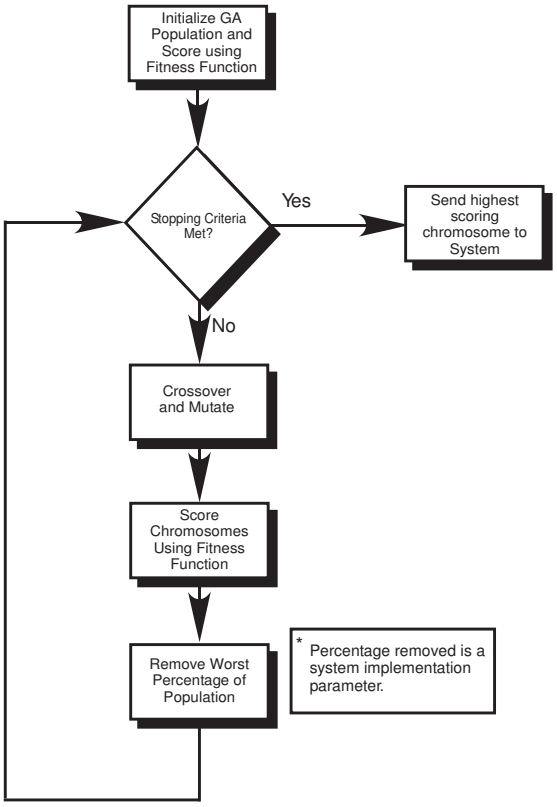

Figure 4.1 Genetic Algorithm System Process Flow

- Combinations of the above conditions.

In this disseration we have chosen to run the genetic algorithm until a fixed number of generations is reached. We choose this condition so we don't restrict ourselves to certain time requirements and we wanted to explore the highest value of fitness the algorithm can reach. In this work we are exploring the different combinations of equations and settings. Future work may include implementing a stopping condition that terminates once the fitness is within a certain threshold of the desired fitness value. We choose not to do this because stopping at a specified fitness won't allow us to observe the "best" solution we could possibly find. The same reasons apply for why we chose not to stop after a specific amount of computation time. A more efficient approach would be to stop once the algorithm plateaus, however, in order to compare different fitness function weights and other methods, we wanted to use a static number of generations to provide for a fair comparison between the different profiles. Although we have choosen to stop after a fixed number of generations, the results we present in Chapter 6 show the processing time per generation. Using this time per generation we can calculate the time needed to run the genetic algorithm for a shorter number of generations if needed.

Figure 4.1 shows the general flow for the genetic algorithm process that we implement in this research.

4.2.1 Population Adaptation Enhancement

In the area of wireless communications optimization, quality-of-service (QoS) requirements may limit the time required to determine a decision. To facilitate these QoS requirements, typically the GA engine would be required to terminate after a predefined number of generations have been executed, in order to guarantee a decision in a set amount of time. However, this does not guarantee that the GA has converged to

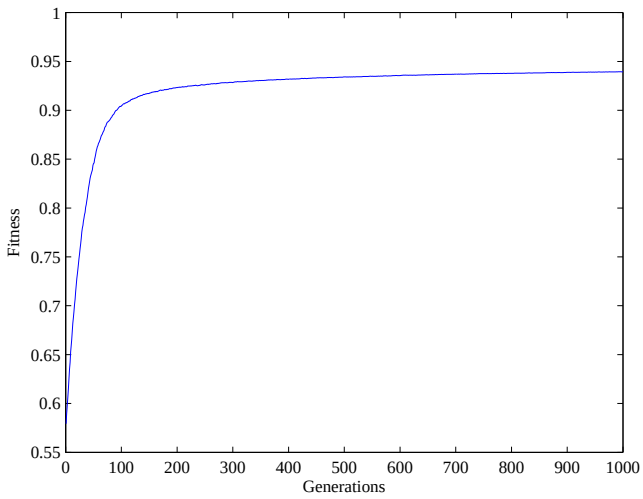

Figure 4.2 Fitness convergence for a standard GA implementation

a adequate set of transmission parameters. Our previous work has shown that a standard GA used in a multi-carrier system using a small number of parameters requires a significant amount of time for determining an optimal solution. Figure 4.2 shows the fitness convergence for a 16 channel GA-based implementation operating in emergency mode (i.e. emphasis on bit-error-rate). This graph provides information about how quickly the system converges to the optimal decision. For a complex cognitive radio system with a large number of parameters, using a standard GA-based implementation becomes infeasible since the time needed to complete one generation increases as the system complexity increases. We show this to be true in Chapter 6 by increasing the number of channels, thus increasing the size of the chromosome and making the system more complex.

We modify the initialization stage of the GA algorithm that enables the engine to take advantage of previous measurements and decisions in order to improve the convergence time of the algorithm. Using the assumption that the wireless channel environment changes slowly, we can seed the initial generation of the GA algorithm with

chromosomes from the final generation of the previous GA cycle. This technique biases the initial generation to the final decision of the last GA cycle. We show that by seeding the initial generation we can achieve increased convergence times, thus decreasing the amount of processing needed to achieve the same results as the standard GA implementation.

Using information about the problem domain, we can use initial seeding techniques to improve the operation of a GA algorithm [54]. In a quasi-static wireless channel environment, we can assume the environment parameters are changing slowly. In this case, the results from the previous evolutions in the GA can be utilized by seeding a percentage of the initial generation with chromosomes from final generation of the last cognition cycle. Doing this will bias the initial generation toward the last decision. Depending on the amount of environmental variation, this seeding will improve the convergence rate of the GA algorithm.

In our population adaption technique, the change in environment parameters can be characterized by a figure of merit called the *environmental variation factor* (EVF), which is used to determine the amount of seeds to be utilized from the previous cognition cycle. The EVF represents the amount of variation that has occurred in the environment since the last cycle of environmental sensing. The technique can significantly reduce the number of generations required for the convergence of the cognition cycle by using the EVF to determine the amount of seeding.

The EVF is defined as the weighted sum of the percentage changes in the environment parameters, which is the single metric for determining the changes in the environmental parameters. For example, an EVF of 0.20 tells us that the average variation over all the environmental parameters was 20%. For our simulations, we restricted the variation in the environment in such a way that the environment could only worsen with respect to the fitness. Had we not restricted the EVF in this way, certain scenarios where

the noise decreased significantly caused uncharacteristicly high fitness scores due to the BER fitness function being normalized to a BER of 0.5. Using this information about the variation in the environment, we can select the appropiate amount of population seeding for the GA algorithm. The assumption is that at low values of EVF, higher seeding percentages will improve the convergence rate of the GA. This is due to the fact that a low EVF represents a wireless environment that has only slightly changed. In this case, the previously determined decision will be a better starting point than a randomly selected population of decisions. However, in the case of a large change in the wireless environment, or a high EVF, the initial population should be more diverse to enable the algorithm to explore a larger portion of the search space. One common situation where a high EVF may be detected is in the case of wireless channels experiencing deep fades. In this case, if the seeding percentage is too high, the initial population not be diverse enough to evolve to the globally optimal decision.

4.3 Rule Based System Framework

This section provides more details into the implementation of the RBS and focuses on each of the different sections that make up the RBS. As seen in Figure 2.2, a RBS consists of several different components. We begin with the domain expert which can be defined as the fitness function derived in Chapter 5. In order to for the fitness function to be used as the expert, we needed to perform a full search space run over all possible parameters using the fitness function, and find the optimal transmission parameters for each possible environment.

The environment parameters that are used in the rules consist of the noise power and path loss of the wireless channel, and the objective weightings of the system. Using the sensed values of these variables the RBS asserts the rule that provides the optimal

fitness function. The first step in the development of the RBS system is to create the rule base for the system. This is done by determining the optimal transmission parameters for each possible combination of environment variables.

Table 6.2 gives the operating values of the noise power and path loss. Using only these two parameters we have a total of 100 rules. Factoring in the five objectives weightings, each ranging range from 0.00 to 1.00, this gives 10 billion possibilities. However, we recognize that looking at all possible combinations of parameter weightings isn't needed and instead focus on specific combinations of weightings that represent interesting scenarios. The combinations we have choosen are defined in Table 5.7. We chose these scenarios in such a way that four of the performance objectives would each have a major emphasis on them. These combinations give the cognitive adaptation engine a diverse selection of performance objectives to simulate. With only these four scenarios we have a total of 400 rules that need to be generated in order to implement a RBS for testing.

A major advantage of the RBS is the ability to generate the rules offline when time is not a key factor. This is a good thing because finding the optimal transmission parameters for each of the environment scenarios requires a complete search of all combinations of transmission parameters to find the combination with the largest fitness score. For a single objective scenario, there exists approximately 600 million possible combinations of parameters that need to be search for an optimal value.

We begin this search by implementing the fitness function in MATLAB and verifying the output of the function is identical to that of the fitness function implemented in the genetic algorithm. To verify this, several debug statements in the GA code were inserted that outputted actual MATLAB code that could easily be executed immediately to verify the MATLAB version of the fitness function achieved the same results. Once it was determined that the fitness functions were identical, the fitness function

was ran for all possible combinations of transmission parameters. This consisted of 10 for loops, two days of processing and about 6 Gigabytes of memory to hold the large 10 dimensional array for a single performance objective combination.

The large array is indexed by the radio parameters and holds the fitness values for each combination. Using a few simple MATLAB commands the maximum fitness value for each environment combination could be found, along with the proper indexes that represent the actual transmission parameters. From these we create the rules for the RBS. This processes was automated by a MATLAB function and can be view in Appendix A.

4.3.1 CLIPS

The inference engine as shown in Figure 2.2, infers the proper rule to be asserted. We have choosen to use the C Language Integrated Production System (CLIPS) expert system [37] as the engine. CLIPS dates back to 1985 where it was developed by NASA at the Johnson Space Center with the intent of gaining insight and knowledge into the construction of expert system tools and to lay the groundwork for replacement tools for the current commericially available tools. Eventually, the CLIPS system's low cost and great performance made it the ideal tool the development of expert systems and eventually became available to groups outside of NASA in 1986. Today CLIPS is widely used in the government, idustry, and acadamia. We have choosen CLIPS due to the fact that is it written in C and in the future this can allow us to implement the RBS into existing cognitive engines easily. In addition, CLIPS's interactive environment makes it very easy to use and provides a simple interface for me to add rules and facts manually when debugging or CLIPS is able to read in large databases of rules and facts created from MATLAB. In the end, the primary reason CLIPS was choosen was the simple interface, the C implementation, the documentation, and the ability to modify

the source or integrate with solutions we already have.

We run the CLIPS shell under a linux environment and begin by importing the rules generated by our MATLAB fitness function. Once these rules are imported, the database exists within the engine and all that is left is to assert facts about the environment. For example, we may assert three facts stating that the noise power is -114 dBm, path loss is 80 dBm, and we are operating in the emergency mode, where we want to focus on minimizing the bit-error-rate of communications. Once these facts are stated in the system, CLIPS will match the facts to a rule that exists in the database, and assert another fact that provides the optimal transmission parameters for the given environment. Our goal is to analysis the implementation issues with the RBS so we did not integrate the CLIPS system into a standalone program. Our tests were all ran within the actual CLIPS shell.

Chapter 5

Multi-Objective Fitness Functions

5.1 Introduction

This chapter will cover the multi-objective fitness function problem representation and describe the analytical techniques that will be used to generate the fitness functions. Section 5.2 provides an overview of multi-objective fitness functions and the challenges that must be overcome to generate an accurate function. Section 6.2 describes the representation of the transmission parameters and the environmentally-sensed parameters. The description of how the performance objectives are used to determine the search direction of the evolutionary algorithm will be presented in Section 5.3. In Section 5.4, a detailed description of the analytical techniques that are used to derive the fitness functions is presented along with the fitness functions generated using a subset of decision variables. The final section presents a summary of the chapter.

5.2 Multi-Objective Fitness Function

5.2.1 Problem Statement

In general, a multi-objective fitness function problem can be presented as trying to determine the correct mapping of a set of m parameters to a set of N objectives. This can be seen algebraically as:

$$\vec{y} = \langle f_1(\vec{x}), f_2(\vec{x}), f_3(\vec{x}), \ldots f_N(\vec{x}) \rangle \tag{5.1}$$

subject to

$$\vec{x} = \langle x_1, x_2, x_3, \ldots x_m \rangle \in X$$

$$\vec{y} = \langle y_1, y_2, y_3, \ldots y_N \rangle \in Y$$

where x is the set of decision variables and X is the parameter space, and y is the set of objectives with Y as the objective space. In the case of a multiple objective evolutionary algorithm, each $f_i(x)$ represents the fitness function for a single objective. The goal is to combine them to get a single fitness function, $f(x)$, taking into account all parameters and objectives.

In real world problems, such as the problem addressed in this thesis, the objectives under consideration might conflict with each other. For example, minimizing power and minimizing BER simultaneously creates a conflict due to the single parameter, transmit power, affecting each objective in a different way. Determining the optimal set of decision variables for a single objective, e.g. minimize power, often results in a non-optimal set with respect to other objectives, e.g. minimize BER and maximize throughput. The optimal set for multiple objective functions lie on what is known as

the *Pareto optimal front* [14–16]. This front represents the set of solutions that cannot be improved upon in any dimension. The solutions on the Pareto front are optimal and co-exist due to the trade-offs between the multiple objectives. A graphical example of a Pareto front, using a simple cognitive radio parameter scenario is shown in Figure 5.1.

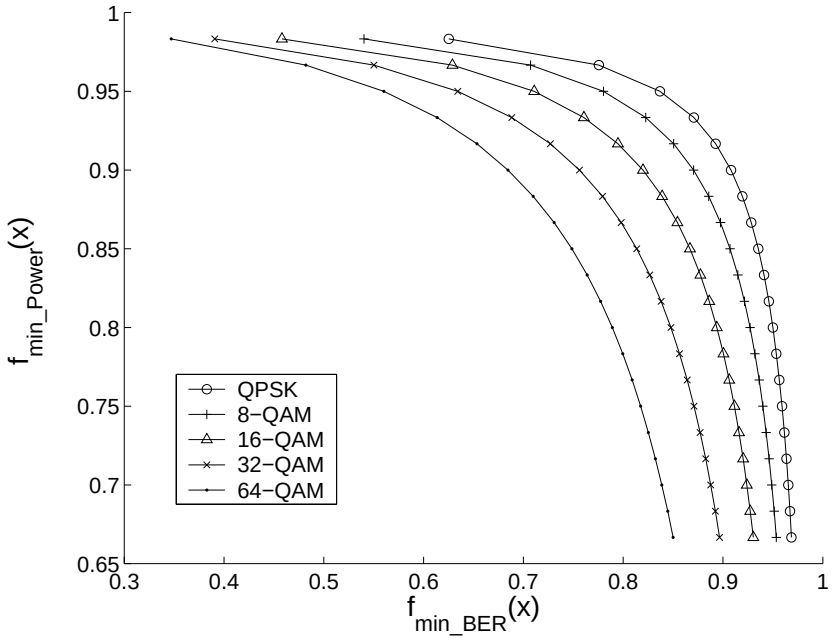

Figure 5.1 Pareto Front Trade-off

The x-axis in the figure represents the score of the single objective fitness function for minimizing BER in the case of several modulation types, while the y-axis is the score for the single objective fitness function for minimize power. The parameter *x* represents the decision variable vectors used as inputs to the fitness functions. In this case, transmit power and modulation were used as decision variables. For each curve, as the fitness score for minimize power decreases, the score for the minimize BER objective increases. This trade-off represents the core of the multiple objective optimization problem. The QPSK curve represents the Pareto front, because no parameter set on that curve can be improved upon to gain a better objective score in respect to both objectives. The other modulation curves represent the dominated solutions to

the bi-objective optimization problem. The fitness functions used for this example are derived in Section 5.4.

In this thesis we develop a set of fitness functions using the set of parameters defined earlier, that are used by cognitive radio engines to determine a single optimal transmission parameter solution. In Figure 5.1, several parameter sets lie on the Pareto front, however the fitness function must provide a search direction that leads the system to a single solution. We use a weighted sum optimization approach to give the fitness function the ability to focus in on a single parameter set. Using weights allow the fitness function to instantly change the global objective of the system. Section 5.3 details the weighted sum formula and how it applies to our fitness scenario.

Before we go into detail about the objective weighting, we must describe how the parameters of the cognitive radio are represented as inputs to the fitness function. Thus, a list of parameters used by the fitness function and their possible ranges are defined in Section 6.2.

5.3 Fitness Objective Representation

Ideally the fitness function must be able to guide the system to one optimal parameter set. A cognitive radio must perform an action based on a single set of parameters, which should be selected from the Pareto front according to some preference information. Preference information, or objective weighting, is used to rank the objectives in order to help the fitness function guide the evolutionary algorithm to one optimal solution.

In addition to needing preference information for each objective, the scalarization of the objective vector is also necessary. Evolutionary algorithms need scalar fitness functions that provide a single scalar value for the given parameter set. In many optimization

problems, when no global criteria, e.g. goals, for the parameters exist, objectives are often combined, or aggregated, into a scalar function. This aggregation optimization method has the advantage of providing a single scalar solution for the fitness function. As a result, this requires no extra interaction with the evolutionary algorithm to determine the optimality of a given parameter set.

There have been several approaches to the optimization of aggregated functions. Weighted sum approaches are presented in [17, 18]. The weighted sum approach attempts to minimize the sum of the positively normalized, weighted, single objective scores. In [55], target vector optimization was developed. Target vector optimization requires a vector of goal values. The optimization is driven toward the shortest distance between any candidate solution and the goal vector. Goal programming was also studied by several authors [56, 57]. In goal programming one objective is minimized while constraining the the remaining objectives to be less than the target values. However, choosing appropriate goals for the constraints can be difficult. Goal programming has also be shown to not generate the Pareto front effectively when the number of objectives is greater than two.

This research proposes to use the weighted sum approach. This methods suits the cognitive radio scenario well since it provides a convenient process for applying weights to the objectives and more importantly provides a single scalar value. Using the weighted sum approach, we define a multiple objective fitness function of the parameter set solution x by the following weighted sum of N objectives:

$$f(x) = \sum_{i=1}^{N} w_i f_i(x) \tag{5.2}$$

with $w_1,\ldots,w_n$ satisfy the following constraints:

$$1 \geq w_1 \geq 0 \text{ for } i = 1,2,\ldots,n \tag{5.3}$$
$$w_1 + w_2 + \ldots + w_n = 1$$

When the weighting for each objective is constant, the search direction of the evolutionary algorithm is fixed. This is the intended property when trying to find a single optimal solution for a given environment. However, changing the objective weighting means the fitness function will immediately start steering the evolutionary algorithm to a new solution. For example, take the case in which a radio is operating in a minimize BER mode. In this mode, the fitness function will give higher scores to parameter sets providing a high transmit power. This is because the weight on the minimize BER objective is the largest. Suppose that the radio then detects low battery power. At this instance, it changes the objective weighting to reflect an emphasis on minimizing power. This is done by reducing the weights on other objectives while at the same time increasing the weight of the minimize power objective. Once the weights change, the fitness function will instantly start giving higher scores to parameter sets which provide for lower power transmission. This is the primary attribute that allows the objective weighting to dictate the goal state of the radio. It also allows for a dynamic system to instantly switch operating goals by simply modifying the objective weighting vector.

Figure 5.2 shows the previous example. The search direction $w^a[.]$ corresponds to a minimized power weight vector in the 2-D objective space. The search direction $w^b[.]$ corresponds to a minimized BER weight vector in the 2-D objective space. As the objective space increases, so does the dimension of search space for a solution. We propose to develop a fitness function using a five dimensional objective space.

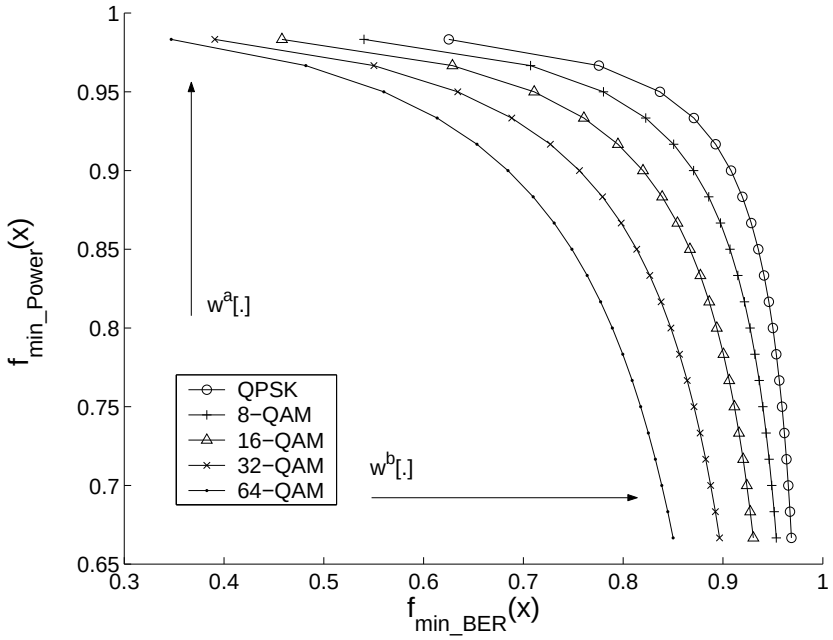

Figure 5.2 Search Direction Example

5.4 Parameter Trade-off Analysis

5.4.1 Single Objective Goals

The weighted sum approach allows us to develop a single objective function for each objective and combine them to create a multiple objective function. To develop the single objective functions, we must determine the dependence relationship between each objective and the set of parameters defined in Section 3.2. The complete table of relationships is displayed in Table 5.1.

Table 5.1 Objective and Parameter Relationships

Objective Name	**Related Parameters**
1 Minimize Bit-Error-Rate	P,PL,m,M,B,R_c,S,N
2 Maximize Throughput	m,M,B,R_c,L,TDD,R_s
3 Minimize Power Consumption	P,M,m,B,R_c,TDD
4 Minimize Interference	P,B,f_c,TDD
5 Maximize Spectral Efficiency	R_s,B,m

Shown in Table 5.1 are the parameter dependencies for each objective. An important task in this research is deriving the relationship between each performance objective and the parameters. Completing required a complex mathematical analysis of the closed form solutions of each objective, using the defined parameters and their potential range of values. In addition to the weighting constraints imposed on the individual functions, each single fitness function score must be normalized between the same range. Otherwise, if fitness function A outputs scores from ranges [0,1] and fitness function B outputs scores from [0,x] where $x > 1$, then the global fitness function would show a bias to function B due to the larger output range. The outputs of the functions developed in this research are all normalized to the range [0,1]. With this output limitation defined, the following subsections describe the individual fitness functions representing the performance objectives described in Section 3.4.

5.4.1.1 Minimize Bit-Error-Rate

Table 5.2 Minimize BER Cognitive Radio Parameters

Transmission Parameters	**Environmental Parameters**
Modulation Type	Path Loss
Modulation Index	Noise Power
Bandwidth	Received Signal Strength
Symbol Rate	
Transmit Power	

One of the most common goals in wireless communications is to get an error free signal, or to minimize the bit error rate of the transmission. Determining the theoretical bit error rate depends on several transmission parameters including the transmit power, modulation type, modulation index, signal power, bandwidth, and noise power.

The most important factor in determining the BER of the system is the channel type and modulation in use. Each modulation and channel type combination uses a different

formula to determine the BER of the system. Another important factor in determining the BER of the system is the ratio of the energy per bit (E_b) to the noise power spectral density (N_0). This ratio essentially is a measure of the amount of energy per bit to the amount of noise power in the system, or the basis of the signal-to-noise ratio. Each of the formulas uses the E_b/N_0 ratio in deriving the BER.

To determine the E_b/N_0 ratio which we will now denote as γ, we start by using the received signal power, S. We know S is equal to the transmitted signal power that is affected by the path loss. From S, we can get the amount of energy per symbol by using the symbol rate, S/R_s. This can then be used to get the energy per bit by dividing by the modulation index or the number of bits in each symbol as shown in Equation (5.4).

$$E_b = \frac{S}{R_s * m} \quad (W/b) \tag{5.4}$$

The total noise power spectral density is simply the noise per Hertz and is computed using Boltzmann's equation.

$$N_0 = k_b * T \quad (J) \tag{5.5}$$

$$N = N_0 * B \quad (W * Hz) \tag{5.6}$$

where k_b is Boltzmann's constant (1.38 x 10^{-23} J/K), T is the system noise temperature (290 K), B is the channel bandwidth, and N is the total measured noise power. The final equation for the value of γ that the BER functions uses is given by:

$$\frac{E_b}{N_0} = \gamma = 10\log_{10}\left[\frac{S}{R_s * m * N_0}\right] = 10\log_{10}\left[\frac{S}{R_s * m}\frac{B}{N}\right] = 10\log_{10}\left[\frac{S}{N}\right] + 10\log_{10}\left[\frac{B}{R_s * m}\right] \quad (dB) \tag{5.7}$$

The following equations give the BER of QAM, PSK, and FSK, using a gray-coded

bit assignment and assuming an AWGN channel model.

For a BPSK signal constellation, the probability of a bit error is defined as [7]:

$$P_{be} = \mathrm{Q}(\sqrt{\gamma}) \tag{5.8}$$

Whereas for M-ary PSK, the probability of a bit error is given as [7]:

$$P_{be} = \frac{2}{\log_2(m)}\mathrm{Q}\left(\sqrt{2 * \log_2(m) * \gamma} * \sin\frac{\pi}{m}\right) \tag{5.9}$$

For M-ary QAM, the probability of a bit error is defined as [7]:

$$P_{be} = \frac{4}{\log_2(m)}(1 - \frac{1}{\sqrt{m}})\mathrm{Q}\left(\sqrt{\frac{3 * \log_2(m)}{m-1}\gamma}\right) \tag{5.10}$$

A more detailed look at the forumlation of the BER functions in Appendix B. We also show the probability of bit errors in Raleigh fading channels along with several other modulation types in Appendix A.

The goal is to create a fitness function with a valid output range of between 0 and 1. To do this, we normalized the BER against it's worst case of 0.5 and took the log base 10 of the BER value in order to provide a linear scale based on the exponent of the BER. Otherwise, higher values of BER would dominate over very small values. Equation (5.11) provides the final objective function for minimizing BER:

$$f_{min\_ber} = 1 - \frac{\log_{10}(0.5) - \log_{10}(P_{be})}{\log_{10}(0.5) - \log_{10}(10^{-6})} \tag{5.11}$$

where P_{be} represents the probability of a bit error or BER for a given modulation scheme and a given channel type normalized to the worst possible BER value of 0.5 and divided over the total possible range of BER values selected. We chose 10^{-6} to be the best case

BER and all experimental values seen below this value will be rounded up to 10^{-6}.

5.4.1.2 Maximize Throughput

Table 5.3 Maximize Throughput Cognitive Radio Parameters

Transmission Parameters	Environmental Parameters
Modulation Type	
Modulation Index	
Bandwidth	
Symbol Rate	
Coding Rate	
Time Division Duplexing	
Frame Length	

The throughput definition we use is equivalent to the goodput, or the amount of good information recieved at the receiver. This is in contrast to the amount of information sent by the transmitter. This less complex definition is used in order to avoid complicated throughput calculations dealing with information bit errors and retransmissions. In addition, we assume only block coding is being used.

Maximizing throughput is useful in a variety of scenarios. Specifically, multimedia environments that stream audio and video would place a large weighting on maximizing throughput. For this objective, we use a theoretical model to calculate the fitness score for an ideal transmission environment. Determining the theoretical throughput of a system depends on the bandwidth in use, coding rate, modulation index, framesize, and percentage of transmit time. The bit error probability plays the major role in determining the throughput degradation of the system. To determine the throughput, we use the probability of a packet error:

$$P_{pe} = 1 - (1 - P_{ber})^{L} \tag{5.12}$$

The total throughput of the system is affected by the number of packet errors, or the total data lost in transmission. Also in consideration is the MAC layer parameter framelength. At high levels of SNR, transmission will be relatively error free with respect to lower levels of SNR, and large framesizes can be used in order to lessen the amount of MAC layer overhead being transmitted. At low levels of SNR more frames will need to be retransmitted as they have a higher chance at failing MAC layer CRC checks due to a bit error. The larger the framesize that fails the check, the larger the throughput that is lost every time a frame is thrown away. Research has shown that by decreasing the framesize during periods of low SNR, significant throughput savings can be achieved along with savings in power consumption associated with retransmitting frames [58]. Equation (5.13) shows the derived equation that gives the relationship between framesize, bit-error-rate, and good throughput, G:

$$G = m * R_s * \frac{L}{L+O+H} * (1-P_{ber})^{(L+O)} = R_b * \frac{L}{L+O+H} * (1-P_{ber})^{(L+O)} \quad (5.13)$$

where R_b represents the raw bit rate of the system in bits per second. H is the MAC and IP layer overhead at a value of 40 bytes and O represents PHY layers overhead at 52.5 bytes. L represents the framelength size in bytes, and P_{ber} is the probability of a bit error.

In addition to the previous parameters, we also take into account block coding and the time division duplex parameters. After normalizing we get the final single objective function:

$$f_{max\_throughput} = \frac{L}{L+O+H} * (1-P_{ber})^{(L+O)} * R_c * TDD \quad (5.14)$$

5.4.1.3 Minimize Power Consumption

Table 5.4 Minimize Power Consumption Cognitive Radio Parameters

Transmission Parameters	Environmental Parameters
Modulation Type	
Modulation Index	
Bandwidth	
Transmit Power	
Coding Rate	
Time Division Duplexing	

Mobile embedded environments may place high weighting on the minimize power consumption objective. Several factors in a wireless radio can contribute to the consumption of power, including bandwidth, modulation type, coding rate, time division duplexing, and the most obvious being transmit power. In general setting these parameters high will allow the transmission to become more error free and provide higher throughput, however more power is consumed. The transmit power and bandwidth parameters are the obvious choices for parameters that can affect this objective. Equation (5.15) shows the partial fitness function used for power consumption. Increasing transmit power and increasing bandwidth will increase the fitness score. These parameters are normalized to provide a valid fitness score range:

$$f_1 = \frac{(P_{max} + B_{max}) - (P + B)}{P_{max} + B_{max}} \tag{5.15}$$

Also affecting the power consumption are parameters such as modulation and increased symbol rate add to computational complexity and overhead to the transmission increasing processing and thus increasing power consumption. This power consumption can vary widely over different system specifications. The goal of this work is to keep the relationships general and not derive an equation for a specific system.

Increasing the modulation index increases the complexity of the system, in turn re-

quired more processing. Specifically, for M-QAM modulation, higher order M-QAM and M-PSK increases complexity in a straightforward linear fashion as it does with spectral efficiency also. One way we recognize that the complexity increases linearly with the number of bits per symbol because it has been shown that for QAM modulations, it requires an extra 6 dB per bit increase in SNR to achieve the same performance as the lower modulation index for AWGN performance.

Increasing the symbol rate also creates a linear increase in sampling rate, which also increases the processing required in a linear way. Equations (5.16) and (5.17) show the general equations that are used to represent the power consumption from increased computational complexity. Again, normalization is used to provide valid fitness scores.

$$f_2 = \frac{\log_2(m_{max}) - \log_2(m)}{\log_2(m_{max})} \tag{5.16}$$

$$f_3 = \frac{R_{s_{max}} - R_s}{R_{s_{max}}} \tag{5.17}$$

Equation (5.18) shows the combined previous equations into the linear objective function.

$$\begin{aligned} f_{min\_power} &= 1 - \left[\alpha * \frac{(P_{max} + B_{max}) - (P + B)}{P_{max} + B_{max}} \right. \\ &+ \left. \beta * \frac{\log_2(m_{max}) - \log_2(m)}{\log_2(m_{max})} + \lambda * \frac{R_{s_{max}} - R_s}{R_{s_{max}}} \right] \end{aligned} \tag{5.18}$$

where α,β, and λ represent weighting factors on the different contributions to the objective function. Each of the sections typically will not contribute the same amount of power consumption. For example, the amount of power consumed by increasing the symbol rate by 100% will most likely not be as large as the power consumed by

increasing the transmit power by 100%. To account for these different scales, we use weighting factors on each section that can be tweaked based upon the specific implementations of modules and hardware within the communication device. The selection of these parameters for our simulation is presented in the cognitive engine simulation Chapter 6.

5.4.1.4 Minimize Spectral Interference

Table 5.5 Minimize Spectral Interference Cognitive Radio Parameters

Transmission Parameters	**Environmental Parameters**
Bandwidth	Frequency Bands
Transmit Power	
Time Division Duplexing	

Minimizing interference is an important goal in shared frequency bands. For example, this goal may be given a high weighting by a secondary spectrum user operating in a primary users band. In this case the primary user has priority in a specific band of frequency, however secondary users are allowed to transmit in the band given that they don't cause interference to the primary users. Transmission parameters such as transmit power, bandwidth and time division duplexing are used to determine the approximate amount of spectral interference that is being caused by transmission. Interference is caused by overlapping transmissions with other users. Ideally, to calculate the total interference you would integrate over the spectral bandwidth that your transmitting on, and find the total power of overlapping transmissions. In this work, we assume uniform power transmission over the transmission bandwidth allowing us to derive the interference equation given by Equation (5.19).

$$f_{interference} = P * B \tag{5.19}$$

More potential exists for interference as the bandwidth increases. In addition, more power interference is potential as the transmit power of the transmitter increases. This can cause increased spectral leakage and also simply more raw power interfering with another communications system. Equation (5.20) shows the normalized spectral interference relationship with the addition of the TDD parameter.

$$f_{min\_interference} = 1 - \frac{(P * B * TDD) - (P_{min} * B_{min} * 1)}{P_{max} * B_{max} * 100} \tag{5.20}$$

5.4.1.5 Maximize Spectral Efficiency

Table 5.6 Maximize Spectral Efficiency Cognitive Radio Parameters

Transmission Parameters	Environmental Parameters
Bandwidth	
Modulation Index	
Symbol Rate	

Maximizing spectral efficiency refers to maxmizing the amount of information that can be transmitted over a given bandwidth. It is a measure of how efficient a given band of frequency is utilized by the physical layer. This objective relates directly with the bandwidth and the amount of information being transmitted. The symbol rate and modulation idex can be used to determine the total amount of information beign transmitted. In order to maximize the spectral efficiency, the system would need to high amounts of information across a little amount of bandwidth. Increasing the modulation index is the primary way of doing this, while keeping the bandwidth constant. Equation (5.21) shows the normalized relationship between these parameters.

$$f_{max\_spectralefficiency} = \frac{\frac{m * R_s}{B}}{\frac{m_{max} * R_{s_{max}}}{B_{min}}} = \frac{m * R_s * B_{min}}{B * m_{max} * R_{s_{max}}} \tag{5.21}$$

5.4.1.6 Multi-carrier Objective Functions

For a multi-carrier system with N independent subcarriers, the objective functions are defined as:

$$f_{min\_ber} = 1 - \frac{\log_{10}(0.5)}{\log_{10}(\overline{P_{be}})} \tag{5.22}$$

where $\overline{P_{be}}$ is the average BER over N independant subcarriers.

$$f_{max\_tp} = \frac{\sum_{i=1}^{N} (\frac{L_i}{L_i + O + H} * (1 - P_{ber_i})^{(L_i+O)} * R_{c_i} * TDD_i)}{N} \tag{5.23}$$

For the maximum throughput multi-carrier fitness function in Equation (5.23), each carrier's fitness score is summed together and then divided over the total number of carriers, N, to get the average fitness score over all subcarriers. This forces the system to improve the overall system fitness. The same follows for the minimize power fitness function in Equation (5.25), the minimize interference fitness function in Equation (5.26), and the maximize spectral efficiency fitness function in Equation (5.27).

$$f_{min\_power} = \left[1 - \alpha * \frac{\sum_{i=1}^{N} (P_{max} + B_{max}) - (P_i + B_i)}{N * P_{max} + B_{max}} \right. \tag{5.24}$$

$$\left. + \beta * \frac{\sum_{i=1}^{N} \log_2(m_{max}) - \log_2(m_i)}{N * \log_2(m_{max})} + \lambda * \frac{\sum_{i=1}^{N} R_{s_{max}} - R_{s_i}}{N * R_{s_{max}}} \right] \tag{5.25}$$

$$f_{min\_interference} = 1 - \frac{\sum_{i=1}^{N} ((P_i + B_i + TDD_i) - (P_{min} + B_{min} + 1))}{N * (P_{max} + B_{max} + 100)} \tag{5.26}$$

$$f_{max\_spectralefficiency} = \frac{\sum_{i=1}^{N} \frac{m_i * R_{s_i} * B_{min}}{B_i * m_{max} * R_{s_{max}}}}{N} \tag{5.27}$$

where P_i is the transmit power on subcarrier i, N is the number of carriers, and P_{max} is the maximum possible transmit power for a single subcarrier. Similarly M_i is the modulation index used on subcarrier i and M_{max} is the maximum modulation index available, while $\overline{P_{be}}$ is the averaged bit error rate over all channels.

5.4.2 Multiple Objective Goals

The weighted sum approach allows us to combine the single objective functions into one aggregate multiple objective function. Equation (5.2) shows that each objective is multiplied by a weight w_i and summed together to give a single scalar value for approximating the value of a parameter set. For the single objective equations, we form the multiple objective function for multiple carriers given in Equation (5.28). Note that the single carrier version can be easily derived by setting N = 1.

Multi-carrier:

$$\begin{aligned} f_{multicarrier} &= w_1 * (f_{min\_ber}) + w_2 * (f_{max\_tp}) + w_3 * (f_{min\_power}) \\ &+ w_4 * \left(f_{min\_interference}\right) + w_5 * \left(f_{max\_spectralefficiency}\right) \end{aligned}$$

The weighting values, w_1, w_2, w_3, w_4, and w_5 determine the search direction for the evolutionary algorithm and must conform to the constraints given in Equation (5.4). To help aid in the creation of example simulations we have defined four example weight vectors representing common scenarios a cognitive may be placed in. Each weight vector shown in Table 5.7 emphasizes different objectives causing an evolutionary algorithm using this fitness function to evolve toward solutions pertaining to the specific

objective.

Table 5.7 Example Weighting Scenarios

Scenario	**Weight Vector [w_1,w_2,w_3,w_4,w_5]**
Low Power Mode (minimize power)	[0.10, 0.20, 0.45, 0.15, 0.10]
Emergency Mode (minimize BER)	[0.50, 0.10, 0.10, 0.10, 0.20]
Dynamic Spectrum Access Mode (minimize interference)	[0.10, 0.20, 0.10, 0.50, 0.10]
Multimedia Mode (maximize throughput)	[0.15, 0.50, 0.10, 0.15, 0.10]

Using the scenario weight vectors and a genetic algorithm engine, we have generated genetic algorithm convergence results, along with the statistics representing the average final decision output by the GA. These results are based on the fitness functions and are presented in Chapter 6. Along with these results are the results of the decisions made by a rule-based system implemented using the fitness function to generate the rule base. These two systems are compared and the trade-offs for each system are analyzed to determine the system that is most feasible in a given situation. The block diagram in Figure 5.3 shows how the fitness function is used in the genetic algorithm, while Figure 5.4 shows how the fitness function fits into the implementation of the rule base.

5.5 Summary

This chapter presented the fitness equations for a subset of the parameters and objectives presented in Chapter 3. The general multiple objective optimization problem was formulated and we discussed how the problem fits into the cognitive radio wireless domain. We use weighted sum approach for the formulation of the global fitness functions that aggregated the single objective fitness functions into a weighted sum. This approach has the advantage of providing a single fitness function that outputs a scalar value. Section 5.3 introduced the single objective fitness functions that were developed. These functions are the product of a small subset of parameters and objectives. They

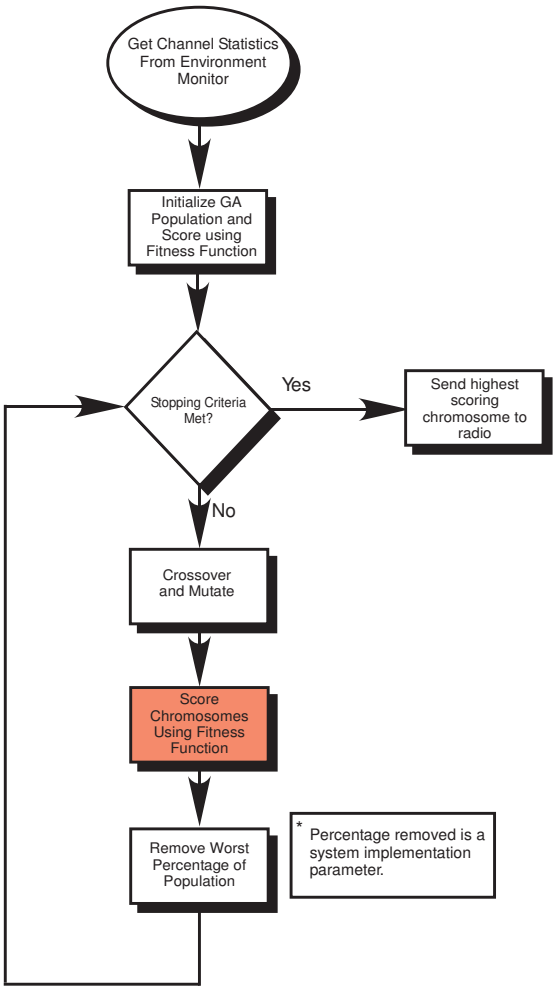

Figure 5.3 Genetic Algorithm Flow

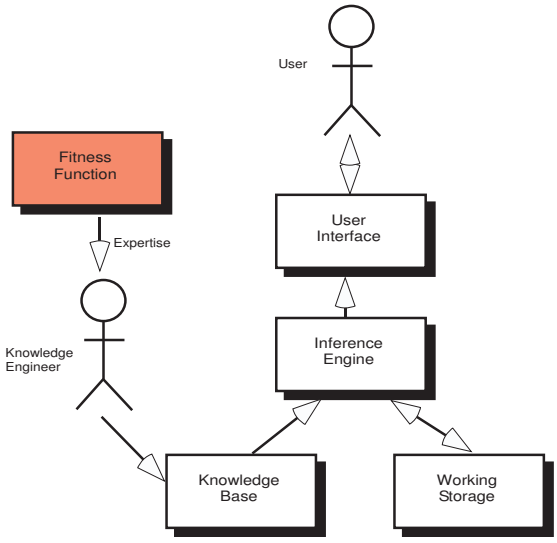

Figure 5.4 Expert System Fitness Function Representation

demonstrate a sample relationship between the parameters and objectives and how the fitness function is formulated. We then showed that by developing multi-carrier fitness functions, single carrier systems could be easily acquired by setting the number of carriers to one. The weighting vector used to direct the genetic algorithm search direction was introduced along with several example weighting scenarios that represent practical cognitive radio situations.

Chapter 6

Cognitive Engine Simulation

6.1 Introduction

This chapter presents the results gathered from the simulations of the cognitive radio engines defined in previous chapters. We being by reviewing the parameter list as defined before, and defining the parameter value ranges that we will be using in the simulations. Most of the parameter values used were choosen to be similiar to systems that would be implemented using the Kansas University Agile Radio (KUAR) hardware platform [59].

After the parameter values are introducted, the cognitive engine implementations are described in detail. Most importantly we describe how the fitness functions derived earlier are implemented within each engine. The following section gives an analysis on the advantages and disadvantages of each engine. We provide a performance comparison between the two methods that emphasize the advantages each engine has in different hardware situations.

Finally, parameter sensitivity results are provided. These results show how much impact certain parameters have on the optimality of the decision. This is an important factor when developing a cognitive engine. Selecting parameters for a cognitive engine

that have no impact on the decision is a waste of resources and can actually be harmful to the decision making process by adding uneccesary complexity. In addition, knowing which parameters have the largest impact on the decision is also important. If these parameters are known, more emphasis should be placed on the security and accuracy of these knobs.

6.2 Cognitive Parameters Representation

Section 3.2 provides a list of transmission parameters compiled to represent common transmission parameters for cognitive radios. The goal in defining these parameters was to select parameters that will be used by a large number of radios. As this parameter list increases, so does the number of control dimensions of the radio.

Table 6.1 displays a list of seven transmission parameters that we use as inputs to the fitness function, along with the ranges selected for each parameter. The trade-off analysis between these preliminary parameters and objectives presented in the next section will provide important numerical relationships that will be used to develop the preliminary fitness functions. Also shown in Table 6.1 are the symbols for each parameter which are used in the fitness equations to represent the value.

Table 6.1 Transmission Parameter Values

Parameter Name	Symbol	**Min. Value**	**Max. Value**	**Step Size**
Transmit Power	P	-8 dBm	24 dBm	1 dBm
Modulation Type	M	N/A	N/A	N/A
Modulation Index	m	2	256	m^2
Bandwidth	B	2 MHz	32 MHz	1 MHz
Channel Coding Rate	R_c	N/A	N/A	N/A
Frame Length	L	94 bytes	1504 bytes	10 bytes
Time Division Duplexing	$T\%$	25%	100%	25%
Symbol Rate	R_s	125 Ksps	1 Msps	125 Ksps

When performing the trade-off analysis, it is important to know the possible ranges

of the parameters used. The ranges allow us to put constraints on the relationships. Without constraints, the trade-off analysis would be impossible due to the infinite parameter space. The ranges chosen were based upon common wireless system specifications present in today's world, with a focus on the values being used for the KU Agile Radio (KUAR) experimental platform [59]. The transmit power, P, ranges from -8 dBm to 24 dBm. This maximum power was selected because it is approximately the maximum specified transmit power in the middle UNII band, given a 1 MHz bandwidth.

The maximum and minimum ranges for the modulation types cannot be specified because they are not numerical values. We have chosen to use quadrature amplitude modulation (QAM), phase shift keying (PSK), and frequency shift keying (FSK) signal constellations to represent the modulation parameter values. For each signal constellation, the modulation index or the number of bits per symbol, varies from $k = 1$ to $k = 8$, giving us a discrete range of 2 to 256 bits per symbol for a specific signal constellation.

The selection of the bandwidth, B, range was selected based more on convenience. The maximum value of 30 MHz is the same bandwidth used by the KUAR project. Similar to modulation type, the channel coding rate is not a continuous numerical value, but instead represents the ratio of redundant code bits to the total number of bits in a block of data. We assume only the only possible coding types are block coding and turbo coding. The link layer parameter, frame length, L, is defined as being variable from 94 bytes to 1504 bytes, which is just above the maximum transmit unit available when using ethernet. We want to explore the trade-off relationships between this parameter and all others, along with the impact on both the system throughput and BER objectives.

The third column in Table 6.2 indicates one possible step size which allows us to calculate the size of the parameter space. Using the values in Table 6.2, the total parameter space has approximately 75 million combinations. The step sizes are typically constrained by the resolution of the hardware devices. The values in this disseration

were chosen to for convenience of implementation and may not represent typical hardware resolutions. We explore how changing these values affect the operation of the GA and the rule-based system.

An important factor in this system is the parameter space of the possible environmental parameters. This parameter space represents the total number of environments that could possible be seen by the radio. Each of these environments will have an optimal transmission parameter setting, and must be represented by a single rule in an expert system. To determine the size of the parameter space, we must first determine the number of possible values for each parameter. Practically, the parameters have continuous ranges and must have a step size defined for each so as to determine the number of possible values per parameter. The third column in Table 6.2 indicates one possible step size to determine the number of values.

The SNR range was determined based upon typical values that would be seen when using a radio such as the KUAR in the given frequency range. The SNR parameter used in this research is the SNR value at the receiving radio. This information is sent back to the transmitting radio over a separate control channel. The control channel is assumed to be perfect and is not in the scope of this research. The SNR can be detected by measuring both the incoming power of the signal, which contains both the signal power plus the noise power. Next, the noise power of the channel must be sensed by measuring the channel when no signal is present. From these two measurements the SNR can be determined. This can be done by allowing the receiving radio to send out a beacon to the transmitter on a regular interval indicating that it is going to measure the noise power. The transmitter will not transmit for a specified period of time in order to allow the receiving radio to detect the noise power. More information on the measuring the SNR of a channel can be examined in [60].

As mentioned previously, we use the SNR at the receiver to determine the fitness

score. In order for the system to adapt properly to dynamic channel attenuation, we must use both the received SNR and the current transmit power in order to determine the channel attenuation and path loss. Our system detects the SNR, determines the channel attenuation and adapts the current power to the appropriate value.

The power consumption numbers are again drawn from the approximate values from version 3 of the KUAR. The spatial knowledge is represented as GPS coordinates. This information can be easily determined through the use of a GPS receiver built into the radio.

Table 6.2 Environmentally Sensed Parameter List

Parameter Name	Symbol	**Min. Value**	**Max. Value**	**Step Size**
Noise Power	N	-114 dBm	-104 dBm	1 dBm
Path Loss	PL	85 dBm	95 dBm	1 dBm
Battery Life	BL	0 %	100%	1%
Power Consumption	PC	10 W %	46 W	1 W
Spatial Knowledge	$SK[]$	N/A	N/A	N/A
Spectrum Information	$S[]$	N/A	N/A	N/A

For the algorithmic computation we will only be using the noise power and the path loss components of the environmental variables. The others are the trigger variables that allow the cognitive radio to trigger changes in the primary objective of communication. The following sections present the results of the simulations using the parameters in the previous tables. After the presentation of the implementations and results, we provide a analysis of the comparison between the two very different cognitive engine implementations. We will highlight on the performance tradeoffs each have, and emphasis the advantages each provides in different operating environments. After the comparisons, we present parameter sensitivity results that will shed light on the realistic impact each parameter has on a cognitive radio system.

6.3 Genetic Algorithm

6.3.1 Genetic Algorithm Implementation

The genetic algorithm used in this work was completely written in C, and compiled and run on linux systems. The engine was built upon the Simple Genetic Algorithm (SGA) code [61] written by researchers at The University of Alabama. As described before, genetic algorithms have several system parameters that must be set specifically for each system. In our genetic algorithm system we choose to use the de facto standard for most genetic algorithms, the DeJong settings [50]. DeJong has shown that this combination of parameters work better than many other parameter combinations for function optimization. However, we make one modification to the DeJong settings due to our search space being extraordinarly large. We increase the population size of each generation from 50 to 200. This change was choosen based upon a crude population scaling law in [62]. This change allows the system to operate on more combinations in parallel. Without this change the small population size causes the GA to work on an extremely small subset of the entire population. This small size prohibits the GA from being able to have spontaneous evolutions because the small chance of randomly finding an optimal chromosome. With a larger population, the processing need is greater however the GA is able to more fully explore the search space, which is greatly needed with large search spaces such as the one in our problem. Table 6.3 shows the Dejong settings alongside the settings we used for our genetic algorithm implementation.

To build the chromosomes, we first determined the number of bits that were needed to represent each parameter. This was found by using the total number of possible values for each parameter and taking $\lceil log2 \rceil$ of this value and modifying the parameter values as needed to conform to this distribution. For example, for the frame size pa-

Table 6.3 DeJong Genetic Algorithm Settings

Parameter	DeJong	**Ours**
Population Size	20	50
Number of Generations	1000	1000
Crossover Type	Two Point	Two Point
Crossover Rate	0.60	0.60
Mutation Type	Bit Flip	Bit Flip
Mutation Rate	0.001	0.001

rameter we initially wanted to vary the length of the frame size from 100 bytes to 1500 bytes in 100 byte increments. This gives us 15 total values for this single parameter, however we need 16 uniformly distributed values. To achieve this, we simply divide 1500 bytes by 16 to get the step value needed to have 16 values. In the frame length case, 93.75 bytes is the step value, however we round this to 94 bytes and make the maximum framesize 1504 instead of 1500 to avoid any minimal biasing effects.

In total, the length of each chromosome for a single channel system consists of 31 bits. As the number of channels grows in the system, the size of the chromosomes also grow. For example, in a 2 channel system, we have 2 independent channels each with their own set of parameters. So the total length of a 2 channel chromosome will be 62 bits, or two times the 31 bits of a single channel system. Figure 6.1 provides a visual representation of this increase in chromosome length. Increasing the size of the chromosome adds complexity to the GA system. As the GA results section will show, it takes longer for systems to converge on an optimal result then they have larger number of channels. This is because the longer chromosome requires a larger amount of processing time in order to determine the fitness. We have already shown that the amount of time required grows linearly with the increase in number of channels. This result is intuitive due to the linear increase in processing needed with the larger chromosome.

The main process begins with the GA engine populating 200 chromosomes with random bits. These chromosomes are the initial population of the system. However,

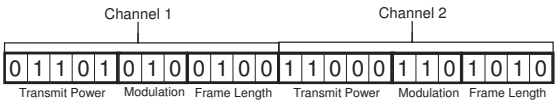

Figure 6.1 Chromosome Length Growth

in the adapted version of the GA system, as we described in Section 4.2.1, we don't initalize all the chromosomes randomly. Based upon the change in the environment since the previous adaptation cycle, we initialize some of the chromosomes with the chromosomes that we ended with in the previous cycle. This biases the initial generation to the solution of the previous cycle. Under the assumption that the environment hasn't changed much, this biasing can save the GA system processing time and allow it to converge at a much faster rate. The results for the adaptive GA system are shown in Section 6.3.2.2

Once this initial population is created, the fitness function is perform on each individual chromosome and the fitness value for each is stored in memory. The fitness value not only represents the optimality of the parameters, but also is the means by which the GA engine determines whether a specific parameter set is selected to move onto the next stage of evolution. Our system uses the stochastic remainder method to select which chromosomes are used for the evolution process. In the stochastic remainder method, the ratio between a single chromosomes fitness and the average fitness of the population is used to determine the number of copies of the chromosome which move on to the evolution process. For example, a chromosome with a fitness to fitness average ratio of 1.50 would be guaranteed one copy is moved onto evolution, and have .50 probablity that a second copy would also be moved on.

Once the intermediate population of chromosomes have been selected, they can now be recombined to create the new population. Random chromosomes are selected to be combined using the two-point crossover process shown in Figure 2.3. The probability

two randomly selected pairs go through the crossover process is a specific GA system variable, in which we use 0.60 as shown in Table 6.3. Once the crossovers are complete, they will go through the mutation process. The chance for a bit mutation is typically very small, but it allows the GA to jump outside of the local search space, thus giving it the ability for spontaneous evolution. In our system we have a 0.1% chance that a bit will flip for each bit in the chromosomes. Once these operations have been applied to the new chromosomes, the fitness function assigns new values to the chromosomes and the process is repeated until we hit 1000 generations. The results from our standard GA simulations are shown in Section 6.3.2.1.

6.3.2 Genetic Algorithm Results

In this section we present the results of the GA simulations. Several simulations were performed that cover a wide range of environment. Our first goal is to explore how the GA converges with the large number of parameters used in the complex fitness function. We look at convergence results from each of the four performance objective scenarios that were defined in Table 5.7.

The next result we analyze are the actual parameter settings that the GA produces. These results will vary based on the performance objectives. We look at how each the performance objective weighting guide the GA to different parameter settings. In addition to the convergence and parameter settings, we show the amount of time needed to produce a result. An important aspect of this work is the amount of processing time needed by the GA in order to produce a valid output. Increasing the complexity of the system increases the time need to produce a result. We explore the effect that the number of channels has on the time needed to converge to a result. We also show that this time is reduced when employing the adaptive techniques described in Section 6.3.2.2.

6.3.2.1 Non-Adaptive Genetic Algorithm Results

We begin with the minimize power performance objective results. Figure 6.2 shows a standard fitness convergence graph obtained from the GA system. This figure shows the best results from varying channels in the system. It can be seen that a system with a single channel converges much faster then the system with 16 channels. This is due to the processing time needed to calculate the fitness over a 16 channel system. These results, as with all of the following results, are averaged over 100 runs with each run using random environment variables. To highlight the effect of the increasing number of channels in the system, Table 6.4 shows the optimal generation where the highest fitness was found for each system. Again, for a single channel system, the system is able to find the best value much earlier than the system with 16 channels.

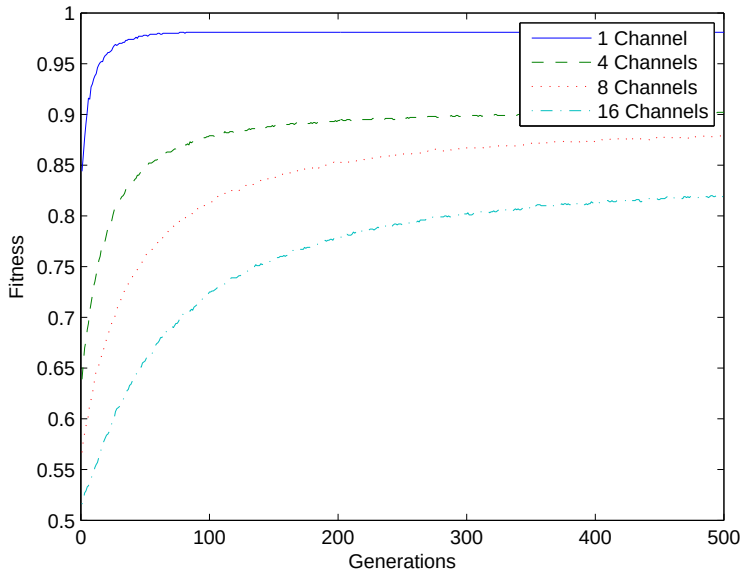

Figure 6.2 Fitness convergence curves for the minimize power consumption performance objective for systems with varying number of channels.

Figure 6.3 shows the convergence results for the Emergency scenario. In this scenario we have a large emphasis on minimizing the BER of the system in order to provide

Table 6.4 Power: Number of Channels vs. Optimal Generation

Number of Channels	Optimal Generation	Optimal Fitness	Time per Generations (ms)
1	77	0.981	1.0
4	789	0.904	3.9
8	845	0.886	7.7
16	892	0.831	17.6

a more error free communication geared toward situations where clear communications is essential. This scenario exhibits the same characteristics as the previous convergence graph. Increasing the number of channels requires a more processing, resulting in slower convergence rates for a higher number of channels. To get a feel on how the number of channels affects the fitness convergence, we can see from Figure 6.3 that a channel with 16 channels causes the fitness to cap the upper limit at approximately 20% lower then the system with a single channel.

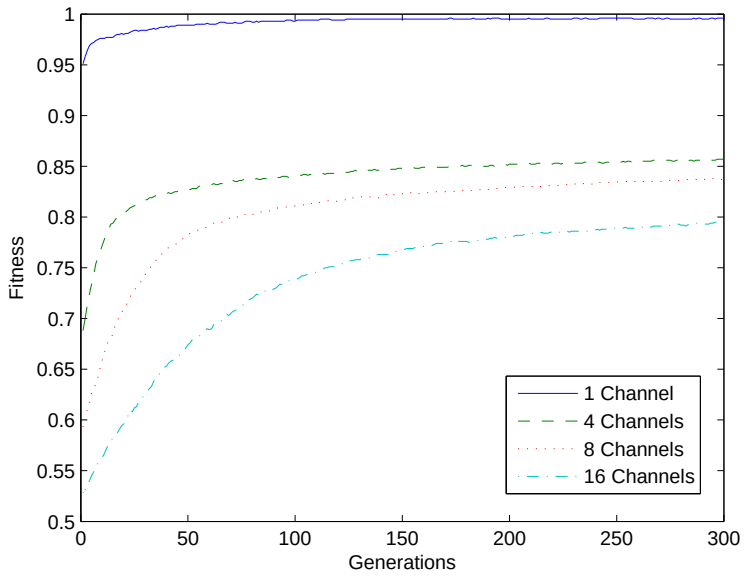

Figure 6.3 Fitness convergence curves for the emergency scenario performance objective for systems with varying number of channels.

The following Figure 6.4 and Figure 6.5 also following this convention. For the multimedia scenario, the 16 channel system decreases the achievable fitness by 26%

Table 6.5 Emergency: Number of Channels vs. Optimal Generation

Number of Channels	Optimal Generation	Optimal Fitness	Time per Generations (ms)
1	244	0.993	1.1
4	848	0.863	4.1
8	927	0.855	8.5
16	991	0.796	16.7

as compared to the single channel system. The DSA scenario has a cap of only 10% lower. This is because the DSA scenario is technically a simpler trade off with fewer parameters that are needed to converge.

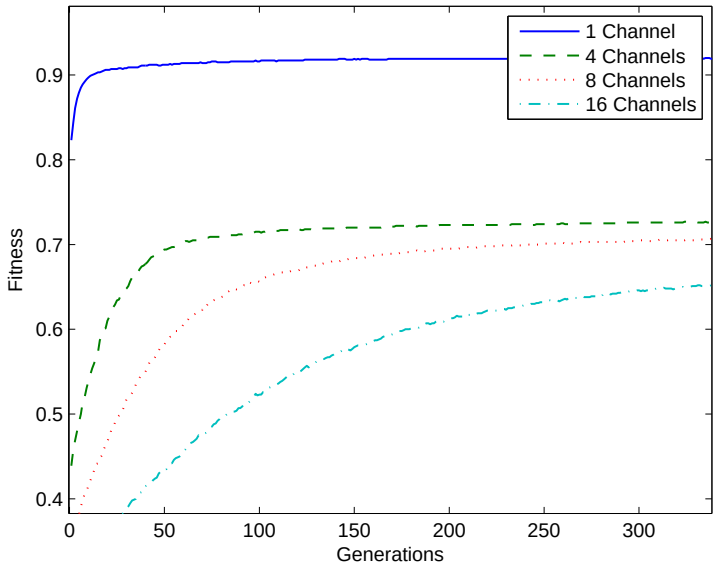

Figure 6.4 Fitness convergence curves for the multimedia scenario performance objective for systems with varying number of channels.

An interesting attribute to point how is how the actual performance objective weights are affecting the convergence results. Typically systems with higher number of channels have a harder time converging. This is a consistant problem throughout all the scenarios we have explored. However, some have a harder time than others. As briefly discussed earlier, this is because the performance objectives vaary in complexity. This complexity is defined by both the actual complexity of the algorithm, but also the num-

Table 6.6 Multimedia: Number of Channels vs. Optimal Generation

Number of Channels	Optimal Generation	Optimal Fitness	Time per Generations (ms)
1	252	0.920	1.1
4	644	0.717	4.3
8	971	0.717	8.5
16	984	0.681	16.7

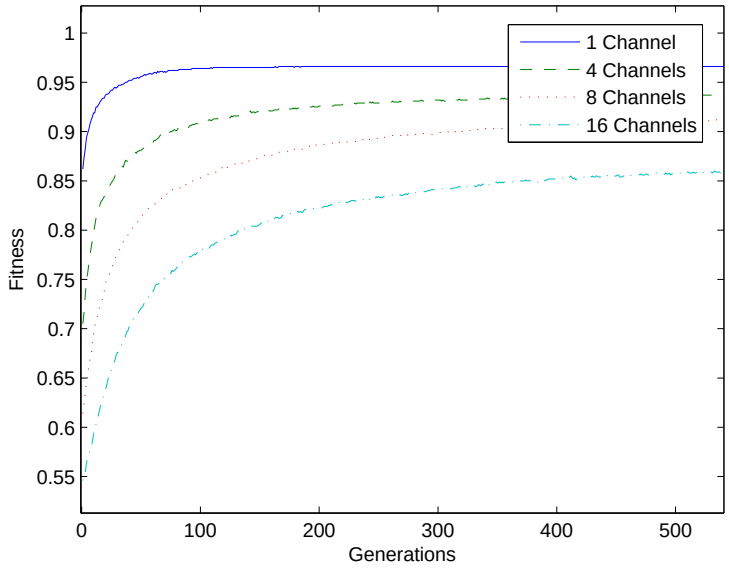

Figure 6.5 Fitness convergence curves for the dynamic spectrum access scenario performance objective for systems with varying number of channels.

Table 6.7 DSA: Number of Channels vs. Optimal Generation

Number of Channels	Optimal Generation	Optimal Fitness	Time per Generations (ms)
1	163	0.966	1.1
4	897	0.940	4.2
8	953	0.918	9.3
16	932	0.870	16.3

ber of parameters and tradeoffs the function creates. For example, the emergency scenario deals with the BER of the system. Several parameters affect the BER that are present in other performance objectives. These tradeoffs make it less clear to the system which parameters to use, required more processing and exploration in order to determine the optimal fitness. Notice that even in these cases, the fitness is always monotonically increasing, albiet slowly. In contrast scenarios such as the DSA scenario, it is clear how to set the parameters because there are not as many tradeoffs between the parameters that affect the spectral interference objective. This results in the more complex systems with more channels not having such a decrease in the fitness cap as other scenarios.

The decreasing performance of GAs with more complex systems is the primary drawback of using such a system. The following section suggests improvements that cause the GA to increase the time it takes to converge to the optimal fitness, and also improves the system performance for systems with higher number of channels. These improvements are based upon previous information and the amount of channel deviation that has happened since the previous GA cycle.

Another important characteristic of the systme is the time per generation. In a practical system, in order to optimize performance we would want the cognitive system to stop after it reaches the generation that gives the highest fitness possible, within a ceratin threshold. For example, in the minimize power scenario as shown in Table 6.4, a single channel system would stop after 77 generations. The average time per generation is 1.0 ms requiring a total of 77 ms for the complete computation. For higher number of channels, the system has a much harder time converging requiring a larger number of generations and an even larger time per generation. We see in the DSA scenario in Table 6.7 that for a 16 channel system the time per generation is 16.7 ms and the average optimal generation is 984. A total of 16.4 seconds is required to come to the

optimal solution in this scenario.

To put this computation time into perspective we look at the time in a 3G system such as High-Speed Downlink Packet Access (HSDPA) in UMTS [63]. In this system each user device transmits an indication of the downlink signal quality, as often as 500 times a second. The base station then decides which users will be sent data on the next 2 ms frame and how much should be sent. So overall, a HSDPA system provides simple link adaptation results in 2 ms. In respect to the GA system, a traditional link adaptation system such as the 3G HSDPA system is significantly faster at updating the parameters and adapting. However, link adaptation is a simple approach to adjusting parameters similiar to the RBS approach. Typically only the power and modulation are adjusted using a standard table of parameter values corresponding to specific values of SNR. The GA system introduces several more parameters that provide more dimensions of control over the system. Unfortunately this larger dimension of control comes at a computational resource price. However, in the next section we provide results on adaptive GA techniques that improve the convergence time of the GA, causing it to require less amount of generations to get to an optimal solution.

6.3.2.2 Adaptive Genetic Algorithm Results

For simulation purposes, we considered following two cases using the preliminary fitness functions:

- Emergency Mode (minimize BER)
- Low Power Mode (minimize power)

Figure 6.6 shows the effect on the convergence rate of varying population seeding percentages over a system with a 10% EVF operating in emergency mode. In our simulation, the EVF represents the percentage change in the noise power and channel

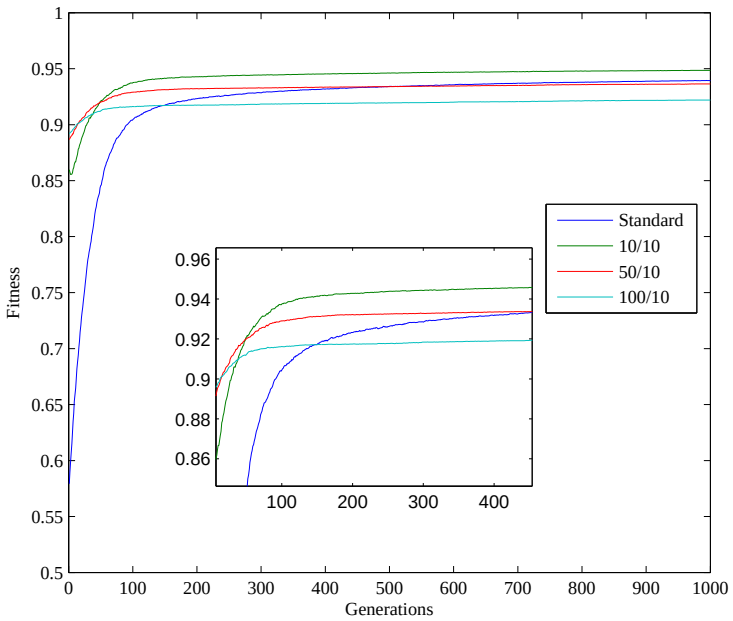

Figure 6.6 Fitness convergence in emergency mode with 10% EVF, where X/Y represents the ratio of the seeding percentage and EVF percentage.

attenuation. The fitness convergence statistics shown represent the average fitness of the best chromosomes for each generation. The simulation results are averaged over 500 different randomly generated environments for each generation. The standard line represents the standard GA implementation that is initialized randomly. The figure shows that as the seeding percentage increases, the initial fitness of the population increases. The seeding is giving the GA algorithm a better estimate of where to begin the search initially, enabling the algorithm to start at an increased initial fitness and converge to a higher value. As a validation of these fitness scores, Figure 6.7 shows the simulatanous BER convergence with respect to the number of generations. This plot verifies that the higher fitness scores are providing lower BER.

As the population seeding increases, the algorithm uses more information from previous cognition cycles to determine a good initial population. Figure 6.7 shows that a 10% seeding value allows the algorithm to start at a higher initial fitness and converge

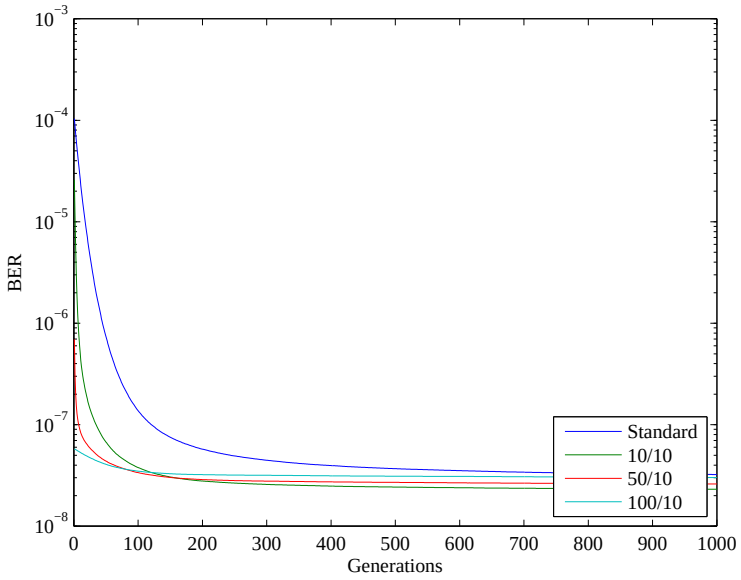

Figure 6.7 BER convergence in emergency mode with 10% EVF, where X/Y represents the ratio of the seeding percentage and EVF percentage.

to a higher value than the standard GA algorithm. In addition, the proposed population adaptation technique reaches within 1% of the standard GAs converged value in 70 generations and continues to improve past this value. This is an 480% improvement in speed over the standard GA implemenation that converges at approximately 337 generations. However, at 50% seeding the GA converges to a lower fitness value than the standard GA. This is due to the large number of similiar chromosomes being seeded initially. This lack of diversity causes the algorithm to become stuck within an area of the search space that is not optimal. This affect becomes more prominent as the environment becomes more dynamic.

As the EVF increases, the wireless environment is allowed to become more dynamic and as a result of our restriction on the variation of the environment the average noise level increases. This causes the population seeding technique to become less effective at higher values, because the information from previous cognitive cycles becomes less accurate when predicting the new location. Figure 6.8 shows the convergence statistics

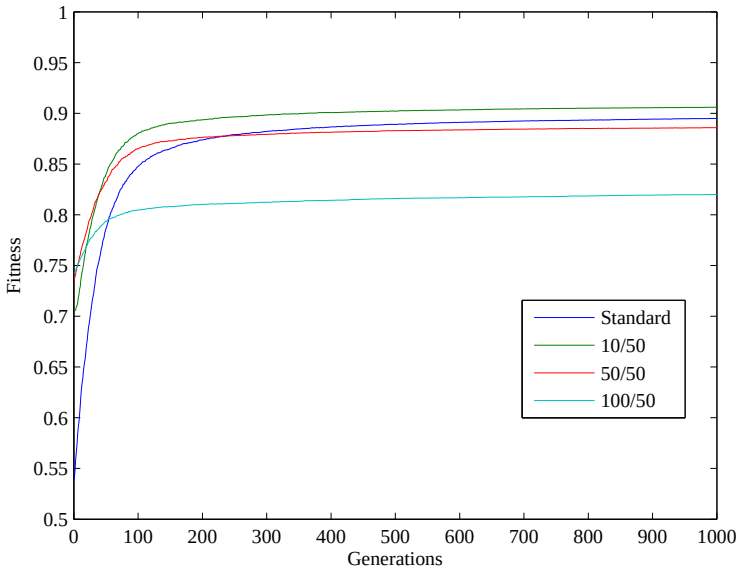

Figure 6.8 Fitness convergence with 50% EVF in emergency mode, where X/Y represents the ratio of the seeding percentage and EVF percentage.

with a 50% EVF value. The initial fitness with a 50% EVF is lower than the environment with only 10% EVF, however it still is initially higher than the standard GA. However, as the seeding is increased, the convergence rate quickly degrades more than the 10% EVF case. This effect is also shown in Figure 6.9.

The figures also show the effect of the increased average noise on the fitness scores. As the EVF increases, the increased average noise causes the average fitness scores to decrease. This is because the BER fitness function must be normalized to a worst case BER of 0.5 for all values of EVF. This causes environments with lower average noises to achieve higher fitness scores. Ideally, the BER fitness function would be normalized to the worst possible BER given the specific environment values. Practically, we can not quickly determine the specific worst case BER, so we normalize the function to a BER of 0.5. This causes the range of possible fitness scores to vary according to the environment values used. However, this effect on the range of fitness scores as seen by the different standard GA lines in Figure 6.6, Figure 6.8, and Figure 6.9, does not

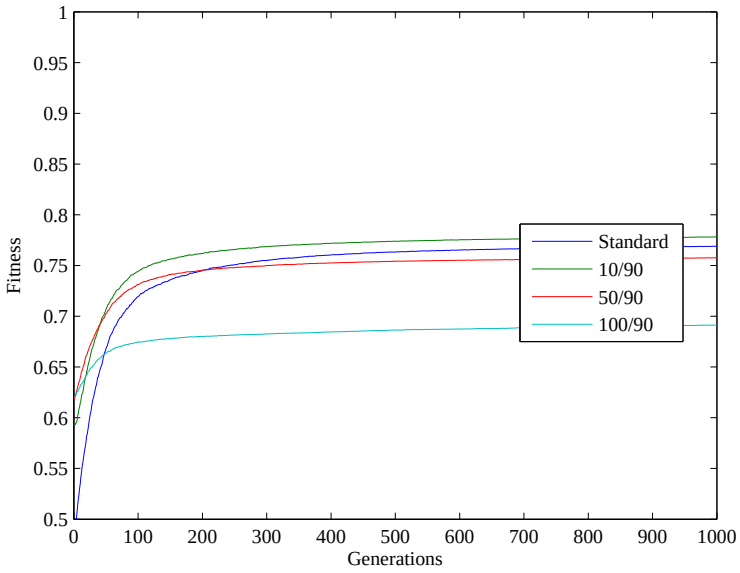

Figure 6.9 Fitness convergence with 90% EVF in emergency mode, where X/Y represents the ratio of the seeding percentage and EVF percentage.

change the fact that the GA is still determining the best possible fitness for the given environment.

Figure 6.9 shows how a highly dynamic environment is affected by population seeding. With 10% seeding the proposed technique is an improvement over the standard GA, however, there is less of an improvement in the case of 90% EVF than the lower EVF situations. In this case, the 10% seeding coverges to within 1% of the standard GAs converged value in 192 generations, whereas the standard GA in the 90% EVF case converges in 426 generations. This indicates an improvement of approximately 220% over the standard GA. We can also see from the plot that the 100% seeding case convergence is much lower relative to the standard GA than the previous plots. This is because as the environment becomes more dynamic, large amounts of seeding only cause the algorithm to become stuck further away from the optimal decision, thus causing a lower average fitness score.

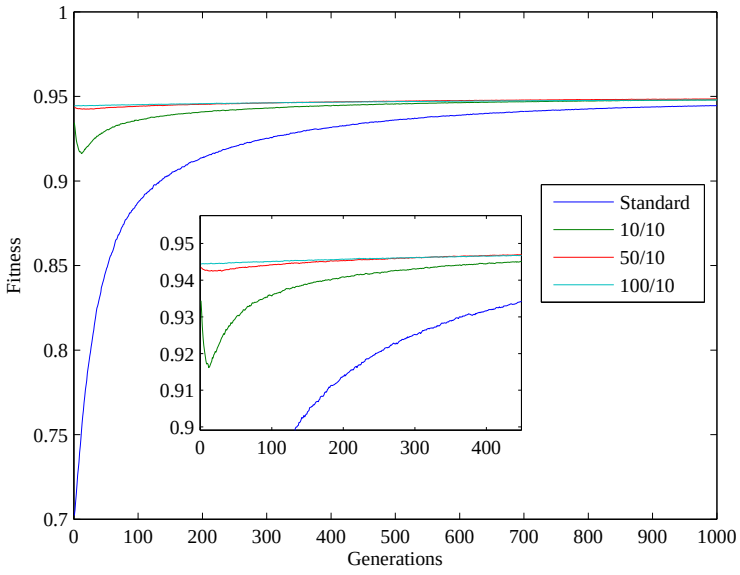

Figure 6.10 Fitness convergence with 10% EVF in Low Power mode, where X/Y represents the ratio of the seeding percentage and EVF percentage.

Figure 6.10 shows the simulation results in low power mode with 10% EVF. Low power mode is defined in a way that changes in the environment do not have such a big affect on the selection of an optimal decision as they do in emergency mode. This is because in low power mode, the performance objectives emphasis is on operating with low power consumption. In our simple case this means lower transmit power translates to higher fitness, disregarding the current environmental state. For example, if a cognitive cell phone detects low battery power, the primary performance objective would switch into low power mode. Figure 6.12 shows the results in low power mode with 50% EVF, which are similiar to the results with 10% EVF. For this mode, the radio can take advantage of higher percentage seeding to achieve significantly improvements in the convergence rate with respect to the standard GA convergence. This work has been published [64], and selected to be published in a journal associated with that conference [65]

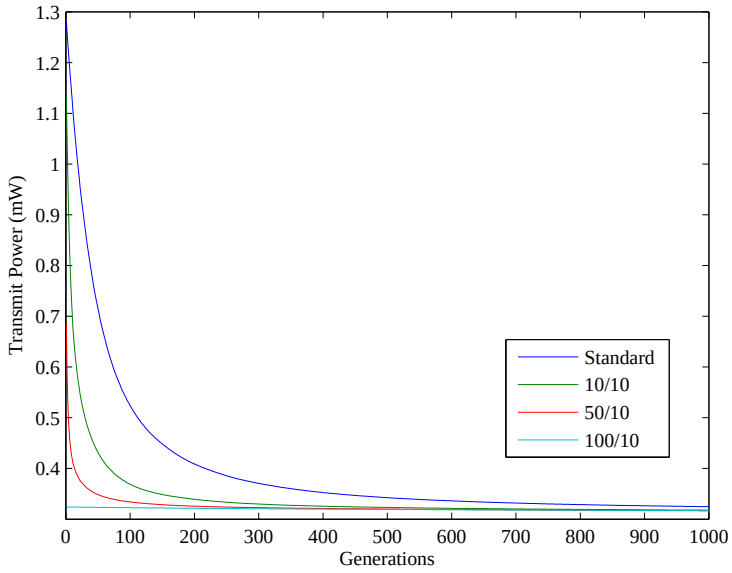

Figure 6.11 Power convergence with 10% EVF in Low Power mode, where X/Y represents the ratio of the seeding percentage and EVF percentage.

6.4 Rule-Based System

6.4.1 Rule-Based System Implementation

The implementation for the RBS consists of the MATLAB generated rules and the CLIPS expert system shell. Initially the CLIPS shell is ran and the rules are manually loaded into the system. At this point, facts are inserted into the system and the expert system is executed using the ”(run)” command. This activated the inference engine which matches the facts to the specific rules and asserts the appropiate facts.

For the RBS we are concerned with several issues. Initially, we wanted to look at the memory usage needed to hold the database of rules. This issue turned out to not be a problem due to the number of possible external environments not being large. However, we explored four different weighting scenarios, each holding 36 combinations of environments resulting in $4 * 36$ or 144 rules. If we were to explore all possible weighting

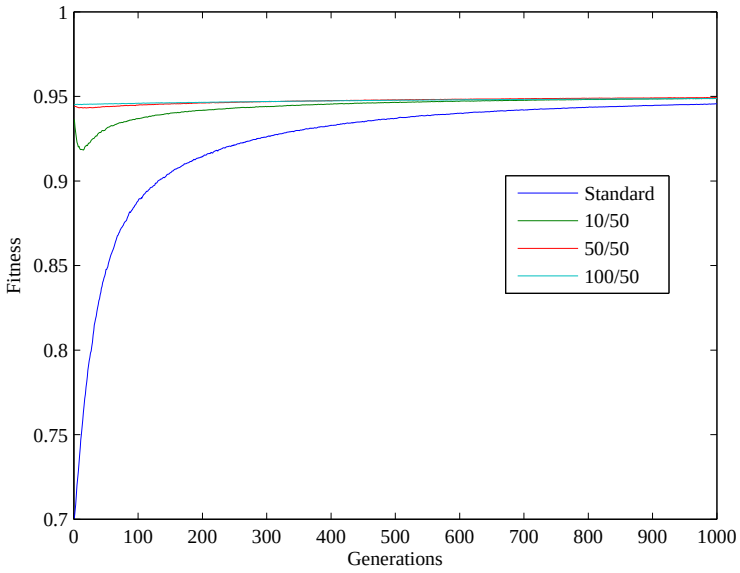

Figure 6.12 Fitness convergence with 50% EVF in Low Power mode, where X/Y represents the ratio of the seeding percentage and EVF percentage.

scenarios, we would need a total of $100^5 * 36$ rules or 360 trillion rules. This number could be dramtically cut down by combining like rules which would be very common.

The second issue explored throughout the implementation of the RBS system is the flexibility of the system as compared to the GA system. Having a hard coded set of rules does not allow the flexibility of parameter changes that the genetic algorithm has. Currently the GA system inputs a simple text XML file holding all the system parameters such as the objective weights and which parameters are to be used in the system. The RBS has no option to select which transmission parameters are to be used. This requires a completely new rule base to be compiled for any hardware change in the system. The GA doesn't require any coding changes if parameters need to be enabled or disabled. This is the results of the static nature of expert systems that must be hard coded before run-time. In addition, information about the operating environment may not be available until the system is needed. For example, gather the proper

ranges for the environmental parameters can require significant time to collect accurate measurements. In a battlefield or hostile environment, collecting measurements before the system can be used may risk lives and cost significant amounts of money. Thus needed a precompiled database of rules may not always be a realistic resource that can be created.

Contrast to those disadvantages, in a system that will not be modified the RBS may be the most efficient engine to use. If it is possible to gather information about the environment and generate the rule base offline then the light processing and memory needed for a specialized system implementation can provide for a cognitive engine that will make quick and optimal transmission parameter decisions. The following section gives the performance and parameter results gathered when using the RBS to find the solutions.

6.4.2 Rule-Based System Results

The results for the RBS are essentially the rules created through the use of the MATLAB code in Appendix A. The RBS implementation brought to light several important research questions that we analyze along with presenting the results of the engine. With a RBS, the environment parameters are matched to specific rules that were created offline and the transmission parameters associated with these rules are applied. When matching against the values of the environment parameters we must be careful not to create rules containing discrete numbers. In practical systems, the inputs to the cognitive engine will come from the sensors available to the system. We cannot assume these sensors will provide nice integer numbers. In fact, many sensors pride themselves in their sensitivity and provide very accurate measurements. With this in mind, the rules that we create must cover specific ranges of environment variable values. So instead of matching on a noise power of -112 dBm, we will match on a noise power within the

range of -111.5 dBm and -112.5 dBm. This implementation requirement results in an interesting design parameter. How does the range of the parameter in the rule affect the number of rules needed, and the optimality of the decision.

The ranges we set introduce error in the decision of the RBS. The transmission parameters that each rule specifies to use are based upon the center of the ranges in our case. Thus, the more deviation from the center value of the environment parameter value, the further away the decision will be from the actual parameters used to generate the decision. Our goal is to determine ranges for the parameters that will keep the number of rules to a minimum while keeping the error of the decision also minimum. These two goals create a decision. Keeping the number of rules to a minimum requires larger parameter value ranges, while keeping the error to a minimum requires smaller parameter value ranges.

Shown below is an example rule that would be created and entered into the CLIPS system:

```
(defrule cognitive_rule_13
  (and (noise_power ?channel_num ?noise_power&:(>= ?noise_power -115.5))
       (noise_power ?channel_num ?noise_power&:(< ?noise_power -114.5))
  (and (path_loss ?channel_num ?path_loss&:(>= ?path_loss 86.5))
       (path_loss ?channel_num ?path_loss&:(< ?path_loss 87.5))
  (scenario ?channel_num power_mode)
=>
  (assert (channel ?channel_num 14  2   2 psk 1500 1.00  125000  25)) )
```

This rule states that if the noise power is less than -114.5 dBm and greater than -115.5 dBm and the path loss is less than 87.5 dBm and greater than 86.5 dBm and we are operating in minimize power mode, set the transmission parameters to the following settings:

- Transmit Power: 14 dBm

- Bandwidth: 2 MHz
- Modulation: BPSK
- Frame Length: 1500 bytes
- Code Rate: 1
- Symbol Rate: 125 Ksps
- TDD: 25%

For this example we set the bin size to be 1 dBm and the transmission parameter settings were generated for a system with noise power of -115 dBm and path loss of 87 dBm. We load a set of rules similiar to this one into the system using the ”(load* ¡rules¿)” command. At this point the system now waits for ”facts” to be asserted. An example declaration for a fact that would match the previous rule would be:

```
(deffacts c2
        (noise_power 2 -114.5)
        (path_loss 2 87.5)
        (scenario 2 power_mode) )
```

This fact would be asserted into the system, and once the inference system is ran, using the CLIPS command ”(run)”, a new fact is instantly asserted that states:

```
(channel 2 14  2 2 psk 1500 1.00  125000  25)
```

In this example, the system sees an environment that is right on the edge of the parameters ranges made for the rule that it matched. Although this rule matches the given environment, the transmission parameters asserted were not generated for this environment, creating a less than optimal decision. In this example we have shown,

the error between the fitness of the environment for which the transmission parameters were created and the environment that was actually seen is only 2%. As we increase the bin size to 1.5 dB we see an error of 5% and as it reaches 2 dB we get an 8% error in optimality. However, with the bin size at 2 dB our rule size is cut in half. Table 6.8 shows the results of the fitness deviation resulting from increasing the bin size of the parameters for the difference scenarios.

Table 6.8 Worst Case Fitness for Various Bin Sizes and Scenarios

Bin Size	Power Scenario	Emergency Scenario	Multimedia Scenario	DSA Scenario
1.0 dB	0.9785	0.9416	0.9247	0.9414
1.5 dB	0.9475	0.9202	0.8960	0.9201
2.0 dB	0.9224	0.9007	0.8607	0.8998
2.5 dB	0.8994	0.8745	0.7778	0.8647

6.5 Performance Comparison

The initial performance comparison we look at are the fitness scores. Each systems fitness varies based upon the specific system parameters that we are usign at the time. For the RBS, with small bin sizes, we achieve a better estimate of the environment giving us a better average fitness value. As the bin size grows the approximation becomes less effective causing the average fitness score to decrease. Similiary, the number of channels in the GA system affects the overall fitness due to the increasing complexity of higher number of channels. Increasing the number of channels increases the chromosome size, and causing a larger processing requirement for each generation. A summary of these results is shown in Table 6.9.

The RBS has the advantage of providing these scores in a channel independant environment. No matter how many channels are in use in the system, if the bin size is 1 dBm and the performance objective uses the minimize power weights, the worst

Table 6.9 Cognitive Engine Results Comparison

Scenario	GA: Single Channel	GA: 16 Channel	RBS: 1.0 dBm	RBS: 2.5 dBm
Minimize Power	0.973	0.882	0.9785	0.8994
Emergency	0.992	0.793	0.9416	0.8747
DSA	0.966	0.869	0.9414	0.8647
Multimedia	0.920	0.680	0.9247	0.7778

case fitness will achieve 0.9785. This is because each rule is ran independantly for each channel. This is not the case for the GA system where the number of channels dictates the number of bits in the chromosome. The nature of the GA system requires the whole chromosome to be processed at once. Thus, systems with higher number of channels require longer amounts of processing time with respect to systems with smaller number of parameters. In addition, the 1000 generation stopping criteria places a cap on the evolution of the genetic algorithms. This results in lower fitness scores for systems with higher number of parameters.

The adaptive genetic algorithm showing much improvement over the non-adaptive engine has been shown to converge to a solution in as little as 70 generations. In addition, a GA system can be designed and deployed in almost any environment with little changes to the system. Using XML configuration files as inputs to the system, the weights and active parameters can be changed on the fly without any offline or pre-design computation. In the end, in terms of raw performance the RBS outperforms the GA due to the amount of processing that can be performed offline.

In terms of practical usage, each system has its place. The RBS thrives in non-mobile environments such as cell tower applications where the environment is not changing dramatically enough and the performance objectives are typically always unchanged. This results in fewer dynamic changes to the system and the cognition can be focuses on adapting to the low level channel environment. A GA system would benefit a very mobile environment such as a handheld battlefield application. If needed the

user could change the performance objective drastically and the system automatically changes goals and converges using the new weights. In addition, parameters can be deactivated dynamically to save resources in a limited resource environment if needed. For example, a soldier may notice that the battery life is running low and instead of allowing the cognitive engine to modify the weights and operate in low power mode, which results in a shorter communication range, the soldier can deactivate the frame length and code rate parameters and set them to static values. This change may save processing and memory usage that may result in longer life for mission critical applications.

Refering to the previous scenario, the specific parameters that the soldier needs to deactivate should be choosen wisely. If the soldier deactivates an important parameter, the communication may be severly degraded. To address this, we do a parameter sensitivity analysis for cognitive radios. This analysis will help users such as a soldier in a battlefield using a cognitive radio such as ours, to understand which parameters are sensitive to the current performance objective and which parameters add little to nothing and are not needed.

6.6 Parameter Sensitivity Analysis

An interesting result of this work is the ease of selecting and deselecting parameters that are adaptable. The GA enables an interface that allows users to select the appropiate parameters for their system. For example, only transmit power and modulation may be available and adaptable on a simple system, while others may include frame length and code rate also. With the ability to select and deselect parameters we can easily do parameter sensitivity analysis that is much needed in the area of cognitive radios. Much hype has been spoken about the ability of cognitive radios to automat-

ically adapt parameters to achieve optimal communications. This work analyzes several issues with complexity and performance issues specifically with regard to Genetic Algorithms. These complexity and performance issues are common to all cognitive engines, not just the GA's we analyze in this work. The primary question, does the adaptability that cognitive radio enables really provide a performance advantage when it comes to processing time and efficient resource usage.

One research question we use to answer the more abstract question is, how effective are the parameters that are adaptable? For example, a cognitive radio with an over abundance of parameters may require extreme amounts of unessecary resource requirements, primarily being processing. Identifying the primary parameters that contribute greatly to the wireless communication will allow developers to "weed out" unessecary parameters that do not add anything except processing time to the system. To explore this, we disable parameters in our system and watch how the fitness is affected when a parameter is not required to be adaptable. We expect to see the fitness converge to low fitness values for the important parameters such as transmit power and modulation, and the fitness converge to similiar values as presented before for parameters that do not contribute much. We also expect the sensitivity to be greatly affected by the performance objective. For example, if a majority of the weight is placed upon maximizing the spectral efficiency of the communications, then the modulation index, or the number of bits per symbol will be very sensitive and greatly affect the fitness, while frame length will have little to no affect.

6.6.1 Power Scenario

We begin our sensitivity analysis with the minimize power performance objective. The largest impact we expect to see is keep the transmit power parameter from being adapted. We set the transmit power parameter to two seperate static values of -8 dBm,

10 dBm and the 20 dBm and observe the affect it has on the fitness. The fitness convergence graph is shown in Figure 6.13 that shows the original fitness graph mapped over the fitness of the systems that are not able to adapt transmit power.

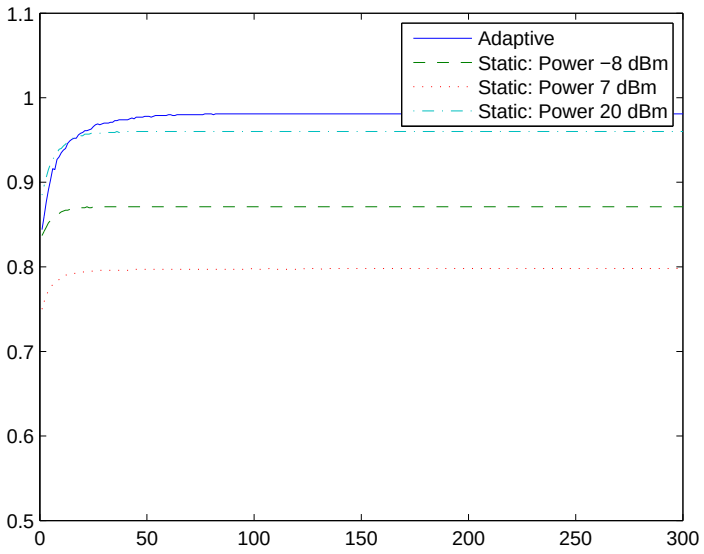

Figure 6.13 Minimize Power: Fitness convergence effect when holding the transmit power parameter static at various values versus being completely adaptable.

Figure 6.13 shows how the fitness is affected due to the static non-adaptability of the power parameter. With the power adaptable we converge at a fitness near perfect, or near 1. However, without being able to adapt the power parameter, the fitness converges at 10% lower for a static setting of 10 dBm and 8% lower at a static value at the maximum of 24 dBm. These results tell us that for this specific performance objective, the transmit power parameter has significant value on the output of the system. Another parameter that we expect to have a major affect on the fitness when being held constant is the time division duplexing parameter, TDD. The value of this parameter tells us how much time we are transmitting. A low value of TDD is ideal for low power scenarios, while higher values correspond to longer transmissions and thus higher power usage.

Figure 6.14 shows the results of holding TDD constant at 100% or ”always on”.

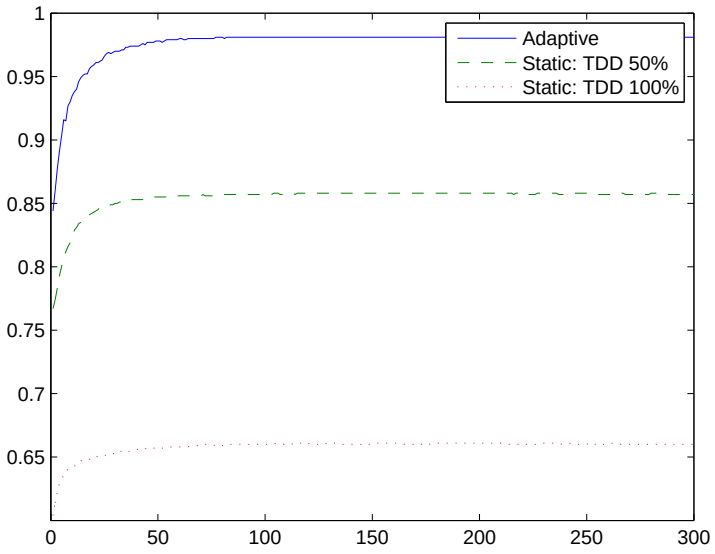

Figure 6.14 Minimize Power:Fitness convergence effect when holding the TDD parameter static at various values versus being completely adaptable.

The results of the TDD sensitivity on the minimize power scenario is very signifiant. While setting the TDD to 100% and not allowing the system to adapt it, the fitness can evolve to only 66% of the adaptable systems fitness value. While TDD is not typically a primary concern when setting transmission parameters, these results show that this parameter can significantly affect transmission, primarily because it controls the basic transmission state of the system. In reality, a communications system will typically never be transmitting 100% of the time. However these results point out the importance of allowing the system to adapt to appropiate values.

Finally we look at the result of not adapting the modulation index for the minimize power scenario. With the emphasis on minimizing the power, holding the modulation index at a more power consuming modulation will cause the fitness to have a lower upper limit. However, the modulation plays a smaller role in the minimize power fitness score than transmission power and will not affect the score as much. Figure 6.15 shows

the results of keeping the modulation index static at eight symbols per second. The system is only loosing about 5% of optimality which give this parameter a low sensitivity. Table 6.10 shows a summary of the parameters we have explored for the performance objective with an emphasis on minimizing power.

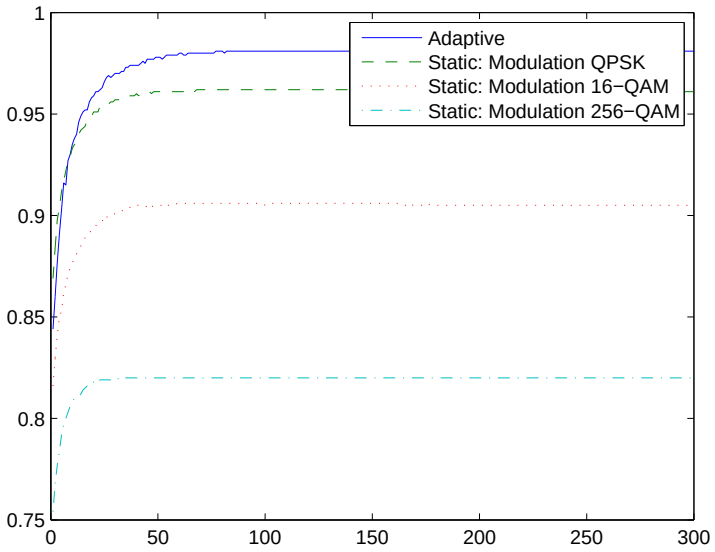

Figure 6.15 Minimize Power: Fitness convergence effect when holding the modulation parameter static at various values versus being completely adaptable.

Table 6.10 Power Scenario: Parameter Sensitivities

Transmission Parameter	**Sensitivity**
Transmission Power	medium
Time Division Duplexing	high
Modulation Index	medium

6.6.2 Emergency Scenario

Our next scenario places the majority of the weight on minimizing the bit-error-rate of the transmission. We expect this scenario to be sensitive to transmission power and modulation parameters. However we also have 20% of the weight on the spectral

efficiency objective. This objective focuses on the number of bits per symbol or the modulation index, also emphasizing importance on the modulation index. Figure 6.16 shows the results of keeping the transmission power static at -8 dBm, 7 dBm and 20 dBm.

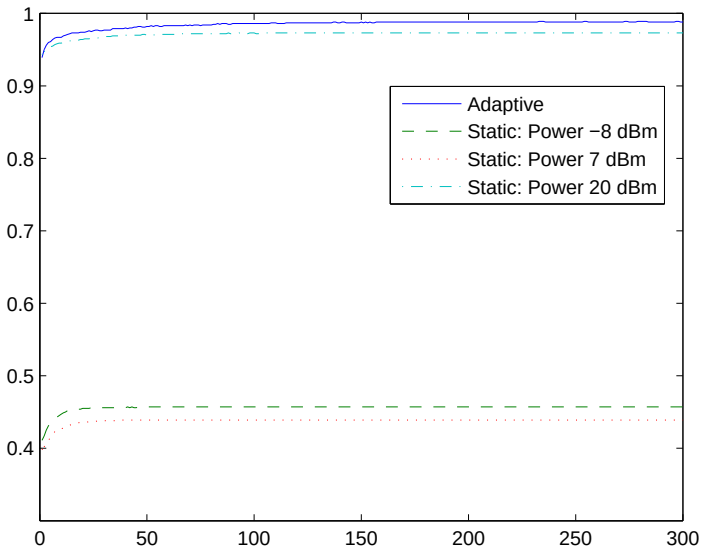

Figure 6.16 Emergency: Fitness convergence effect when holding the transmit power parameter static at various values versus being completely adaptable.

Figure 6.16 shows the wide range of fitness that the transmission power controls in this objective. At the low power value of -8 dBm, the fitness has an upper limit 46% lower then if it were able to be adaptable. This highly elastic fitness convergence curve makes the transmission power a highly sensitive parameter. Figure 6.17 gives the fitness convergence of keeping the modulation index static at both 4,16 and 32 symbols.

Figure 6.17 shows that the modulation index does not affect the system in the same linear manner as the transmission power. When determining BER the modulation index plays an important role in determining the energy per bit of transmission. At higher

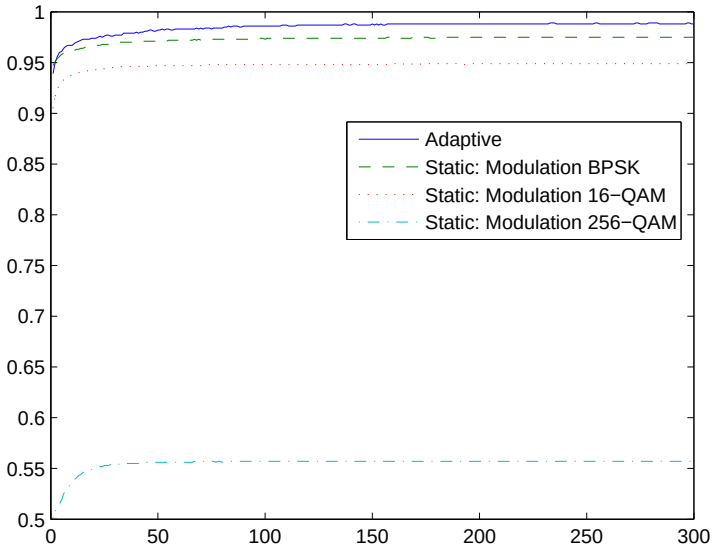

Figure 6.17 Emergency: Fitness convergence effect when holding the modulation parameter static at various values versus being completely adaptable.

modulations the energy per bit is spread thin, causing lower BER resulting in a low fitness. We consider this high sensitivity because a large change in fitness results from changing the modulation index. Figure 6.18 gives the fitness convergence of keeping the symbol rate static at 125000, 500000 and 1000000.

From the symbol rate sensitivity shown in Figure 6.18, we can see that the adaptability of the symbol rate has little affect on the fitness when the focus is on minimizing the BER. This result is expected because of the fact that the symbol rate has little affect on the system BER, and little weight is given to other objectives that use symbol rate.

Table 6.11 shows the summary of the explored parameters and the sensitivities that they exhibit in the emergency scenario.

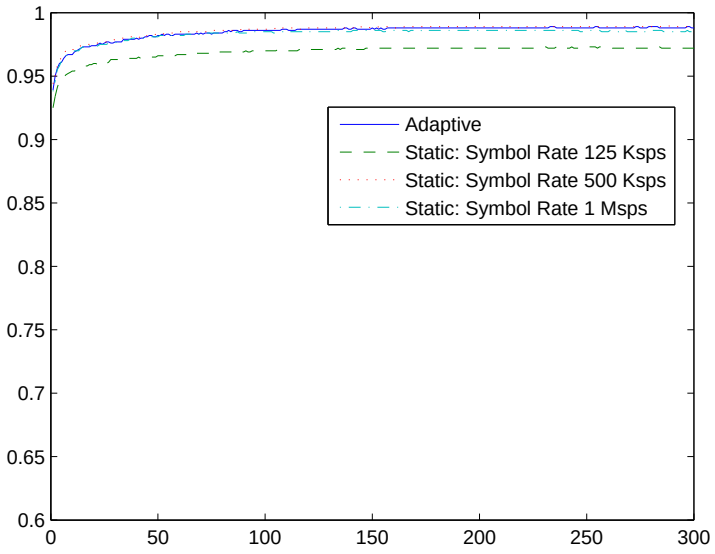

Figure 6.18 Emergency: Fitness convergence effect when holding the symbol rate parameter static at various values versus being completely adaptable.

Table 6.11 Emergency Scenario: Parameter Sensitivities

Transmission Parameter	**Sensitivity**
Transmission Power	high
Symbol Rate	low
Modulation Index	high

6.6.3 Multimedia Scenario

The third scenario focuses on multimedia applications that require high throughput. Maximizing the data throughput is the primary focus of this objective, with 50% of the weight on the maximize throughput objective. We expect that the modulation index and frame length will be the most sensitive parameters in this scenario. Figure 6.19 shows the results of keeping the transmission power static at -8, 10 and 20 dBm. We again see from the Figure that certain values of transmission power result in a fitness upper limit that is 50% lower then the limit that results from a fully adaptable system. This results again in transmission power being a highly sensitive parameter in the multimedia scenario.

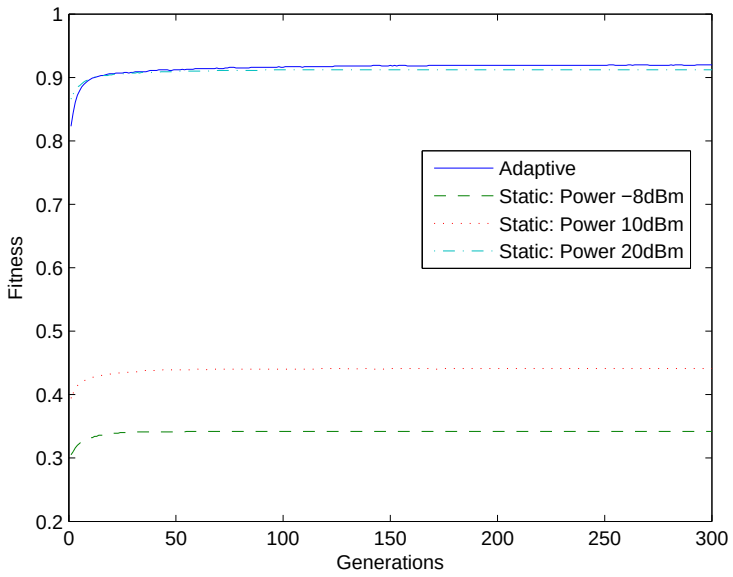

Figure 6.19 Multimedia: Fitness convergence effect when holding the transmit power parameter static at various values versus being completely adaptable.

Figure 6.20 gives the fitness convergence of keeping the modulation index static at 2, 8 and 32 symbols.

Figure 6.21 gives the fitness convergence of keeping the symbol rate static at 125000,

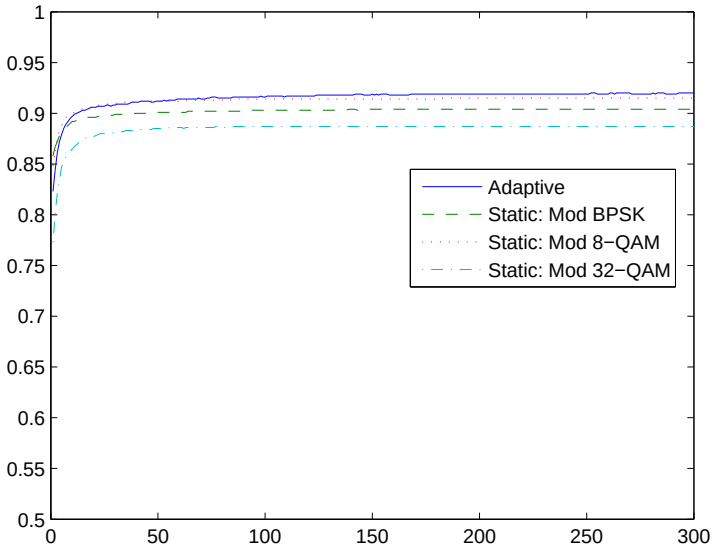

Figure 6.20 Multimedia: Fitness convergence effect when holding the modulation parameter static at various values versus being completely adaptable

500000 and 1000000 symbols per second. The fitness deviates over a 0.05 range when the symbol rate is held constant at the defined values.

Figure 6.22 gives the fitness convergence of keeping the frame length static at 100, 700 and 1500 bytes. We see from the Figure that at lower frame lengths we typically get lower fitness scores, about 30% lower.

Table 6.12 Multimedia Scenario: Parameter Sensitivities

Transmission Parameter	**Sensitivity**
Transmission Power	high
Symbol Rate	low
Frame Length	high
Modulation Index	low

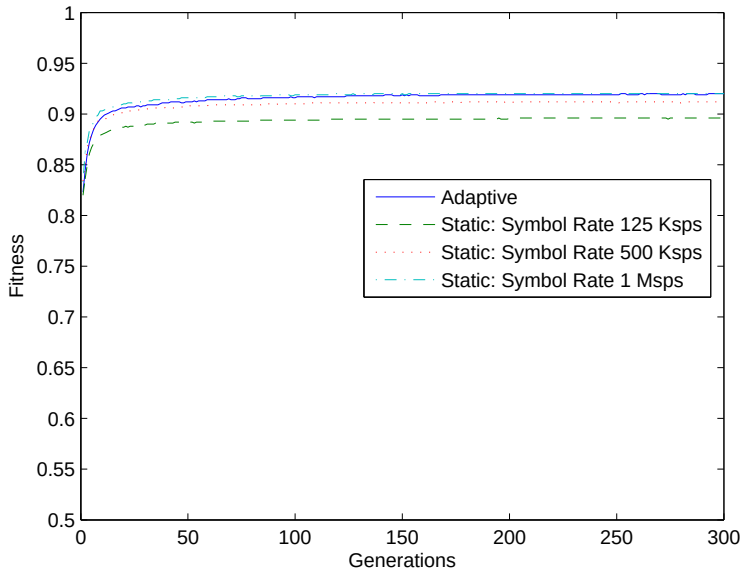

Figure 6.21 Multimedia: Fitness convergence effect when holding the symbol rate parameter static at various values versus being completely adaptable

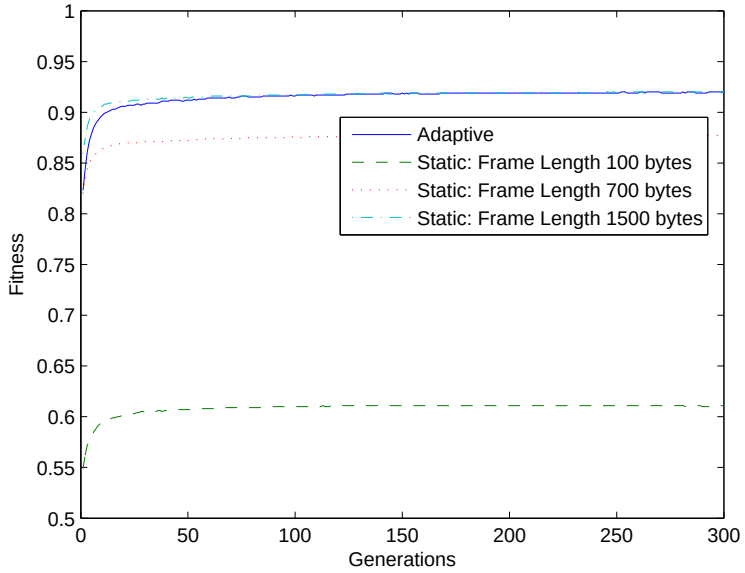

Figure 6.22 Multimedia: Fitness convergence effect when holding the frame length parameter static at various values versus being completely adaptable

6.6.4 DSA Scenario

The final scenario we will explore is the dynamic spectrum allocation scenario. This performance objective focuses on reducing the amount of spectral interference that the system adds to the wireless spectrum as a hole. This can be done by lowering the transmission power of communications which dampens the overall noise power in reference to others in the same band. Interference can also be reduced by decreasing the transmission bandwidth of communication which in turn reduces the total throughput of the system, but allows others to communicate with less interference in neighboring bands. The most obvious way to reduce interference is to simply not transmit. This will be seen in the *TDD* sensitivity which as shown in Figure 6.23 has high sensitivity for this objective.

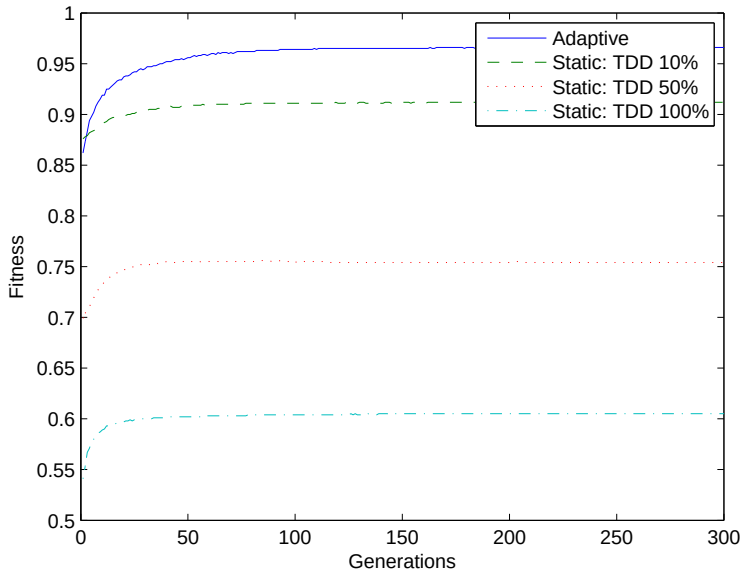

Figure 6.23 DSA: Fitness convergence effect when holding the TDD parameter static at various values versus being completely adaptable

Figure 6.24 gives the fitness convergence of keeping the transmission static at -8, 10 and 20 dBm. Unlike the previous scenarios, the transmission power does not affect

the system as much in this scenario. In addition the result of changing the power is a non-linear change in the fitness cap. The low cap is with the 7 dBm power, with the 20 dBm slightly higher and the -8 dBm cap is near the adaptable system. This non-lineary occurs because of the affect transmission power has on the other performance objectives. In this scenario, a lower transmission power will result in less interference, however, the throughput will also be lower. We have a weight of .20 on the maximize throughput which gives higher fitness to higher throughput.

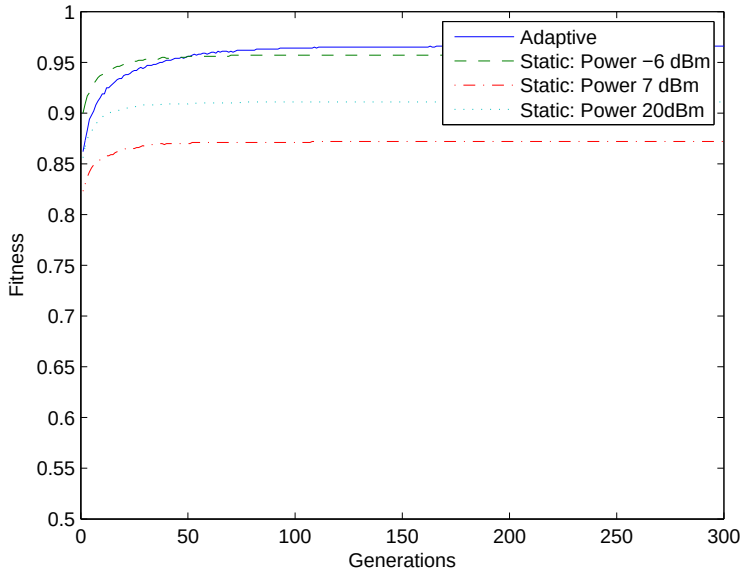

Figure 6.24 DSA: Fitness convergence effect when holding the transmit power parameter static at various values versus being completely adaptable

Figure 6.25 gives the fitness convergence of keeping the bandwidth static at 2, 15 and 30 MHz. We see from this Figure that varying the bandwidth can significantly reduce the upper fitness cap. A drop of about 23% occurs when the bandwidth is set at the maximum of 30 MHz, resulting in a large amount of interference. This verifies that in this scenario, the bandwidth is highly sensitive.

Figure 6.26 gives the fitness convergence of keeping the frame length static at 100, 700 and 1500 bytes. You can see that there is little to no deviation between all four

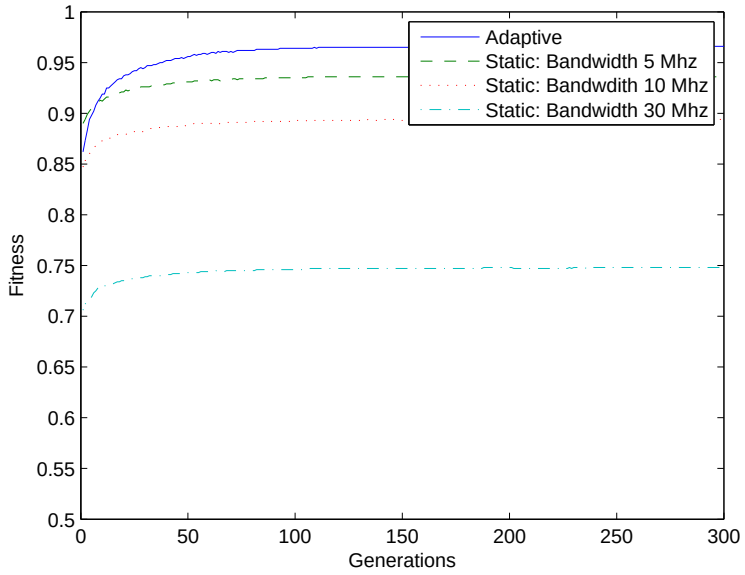

Figure 6.25 DSA: Fitness convergence effect when holding the bandwidth parameter static at various values versus being completely adaptable

of the fitness convergence curves. This implies that the frame length parameter has very little sensitivity to the DSA performance objective. We can take advantage of this information when implementing other cognitive radio systems, whether they be GA-based, RBS-based, or even Case based reasoning systems, by disregarding any computation involving the frame length when the system is in this mode. This can be done by removing the parameter from any rules, which would decrease the size of the rule base, or ignore the frame length parameter when trying to match specific cases in a CBR system.

Again we provide a summary for this objective in both Figure 6.26, which shows graphically how the fitness is affected by the worst static parameter settings we explored, and Table 6.13 which shows the summary of the sensitivities of each of the parameters described in this section.

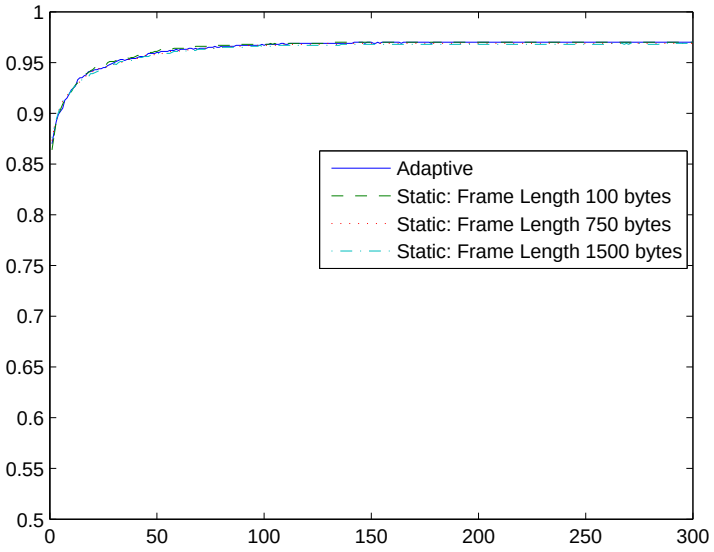

Figure 6.26 DSA: Fitness convergence effect when holding the frame length parameter static at various values versus being completely adaptable

Table 6.13 DSA Scenario: Parameter Sensitivities

Transmission Parameter	**Sensitivity**
Transmission Power	medium
TDD	high
Frame Length	low
Bandwidth	high

Chapter 7

Conclusion

7.1 Research Achievements

In this disseration a number of contributions have been made in the area of cognitive engine adapatation techniques. The research achievements of this disseration are the following:

- Multiple objective fitness functions representing the relationships between the transmission parameters, environmental measurements, and performance objectives were developed. Using theoretically relationships between several different parameters and the weighted aggregate sum approach, equations for each performance objective were developed to be used within cognitive adaptation engines. The objectives of these multiple fitness functions are controlled by the values of the weights on each function.

- These equations were implemented by two different machine learning techniques that were used in a cognitive adaptation engine. From this implementation, we explored the advantages and disadvantages of the genetic algorithm approach and the rule based system approach. We discovered that the genetic algorithm

approach was more flexible and provided an interface that allows the user to easily adjust system parameters. However the genetic algorithm suffered from a high requirement for processing power which caused it to perform slower then the RBS. The RBS was able to provide solutions very fast while needing a lower amount of memory for storage then expected. However, the need for a static database of rules restricted the flexiblity of the RBS.

- We developed an adaptive improvement to a standard genetic algorithm initialization procedure that improves the speed of convergence of the genetic algorithm. Based on the observed change in the environment, this adaptive technique biases the initial generation of the GA population to lean toward the solution of the previous run. This technique improves performance only if the environment has not changed significantly, and as long as the appropiate amount of chromosomes are seeded.

- Sensitivity analysis was performed on several parameters in the system. We showed that this sensitivity varied based on the specific performance objective of the system. Sensitivity analysis uncovered the parameters that have little effect on the system. This could allow wireless system designers to design a system without the less sensitive parameters in order to lower system complexity and resource usage. High sensitivity parameters were also shown to exist. These parameters have a large effect on the system when altered. This information can be used to suggest which parameters be used in the cognitive adaptation process.

The list of publications related to the work presented in this disseration is as follows:

Book Chapter

B1. Timothy Newman, Alexander M. Wyglinski, and Joseph Evans, Cognitive Radio Implementation for Efficient Wireless Communication. *Encyclopedia on Wireless and Mobile Communications*, Borko Furht, Editor, CRC Press, 2007.

B2. Timothy Newman, Muthukumaran Pitchaimani, Benjamin Ewy, and Joseph Evans, Architectures for Cognition in Radio Networks, *Invited submission to Cognitive Radio Networks*, Yang Xiao and Fei Hu, editor, CRC Press, 2008.

B3. Joseph Evans and Timothy Newman, VLSI Implementations of Digital Filters. *Invited section in Circuit and Filter Handbook, 2nd Edition*, W. K. Chen, editor-in-chief, CRC Press, 2008.

Journal Papers

J1. Timothy R. Newman, Rakesh Rajbanshi, Alexander M. Wyglinski, Joseph B. Evans, and Gary J. Minden, "Population Adaptation for Genetic Algorithm-based Cognitive Radios," *ACM/Springer Mobile Ad Hoc Networks – Special Issue on Cognitive Radio Oriented Wireless Networks and Communications*, 2008.

J2. G. J. Minden, J. B. Evans, L. Searl, D. DePardo, R. Rajbanshi, J. Guffey, Q. Chen, T. Newman, V. R. Petty, F. Weidling, M. Lehnherr, B. Cordill, D. Datla, B. Barker, and A. Agah, "An agile radio for wireless innovation," *IEEE Commun. Mag.*, May 2007.

J3. Timothy R. Newman, Brett A. Barker, Alexander M. Wyglinski, Arvin Agah, Joseph B. Evans, and Gary J. Minden, " Cognitive Engine Implementation for Wireless Multicarrier Transceivers," *Wiley Journal on Wireless Communications and Mobile Computing*, vol 7. (9), November 2007.

Conference Papers

C1. Timothy R. Newman, Rakesh Rajbanshi, Alexander M. Wyglinski, Joseph B. Evans, and Gary J. Minden, "Population Adaptation for Genetic Algorithm-based Cognitive Radios," *Proceedings of the Second International Conference on Cognitive Radio Oriented Wireless Networks and Communications* (Orlando, FL, USA), August 2007.

C2. Gary J. Minden, Joseph B. Evans, Leon Searl, Daniel DePardo, Victor R. Petty, Rakesh Rajbanshi, Jordan Guffey, Qi Chen, Timothy R. Newman, Frederick Weidling, Dinesh Datla, Brett Barker, Megan Peck, Brian Cordill, Alexander M. Wyglinski, and Arvin Agah, "KUAR: A Flexible Software-Defined Radio Development Platform," *Second IEEE Symposium on New Frontiers in Dynamic Spectrum Access Networks* Dublin, Ireland), November 2006.

C3. Timothy R. Newman, and Gary J. Minden, "A Software Defined Radio Architecture Model to Develop Radio Modem Component Classifications," *IEEE Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN '05)* Baltimore, MD, Novemeber 8-11, 2005.

7.2 Future Work

There exists a number of topics resulting from this research that can be continued.

- Several other possible machine learning techniques exist that can be implemented in addition to the techniques explored in this work. Techniques such as case-based reasoning systems or neural networks can be used as the adaptation technique or be implemented in tandem with the techniques described in this disseration. For example, using case-based reasoning systems to remember "good" solu-

tions to specific environments and use this information to seed the GA could possibly improve performance of the system if it encounters environments it hasn't seen in long periods of time.

- We explored four different weighting scenarios that emphasized different performance objectives. We noted on several occasions such as the sensitivity analysis that the results differed based upon the weight combinations. Lots of room exists to explore a more wide range of weight combinations in order to fully uncover how the performance objectives affect different aspects of the cognitive adaptation.

- Our system operates without the need for feedback information from the network or the communication partner. The fitness functions were developed using theoretical equations which are not exact and sometimes can not approximate the environment correctly. With feedback from the the opposite communication node, the cognitive adaptation engine would have extended information about how well the current settings are working. Using this information, other adaptation techniques can be used to alter the parameters if the theoretical fitness functions are not modeling the environment properly.

- Fitness function improvements can be made in order to allow the system to model a wider range of environments. Implementing BER functions that model other environments such as the ones described in Appendix B give the system more flexibility to work in additional environments.

Appendix A

MATLAB Rule Generation

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%  Set variable values
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

v = .10;
w = .50;
x = .10;
y = .10;
z = .20;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%  Constant Values
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

mac_oh = 40;         % MAC Layer overhead in (Bytes)
phy_oh = 52.5;       % PHY Layer overhead (Bytes)
kb = 1.38 * 10^-23; % Boltzmann's constant (J/K)
T = 290;             % System noise temperatue (K)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%  Minimum and Maximum values
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
power_max = 24;          % Maximum Power (dBm)
power_min = -8;           % Minimum Power (dBm)

bandwidth_max = 34; % Maximum bandwidth (MHz)
```

```
bandwidth_min = 2;  % Minimum bandwidth (MHz)

mod_max = 256;      % Maximum modulation index.
mod_min = 2;

symbol_rate_max = 1000000;  % Maximum symbol rate (Symbols / second)
symbol_rate_min = 125000;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%  Transmission paramet er ranges
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
power = [power_min:2:power_max];
bandwidth = [bandwidth_min:2:bandwidth_max];
tdd = [1:10:100];
mod_index = [ 2 4 16 64 256 ];
symbol_rate = [125000 250000 500000 625000 750000 1000000];
frame_length = [100 : 100 : 1500 ];
coding_rate = [1 1/2 3/4 5/6];
mod_type = {'qam','psk','pam'};

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%  Environmental parameter ranges
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
path_loss = [87:1:92];
noise_power = [-117:1:-112];

% Total number of parameter combinations
combinations = length(power) * length(bandwidth) ...
    * length(tdd) * length(mod_index) * ...
    length(symbol_rate) * length(frame_length) ...
    * length(mod_type) * length(coding_rate) * ...
     length(path_loss) * length(noise_power)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%  Fitness Function
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Initialize the indexes that hold the position in the array
mod_index_index = 1;
tdd_index = 1;
symbol_rate_index = 1;
```

```
power_index = 1;
bandwidth_index = 1;
mod_type_index = 1;
coding_rate_index = 1;
frame_length_index = 1;
snr_index = 1;
np_index = 1;
pl_index = 1;

fitness_array = [ ];


% Go through all possible combinations of transmission parameters and
% calculate the fitness.  These fitness values are held in a 10-dimensional
% array indexed by the parameter indexes defined above.

for p = power
 for b = bandwidth
  for m = mod_index
   for pl = path_loss
    for sr = symbol_rate
     for np = noise_power
      for cr = coding_rate
       for mt = mod_type
        for t = tdd
         for f = frame_length
          %% Populate the fitness array with
          %% outputs from the fitness
          %% function
          fitness_array(np_index,pl_index,power_index,
          bandwidth_index,tdd_index,mod_index_index,
          symbol_rate_index,frame_length_index,
          coding_rate_index,mod_type_index) = ...

          fitness_score(np,pl,p,b,t,m,sr,f,cr,char(mt),v,w,x,y,z,
          power_max,power_min,symbol_rate_max,symbol_rate_min,
          bandwidth_max,bandwidth_min,mod_max,mod_min);

          frame_length_index = frame_length_index + 1;
         end
         tdd_index = tdd_index + 1;
         frame_length_index = 1;
```

```
        end
        tdd_index = 1;
        mod_type_index = mod_type_index + 1;
       end
       mod_type_index = 1;
       coding_rate_index = coding_rate_index + 1;
      end
      coding_rate_index = 1;
      np_index =np_index + 1;
     end
     np_index = 1;
     symbol_rate_index = symbol_rate_index + 1;
    end
    symbol_rate_index = 1;
    pl_index = pl_index + 1;
   end
   pl_index = 1;
   mod_index_index = mod_index_index + 1;
  end
  mod_index_index = 1;
  bandwidth_index = bandwidth_index + 1;
 end
 power_index = power_index + 1;
 bandwidth_index = 1;
end

% Determine the maximum,minimum, and range of fitness
% for each environment.

for i = 1:1:6
    for j = 1:1:6
        A = fitness_array(i,j,:,:,:,:,:,:,:,:);
        max_fitness(i,j) = max(A(:));
        min_fitness(i,j) = min(A(:));
        range_fitness(i,j) = max(A(:)) - min(A(:));
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%  Open CLIPS rules file for writing
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
filename = sprintf('rules/Cognitive_Rules_emerg_1.clp');

fid = fopen(filename, 'w');
if (fid == -1)
    error('write_file: cannot open file for writing');
end

% Loop through each environment and create a CLIPS rule for the
% optimal fitness found.

for i = 1:1:6
    for j = 1:1:6
        rule_count = rule_count + 1;

        % Find the max fitness and search for the corresponding
        % transmission parameters in the array.

        A = fitness_array(i,j,:,:,:,:,:,:,:,:);
        [ o k ] = max(A(:));
        [np_index pl_index power_index bandwidth_index
        tdd_index mod_index_index symbol_rate_index
        frame_length_index coding_rate_index mod_type_index]
        = ind2sub(size(A),k);

        % Translate the index found for the max fitness
        % into real parameter values.

        p = power(power_index);
        b = bandwidth(bandwidth_index);
        t = tdd(tdd_index);
        mi = mod_index(mod_index_index);
        sr = symbol_rate(symbol_rate_index);
        f = frame_length(frame_length_index);
        cr = coding_rate(coding_rate_index);
        mt = char(mod_type(mod_type_index));

     %% Print rules to the CLIPS rules file
     fprintf(fid, '(defrule cognitive_rule_%i\n',rule_count);
     fprintf(fid, '  (noise_power %i)\n',-118+i);
     fprintf(fid, '  (path_loss %i)\n',86+j);
     fprintf(fid, '  (channel_number ?channel_num)\n');
     fprintf(fid, '  (scenario emerg_mode)\n');
```

```
      fprintf(fid, '=>\n');
      fprintf(fid, '  (assert (channel ?channel_num %2.0f
      %2.0f %3.0f %s %4.0f %1.2f %7.0f %3.0f)) )\n\n\n',
      p,b,mi,mt,f,cr,sr,t);

    end
end
```

Appendix B

BER Equations

Section 5.4.1.1 gave several basic BER equations including M-ary PSK and M-ary QAM forumlas in AWGN channels. However, we present more forumlas in this appendix because different channel types exist other than AWGN. We present breifly cover the derivation of BER forumlas and present equations for Raleigh and Rician fading channels.

For fading channels in general, [66] provides generalized closed form equations for BPSK, M-QAM, and M-PSK modulations. Each forumla follows the general quation format for the probability of a symbol error as shown in B.1.

$$P_e = \int_0^\infty P_{AWGN}(x)p(x)dx \tag{B.1}$$

Where $p(x)$ is the probability density function (PDF) of the channel. For each of the following equations, $I(\overline{\gamma}, g, \theta)$, represents specific definite integrals for the Rayleigh and Rician fading channels, where $\overline{\gamma}$ is the average signal to noise ratio, g is a modulation coefficient, and θ is the variable of integration.

Rayleigh:

$$p(\gamma,\overline{\gamma}) = \frac{1}{\overline{\gamma}} exp\left(-\frac{\gamma}{\overline{\gamma}}\right) \tag{B.2}$$

$$I(\overline{\gamma},g,\theta) = \left(1+\frac{g\overline{\gamma}}{sin^2\theta}\right)^{-1} \tag{B.3}$$

Rician:

$$p(\gamma,\overline{\gamma},n) = \frac{\left(1+n^2\right)e^{-n^2}}{\overline{\gamma}} exp\left(\frac{\left(1+n^2\right)\gamma}{\overline{\gamma}}\right) I_0\left(2n\alpha\sqrt{\frac{\left(1+n^2\right)\gamma}{\overline{\gamma}}}\right) \tag{B.4}$$

where n^2 is the Ricean factor

$$I(\overline{\gamma},g,\theta) = \frac{\left(1+n^2\right)sin^2\theta}{(1+n^2)sin^2\theta+g\overline{\gamma}} exp\left(-\frac{n^2 g\overline{\gamma}}{(1+n^2)sin^2\theta+g\overline{\gamma}}\right) \tag{B.5}$$

[66] uses these PDFs and definite integrals to defined the generalized closed form equations for the different modulation schemes as shown in the following equations.

BPSK:

$$P_e = \frac{1}{\pi}\int_0^{\frac{\pi}{2}} I(\overline{\gamma},g,\theta)d\theta \tag{B.6}$$

where $g = 1$ for BPSK

M-PSK:

$$P_e = \frac{1}{\pi}\int_0^{\frac{(M-1)\pi}{M}} I(\overline{\gamma},g,\theta)d\theta \tag{B.7}$$

where $g = sin^2\left(\frac{\pi}{M}\right)$ for M-PSK

M-QAM:

$$P_e = \frac{4}{\pi}\left(1-\frac{1}{\sqrt{M}}\right)\int_0^{\frac{\pi}{2}} I(\overline{\gamma},g,\theta)d\theta - \frac{4}{\pi}\left(1-\frac{1}{\sqrt{M}}\right)^2\int_0^{\frac{\pi}{4}} I(\overline{\gamma},g,\theta)d\theta \tag{B.8}$$

where $g = \frac{3}{2(M-1)}$ for M-QAM

As mentioned in the future work, new fitness functions can be defined that take into account these BER models. This would extend the amount of useable environments that these cognitive adaptation engines can effectively operate within.

References

[1] Joint Tactical Radio Systems, "Software communications architecture specification," November 2002.

[2] C. J. Rieser, "Biologically inspired cognitive radio engine model utilizing distributed genetic algorithms for secure and robust wireless communications and networking," Ph.D. dissertation, Virginia Polytechnic Institute and State University, April 2004.

[3] A. P. R. Etkin and D. Tse, "Spectrum sharing for unlicensed bands," in *IEEE International Symposium on New Frontiers in Dynamic Spectrum Access*, 2005.

[4] Spectrum Policy Task Force, "Report of the spectrum policy workgroup," Novemeber 2002. [Online]. Available: http://www.fcc.gov/sptf/files/SEWGFinalReport_1.pdf

[5] S. M. Mishra, D. Cabric, and C. Chang, "A real time cognitive radio testbed for physical and link layer experiments," in *IEEE International Symposium on New Frontiers in Dynamic Spectrum Access*, 2005, pp. 562–567.

[6] J. Chapin and V. Bose, "The vanu software radio system," in *Software Defined Radio Technical Conference*, November 2002.

[7] J. G. Proakis, *Digital Communications*, 4th ed. McGraw-Hill, 2000.

[8] J. P. Ignizio, *Introduction to Expert Systems: The Development and Implementation of Rule-Based Expert Systems*. McGraw-Hill, 1991.

[9] B. G. Buchanan and E. H. Shortliffe, *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*. Addison-Wesley, 1985.

[10] R. Barletta, "An introduction to case-based reasoning," *AI Expert*, vol. 6, no. 8, pp. 42–49, 1991.

[11] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[12] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison Wesley, 1989.

[13] J. H. Holland, *Adaptation in natural and artificial systems*. MIT Press, 1992.

[14] C. M. Fonseca and P. J. Fleming, "An overview of evolutionary algorithms in multiobjective optimization," *Evolutionary Computation*, vol. 3, no. 1, pp. 1–16, 1995.

[15] P. J. Fleming, "Designing control systems with multiple objectives," in *IEE Colloquium Advances in Control Technology*, 1999, pp. 4/1–4/4.

[16] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[17] L. Zadeh, "Optimality and non-scalar-valued performance criteria," *IEEE Transactions on Automatic Control*, vol. 8, pp. 59–60, 1963.

[18] A. Goicoechea, D. Hansen, and L. Duckstein, *Multiobjective Decision Analysis with Engineering and Business Applications*. John Wiley and Sons, 1982.

[19] R. Steuer, *Multiple Criteria Optimization: Theory, Computation, and Application*. Robert E. Krieger Publishing, 1989.

[20] K. Yoon and C. Hwang, *Multiple Attribute Decision Making, and Introduction*. Sage Publications, 1995.

[21] C. Rieser, T. Rondeau, C. Bostian, and T. Gallagher, "Cognitive radio testbed: further details and testing of a distributed genetic algorithm based cognitive engine for programmable radios," in *Military Communications Conference, MILCOM*, 2004.

[22] J. Mitola III, "An integrated agent architecture for software defined radio," Ph.D. dissertation, Royal Institute of Technology (KTH), May 2000.

[23] Vanu Inc., "Vanu anywave base station." [Online]. Available: http://www.vanu.com/products/basestation.html

[24] Free Software Foundation, "GNU Radio 2007." [Online]. Available: http://www.gnuradio.org

[25] The Mobile and Portable Radio Research Group, "Ossie open sca implementation." [Online]. Available: http://ossie.mprg.org/

[26] D. I. Kim, E. Hossain, and V. Bhargava, "Dynamic rate and power adaptation for provisioning class-based QoS in Cellular Multirate WCDMA systems," *IEEE Transactions on Wireless Communications*, vol. 3, no. 4, pp. 1590–1601, 2004.

[27] C. Kose and D. Goeckel, "On power adaptation in adaptive signaling systems," *IEEE Transactions on Communications*, vol. 48, no. 11, pp. 1769–1773, 2000.

[28] J. Paris, M. del Carmen Aguayo-Torres, and J. Entrambasaguas, "Optimum discrete-power adaptive qam scheme for rayleigh fading channels," *IEEE Transactions on Communications*, vol. 5, no. 7, pp. 281–283, 2001.

[29] S. T. Chung and A. Goldsmith, "Degrees of freedom in adaptive modulation: a unified view," *IEEE Transactions on Communications*, vol. 49, no. 9, pp. 1561–1571, 2001.

[30] S. Falahati, A. Svensson, T. Ekman, and M. Sternad, "Adaptive modulation systems for predicted wireless channels," *IEEE Transactions on Communications*, vol. 52, no. 2, pp. 307–316, 2004.

[31] A. M. Wyglinski, F. Labeau, and P. Kabal, "Bit loading with ber-constraint for multicarrier systems," *IEEE Transactions on Wireless Communications*, vol. 4, no. 4, pp. 1383–1387, July 2005.

[32] H. Chen, "Machine learning for information retrieval: neural networks, symbolic learning, and genetic algorithms," *J. Am. Soc. Inf. Sci.*, vol. 46, no. 3, pp. 194–216, 1995.

[33] R. P. Lippmann, "An introduction to computing with neural nets," pp. 36–54, 1988.

[34] J. Carbonell, R. Michalski, and T. Mitchell, "An overview of machine lerning." [Online]. Available: citeseer.ist.psu.edu/527935.html

[35] A. Agnar and P. Enric, "Case-based reasoning : Foundational issues, methodological variations, and system approaches," 1994.

[36] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science, Number 4598, 13 May 1983*, vol. 220, 4598, pp. 671–680, 1983.

[37] "CLIPS expert system web site." [Online]. Available: http://www.ghg.net/clips/CLIPS.html

[38] "Nutech solutions - intelligent business engines." [Online]. Available: http://nutechsolutions.com/

[39] M. Moustafa, I. Habib, and M. Naghshineh, "Wireless resource management using genetic algorithm for mobiles equilibrium," in *Sixth IEEE Symposium on Computers and Communications*, 2001.

[40] M. Y. Alias, S. Chen, and L. Hanzo, "Multiple-antenna-aided ofdm employing genetic-algorithm-assisted minimum bit error rate multiuser detection," *IEEE Transactions on Vehicular Technology*, vol. 54, no. 5, pp. 1713–1721, 2005.

[41] D. E. Goldberg, *A comparative analysis of selection schemes used in genetic algorithms*. Morgan Kaufmann, 1991.

[42] "DARPA XG program web site." [Online]. Available: http://www.darpa.mil/ato/programs/XG/

[43] S. Yarkan and H. Arslan, "Exploiting location awareness toward improved wireless system design in cognitive radio," *Communications Magazine, IEEE*, vol. 46, no. 1, pp. 128–136, January 2008.

[44] T. Weingart, D. Sicker, and D. Grunwald, "A statistical method for reconfiguration of cognitive radios," *Wireless Communications, IEEE [see also IEEE Personal Communications]*, vol. 14, no. 4, pp. 34–40, August 2007.

[45] M. Pursley and T. Royster, "Low-complexity adaptive transmission for cognitive radios in dynamic spectrum access networks," *Selected Areas in Communications, IEEE Journal on*, vol. 26, no. 1, pp. 83–94, Jan. 2008.

[46] H. Arslan and S. Yarkan, *Enabling Cognitive Radio Through Sensing, Awareness, and Measurements*. Springer, 2007.

[47] R. Rajbanshi, A. M. Wyglinski, and G. J. Minden, "An efficient implementation of NC-OFDM transceivers for cognitive radios," in *Proceedings of the First International Conference on Cognitive Radio Oriented Wireless Networks and Communications*, Mykonos, Greece, Jun. 2006.

[48] R. Rajbanshi, Q. Chen, A. M. Wyglinski, J. B. Evans, and G. J. Minden, "Comparative study of frequency agile data transmission schemes for cognitive radio transceivers," in *First International Workshop on Technology and Policy for Accessing Spectrum*, Boston, MA, USA, Aug. 2006.

[49] S. Forrest, "Genetic algorithms," *ACM Computing*, vol. 28, no. 1, pp. 77–80, March 1996.

[50] K. DeJong and W. Spears, "An analysis of the interacting roles of population size and crossover in genetic algorithms," in *Proc. First Workshop Parallel Problem Solving from Nature*, 1990.

[51] J. W. Myers, K. B. Laskey, and K. A. DeJong, "Learning bayesian networks from incomplete data using evolutionary algorithms," in *Proceedings of the Genetic and Evolutionary Computation Conference*, vol. 1, 13-17 1999, pp. 458–465.

[52] D. G. Li and A. C. Watson, "Genetic algorithms in optical thin film optimization design," in *ICCIMA '99: Proceedings of the 3rd International Conference on Computational Intelligence and Multimedia Applications*, 1999, p. 86.

[53] M. Srinivas and L. Patnaik, "Genetic algorithms: a survey," *Computer Magazine*, vol. 27, no. 6, pp. 17–26, 1994.

[54] B. A. Julstrom, “Seeding the population: improved performance in a genetic algorithm for the rectilinear steiner problem,” in *SAC*, 1994, pp. 222–226.

[55] A. E. Hans, “Multicriteria optimization for highly accurate systems,” in *Multicriteria Optimization in Engineering and Sciences*, 1988, pp. 309–352.

[56] M. Schniederjans, *Goal programming : methodology and applications*, K. A. Publishers, Ed. Kluwer Academic Publishers, 1995.

[57] J. Ignizio, *Goal Programming and Extensions*, L. Books, Ed. Lexington Books, 1976.

[58] P. Lettieri and M. B. Srivastava, “Adaptive frame length control for improving wireless link throughput, range and energy efficiency,” in *INFOCOM (2)*, 1998, pp. 564–571. [Online]. Available: citeseer.ist.psu.edu/lettieri98adaptive.html

[59] G. J. Minden, J. B. Evans, L. S. Searl, D. DePardo, R. Rajbanshi, J. Guffey, Q. Chen, T. R. Newman, V. R. Petty, F. Weidling, M. Peck, B. Cordill, D. Datla, B. Barker, and A. Agah, “Cognitive radios for dynamic spectrum access - an agile radio for wireless innovation,” *Communications Magazine, IEEE*, vol. 45, no. 5, pp. 113–121, May 2007.

[60] K. S. Shanmugan, *Random Signals: Detection Estimation and Data Analysis*, Wiley, Ed. Wiley, 1988.

[61] R. Smith, D. Goldberg, and J. Earickson, “Sga-c: A c-language implementation of a simple genetic algorithm,” 1994. [Online]. Available: citeseer.ist.psu.edu/smith91sgac.html

[62] D. E. Goldberg, K. Deb, and J. Horn, “Massive multimodality, deception, and genetic algorithms,” in *Parallel Problem Solving from Nature, 2*, 1992.

[63] High Speed Packet Access, “Mobile Broadband Today.” [Online]. Available: http://hspa.gsmworld.com/

[64] T. Newman, R. Rajbanshi, A. Wyglinski, J. Evans, and G. Minden, “Population adaptation for genetic algorithm-based cognitive radios,” in *Processings of the Second International Conference on Cognitive Radio Oriented Wireless Networks and Communications*, 2007.

[65] ——, “Population adaptation for genetic algorithm-based cognitive radios,” *ACM/Springer Mobile Ad Hoc Networks – Special Issue on Cognitive Radio Oriented Wireless Networks and Communications*, 2008.

[66] M. Simon and M. Alouini, “A unified approach to the performance analysis of digital communication over generalized fading channels,” *Proceedings of the IEEE*, vol. 86, no. 9, pp. 1860–1877, Sep 1998.