OPTIMIZATION, LEARNING, AND DECISION MAKING IN A COGNITIVE ENGINE

Thomas W. Rondeau (trondeau@vt.edu); Bin Le (binle@vt.edu); David Maldonado (davidm@vt.edu), David Scaperoth (scaperot@vt.edu), Allen B. MacKenzie (mackenab@vt.edu), and Charles W. Bostian (bostian@vt.edu) (Center for Wireless Telecommunications, Virginia Tech, Blacksburg, VA, USA)

ABSTRACT

This paper presents further details and results of the cognitive engine developed at the Center for Wireless Telecommunications (CWT) at Virginia Tech. It provides some general design considerations for a cognitive engine as well as specific implementation details of CWT's cognitive engine. We further present results taken from the operation of the cognitive engine on real radio hardware and show how different sets of objectives can alter the quality of service required by different users and applications.

1. INTRODUCTION

Cognitive radios (CR) offer promises of enhanced future communications. They will play an important role in both improved spectrum utilization and better quality of service (QoS) for many applications and users, such as public safety, the military, and consumers. Cognitive radios represent the use of artificial intelligence (AI) on flexible communications devices, most likely software defined radios (SDR), to enable onboard, real-time optimization of frequency, time, power, and space. The objective is intelligent resource optimization.

This paper discusses the implementation of a cognitive engine: the AI algorithms used to control the reconfigurable radio platform. Section 2 covers the SDR platforms currently operating with the cognitive engine. Section 3 describes the learning and optimization theory of the cognitive engine. Section 4 discusses how the cognitive engine operates with a real radio platform, and Section 5 provides the results of the cognitive engine's control of one of the radios. Section 6 discusses important future development and operation of a cognitive engine for CR.

2. RECONFIGURABLE RADIOS

The success of cognitive radios requires a) a reconfigurable platform and b) intelligent oversight that understands how to reconfigure the platform. The reconfigurable radio platform is still its own challenge, yet it is progressing rapidly. The CWT CR work is particularly interested in and using two current platforms that are making great strides towards the reconfigurability required by a successful cognitive radio. These are the GNU Radio and the IRIS (Implementing Radios in Software) platform from Trinity College's Center for Telecommunications Value-Chain Research (CTVR) [1] [2].

2.1. GNU Radio

The GNU Radio project is an open source project to build a software defined radio. It includes a suite of signal processing blocks that, when connected together, form a flow graph to perform the required SDR capabilities. The RF front end generally relies upon the Universal Software Radio Peripheral (USRP) [3], a motherboard that does basic IF processing of up and down conversion, decimation and interpolation, and filtering, and a set of daughterboards to perform the final analog up and down conversion, filtering, and amplification. The GNU Radio is free and open source, and the USRPs are low cost and attractive for research.

The current status of the GNU Radio provides blocks to do much of the basic digital and analog communications, including narrowband AM, FM, GMSK, BPSK, and QPSK. The GNU Radio and the USRP together provide a range of frequencies and transmit powers (depending on the daughterboard in use), flexible specifications for pieces of the radio such as the pulse shape filter, the AGC loop, and the Costas loop, among others.

2.2. IRIS

The IRIS platform is another SDR we are working with that provides a flexible radio system to control with the cognitive engine. The IRIS system includes many of the same blocks as the GNU Radio, uses the USRP hardware platform (for the same frequency and power capabilities as the GNU Radio), and has the additional blocks for higher order QAM modulations, channel coding, and OFDM. The intent of IRIS is reconfigurability through a simple component structure interface defined by an XML document.

3. THE COGNITIVE ENGINE

Details of the initial implementation of the cognitive engine appear in [4], showing its use with a legacy hardware radio system and a software simulation. The hardware system was far too limited in its reconfigurability, and simulations are a poor measure of a CR's performance. Because the goal of CR is intelligent reconfiguration on all levels of the communications stack, assumptions made in a simulation mask the challenges faced in a real-world radio deployment for which the CR is designed to solve.

What we needed was a reconfigurable radio platform so that the cognitive engine has a much wider range of capabilities to play with and can operate in a real environment. The GNU Radio and IRIS provide this capability, and results of their operation with the cognitive engine are shown later.

3.1. Operational Theory

The cognitive engine performs three major functions: optimization, decision making, and learning. The cognitive engine takes in sensing and monitoring information and produces a new set of parameters that defines the radio's functionality. We call the inputs to a cognitive engine its *meters* and outputs the *knobs* of the radio, equivalent to sensors and actuators of traditional AI systems [5].

The operational theory is best summarized in the cognition loop of Figure 1. In this loop, the radio provides information about the current state of the environment and communications system, quantifying things like channel propagation effects, quality of service metrics like bit error rate (BER) and frame error rate (FER), data rate, and the presence of other radios, either cooperative or interferers. The cognitive engine uses this information to determine any adjustments to the radio. The full system consists of both a learning machine, implemented with case-based learning, and an optimization process, implemented by a multi-objective genetic algorithm (GA). The learning and optimization procedures operate together, but on parallel paths. If the decision maker of the cognitive engine finds that the information stored in the cognitive engine's memory is good enough, the optimization process can be skipped. Memory can also aid the optimization process by providing information to guide and bolster the optimization. In the end, if no information is available for the current situation, the optimization process can start from nothing and still produce a solution.

3.2. Case-Based Learning

The learning and decision making mechanisms follow case-based decision theory [6], which is closely related to case-

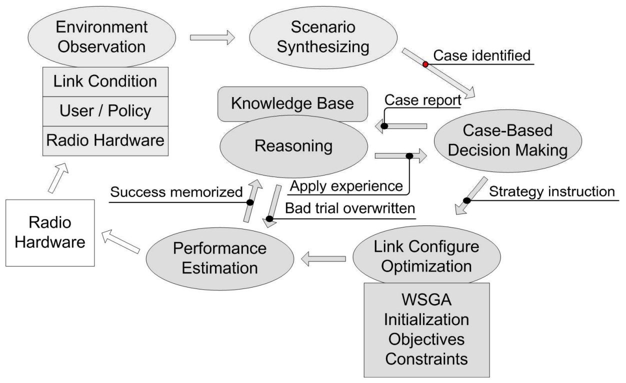


Figure 1. Cognition Loop.

based reasoning [7], or we can just generically refer to it as case-based learning.

Formally, case-based learning defines a set of problems $q \in P$, a set of actions $a \in A$, and a set of results $r \in R$. A case, c, is a tuple of a problem, an action, and a result such that $c \in C$ where $C = P \times A \times R$. Furthermore, memory, M, is formally defined as a set of cases c currently known such that $M \subset C$.

When the sensor system observes a new problem, p, the cognitive engine must determine the action, a, to take in response. The problem input could be a change in channel condition, spectrum use (the presence of a primary user), or a change in the desired quality of service from the user/application domain. To determine the best action to take, the case-base system analyzes the new problem against past cases in memory. The analysis determines how similar the new problem is to the past cases as well as how useful the past actions were in solving the problem. The action defined by the current cognitive engine is the waveform to use defined in the physical (PHY) and medium access control (MAC) layer. As the work progresses, we will extend the actions to incorporate changes at the network, transport, and other higher layers as they are defined for full cross-layer operation.

To discover how similar two cases are, a similarity function is defined, represented in equation 1.

$$s : P \times P \rightarrow [0,1] \qquad (1)$$

The usefulness of the past cases is calculated from a utility function as represented in equation 2.

$$u : R \rightarrow \Re \qquad (2)$$

Case analysis comes down to which case is both most similar to the new problem as well as how successful the action was in the past. We can look at this as a similarity-weighted utility function as shown in equation 3. The resulting chosen case may not be the most similar case if the action of another, less similar case, has performed better.

$$U(a) = s(p,q)u(r) \quad \text{where } (q,a,r) \in M \tag{3}$$

This equation is only one way of analyzing the results of cases. The challenge of this technique is to create effective similarity and utility functions for the types of information received through the sensors. If the cognitive engine has multiple domains of interest, like the propagation, spectrum, and user/application domains, then the case-base must both represent and quantify each of these. Such an analysis is part of our future work and not presented here.

3.3. Genetic Optimization

The main contribution of this paper is our solution to the optimization problem. As stated above, when the case-base produces a suboptimal action to take or possibly no action at all if the cognitive engine is not cognizant of the new problem, then optimization must take place to produce an action, or waveform, suitable for the new problem.

The challenge in the radio optimization is determining the conflicting goals for different levels of QoS. A simple optimization scenario might call for minimizing BER and minimizing power consumption, which are competing objectives. As the radio's power consumption is decreased by turning down the transmitter power or using suboptimal but more computationally efficient signal processing algorithms, the BER will increase. This scenario has two objectives, but many more objectives exist that have dependent relationships such as data rate, occupied bandwidth, spectral efficiency, latency, and computational complexity among others. See chapter seven of [8] for a detailed analysis of the multi-objective nature of radio reconfiguration and [4] [9] for more discussion.

The genetic algorithm [10] must represent both the problem and the solution space effectively. Figure 2 shows part of the chromosome that defines a radio.

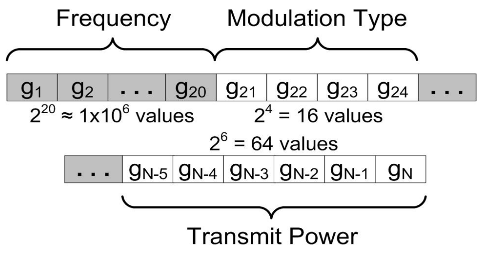

Frequency
Modulation Type
g_1 g_2 . . . g_{20} g_{21} g_{22} g_{23} g_{24} . . .
$2^{20} \approx 1\text{x}10^6$ values
$2^4 = 16$ values
$2^6 = 64$ values
. . . g_{N-5} g_{N-4} g_{N-3} g_{N-2} g_{N-1} g_N
Transmit Power

Figure 2. Genetic algorithm chromosome structure.

This particular chromosome structure deviates from most traditional GAs because of the variable number of bits used to represent any gene. This gives the algorithm a large amount of flexibility when representing real parameters where a radio might be capable of thousands of center frequencies over multiple GHz but only has a few modulations from which to choose. The chromosome can therefore give 20 or so bits to the frequency gene and only 4 to the modulation gene. A key result of this structure is that it makes the GA independent of what radio it is optimizing as the chromosome is developed at run time based on simple hardware specifications of the radio fed in via XML during initialization [11].

Analysis and optimization of multi-objective problems is complex but with a rich history over the past three decades [12]. The general analysis comes down to finding the non-dominated solutions in the solution space, which is known as the Pareto front. These solutions are called non-dominated when optimization in any dimension negatively impacts other dimensions. Genetic algorithms are well-known for successfully optimizing multi-objective problems, and it is fairly easy to produce the Pareto front. The real challenge is to find the proper solution on the Pareto front that best satisfies the quality of service needs of the problem.

The population fitness analysis uses what is known as Pareto ranking [13]. For this, we use the concepts of inferiority and superiority. Summarizing from [13]:

Inferiority: $\bar{u}$ is said to be inferior to $\bar{v}$ *iff* $\bar{v}$ is partially less than $\bar{u}$:

$$\forall i = 1,...,n, \quad v_i \le u_i \ \wedge \ \exists i = 1,...,n : v_i < u$$

That is, if any of the *n* objectives of $\bar{v}$ is less than any objective of $\bar{u}$, $\bar{v}$ is inferior to $\bar{u}$.

Superiority: $\bar{u}$ is superior to $\bar{v}$ *iff* $\bar{v}$ is inferior to $\bar{u}$.

Pareto ranking then ranks each member of the population by the number of individuals to which the given member is superior.

The final piece of the genetic algorithm is the set of fitness, or objective, functions that are read in from an external dynamic library. These functions represent different dimensions of the QoS. When analyzing a particular waveform, the objectives must accurately represent the performance of the waveform. When the Pareto front has been optimized, the final challenge in the algorithm's performance is to make a decision about which waveform on the Pareto front best represents or satisfies the QoS needs. Some of the individuals will be better in certain dimensions than others, where the quality of service measure might value some objectives more than others such as reduced power consumption over lower BER. To accomplish this, each objective is weighted by its importance, and the weighted performance of the individuals on the final Pareto front is used to determine the selected waveform. The analysis is performed much like the Pareto ranking, but instead of being given a single point for every dominated individual, the individuals receive a weighted sum for each objective.

4. REAL-WORLD OPERATION

One of the main purposes of the design of the cognitive engine is its ability to work on real radios. To create a structure that is highly adaptable to represent different platforms, a standard input and output mechanism is defined to minimize the amount of work required to translate between platforms. The easiest way from both a human and machine-readable format is the use of XML. The work is detailed in [11] and briefly summarized here.

The chromosome structure of the GA is defined such that each gene represents at least the minimum number of states (or alleles) that a gene may have. For one million different frequency settings, the algorithm uses a 20-bit representation. The information required by the algorithm to establish this is first fed in through an XML hardware definition document that defines what knobs the radio has to turn and the range through which they can be turned.

When the GA finishes the optimization process, it must then instruct the radio platform to reconfigure itself to the new waveform. Again, the output is done using a simple, standard XML file. From here, a simple translation must occur to convert the output of the GA to the required input format that controls the radio knobs. This is often a small program: a Python parser to control the GNU Radios and a Java application that translates the genetic algorithm output XML to another XML file used to configure the IRIS radios. A high-level flow of this is shown in Figure 3.

Table 1 lists the knobs and ranges available to the cognitive engine when using the GNU Radio.

5. EXPERIMENTS

To test how the GA optimization process works, the cognitive engine was applied to the GNU Radio to produce waveforms that were simply tradeoffs between low BER and

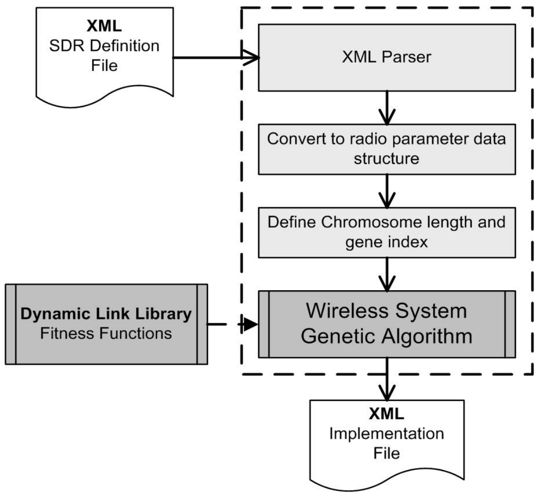


Figure 3. High level flow of genetic algorithm.

Table 1. GNU Radio Knobs and Ranges

Knob	Range	Step
Center Frequency	400 – 500 MHz	1 kHz
	2300 – 2900 MHz	1 kHz
Transmit Power	-12 – 5.5 dBm	0.5 dB
Symbol Rate	125 – 1000 kSps	125 kSps
Modulation	BPSK, QPSK, GMSK	
Pulse shape factor	0.1 – 1.0	0.01

power efficiency (modeled here as simply the transmitter power) at different objective weights. Table 2 shows how the weights balance the objectives. By setting a weight to zero, the GA ignores this objective, so the first two results show how the algorithm behaves in a single objective case. This case perfectly attains the desired goals. The multi-objective cases then show how the objectives are balanced based on their respective weights.

To add to the algorithm, additional objectives will direct the GA to solve problems for different QoS needs like data rate as shown in Table 3. The BER in both tables is arbitrarily large because of an assumed noise floor that was set larger than normal to produce clearer differences in the BER performance.

The results of these tests show the interdependency of the objectives, some of them with highly complex relationships, which will only increase as the objective functions mature and additional functions are defined. To discuss a few aspects of these results, the relationship between the spectral efficiency and BER of a waveform is interesting. As the spectral efficiency increases, the BER increases as well; a result from the GA's choice of using GMSK for higher spectral efficiency, which has a lower BER performance than either BPSK or QPSK. At the same time, the symbol rate of the resulting waveform is not at its maximum because the higher the symbol rate, the larger the bandwidth, which results in a decrease in BER. The GA tries to establish and maintain a trade-off of all the objectives while pushing in a direction set by the weights.

Understanding which weights to use for different quality of services is important and complicated. This is one more area in which the learning machine can operate together with the optimization algorithm. As observed QoS does not match desired QoS, the learning machine can adjust the objectives and weights to correct for flaws. Because of the complex interactions among the objectives, the proper weighting should be learned from trial and error in the cognitive engine.

6. FUTURE COGNITIVE RADIO RESEARCH

6.1. Implementation vs. Theory

When processing the objective functions, the cognitive engine works off known communications theory; however,

Table 2. BER (median) and Power (mean) analysis over 1000 runs of the GA

Weights	(1.0, 0)	(0, 1.0)	(1.0, 0.5)	(0.5, 1.0)	(1.0, 1.0)
BER	$1.41\text{x}10^{-7}$	$2.94\text{x}10^{-1}$	$2.54\text{x}10^{-3}$	$5.74\text{x}10^{-2}$	$2.25\text{x}10^{-2}$
Power (dBm)	2.26	-9.44	-2.70	-5.69	-4.63

Table 3. BER (median), Power, Data Rate, Spectral Efficiency (means) analysis over 1000 runs of the GA

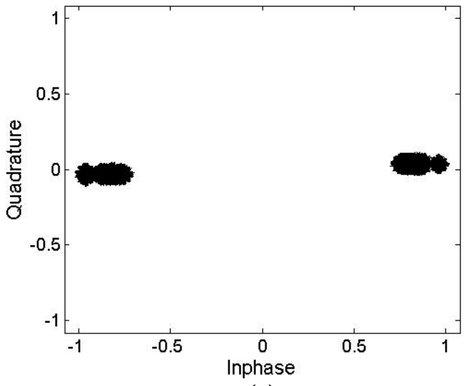
Weights	(1.0, 0.5, 0.5, 0.25)	(1.0, 0.5, 0.5, 1.0)	(1.0, 0.5, 1.0, 0.25)	(1.0, 0.5, 1.0, 1.0)
BER	$1.09\text{x}10^{-2}$	$1.58\text{x}10^{-2}$	$4.87\text{x}10^{-3}$	$1.27\text{x}10^{-2}$
Power (dBm)	-3.22	-3.67	-2.93	-3.31
Data Rate (bps)	$1.00\text{x}10^{6}$	$1.25\text{x}10^{6}$	$1.25\text{x}10^{6}$	$1.25\text{x}10^{6}$
Spec. Eff. (bps/Hz)	2.59	3.02	2.42	2.56

theory and reality do not necessarily match up for a given implementation. An example of this is the difference between Gray coded QPSK and BPSK reception with the GNU Radio using a coherent receiver. In theory, these two have the same performance, yet the actual performance for QPSK is worse than BPSK. Unfortunately, actual BER data are not available to see the difference in SNR between the two, but Figure 4 shows the problems with the constellations of BPSK and QPSK under the same symbol rates and transmit power. The FFT of QPSK in Figure 5 shows the high SNR.

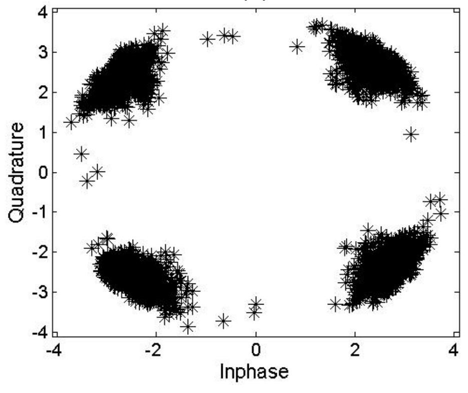
To work with differences between theory and practice, the cognitive engine must learn from its mistakes, which requires a feedback loop from the radio to the engine. When the cognitive engine implements a waveform, it does so under the assumption that the real performance of the radio matches the theoretical performance modeled by the objective functions. If the radio reports a higher BER than the objective function predicted, this information will be fed back through the optimization system to improve the modeling in the future. The cognitive engine might not understand what the problem was (maybe the gain of the Costas loop in the receiver was not set properly), only that there was a problem and to attempt to correct for it locally.

6.2. Computational Complexity vs. Performance

Aside from communications-oriented issues with SDR waveform developments, like in the previous example, computational concerns play a large role in SDR implementation. These concerns can translate directly to the cognitive engine. Currently, the GNU Radio only works with DBPSK and DQPSK, differential modulations, with a coherent receiver. For these receivers, the phase and frequency synchronization are done using a Costas loop in software. A look at the footprint of the receiver using OProfile [14] in Table 4 provides insight into the nature of the receiver by showing the percent of computational time allocated to different symbols (or functions) of the receiver.



(a)



(b)

Figure 4. Constellations for a) BPSK and b) QPSK with 125 ksps and transmit power of 0 dBm.

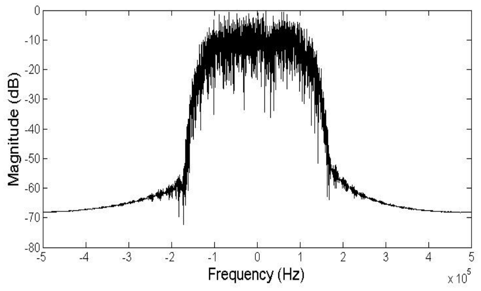


Figure 5. Normalized Frequency plot of QPSK with 125 ksps at a transmit power of 0 dBm.

Table 4. Profile of the computational time of the blocks in a GNU Radio QPSK receiver

symbol name	%
gr_costas_loop_cc	13.11
gr_constellation_decoder_cb	9.29
.loop2	5.33
gr_single_threaded_scheduler	4.74
gr_clock_recovery_mm_cc	4.25
gr_fft_filter_ccc	4.09
gr_costas_loop_cc::phase_detector_4	3.31
gr_fir_ccf_simd::filter	3.02
.loop1	2.97
gr_multiply_const_cc	2.12
gr_correlate_access_code_bb	1.37

This table represents only the top-most computationally intensive functions used by the GNU Radio library during execution, although there are many more functions involved. The most computationally intensive part is the Costas loop. Given possible constraints on power consumption of the receiver, such as in a handheld unit with a lower battery life, it could be better to implement a receiver without a Costas loop. If the receiver was changed to using non-coherent reception, it would theoretically degrade the performance by about 2 dB [15]. However, given the constraints, the transmitter could make up for this small performance degradation with a slightly higher power output. The increased BER might not be a particular problem if the application was voice with a higher tolerance for error. There is a direct trade-off here between performance and power. A suboptimal solution might have other benefits that outweigh the few dB of improvements a more complex system would provide.

System latency is another trade-off that a cognitive engine can optimize. Already, mobile phones use convolutional codes because of their lower latency over a block code with better performance. Perhaps a slight increase in performance from a feed-forward automatic gain control loop might not outweigh the slight penalty in system latency over a feed-back AGC loop.

To the cognitive engine, anything that has a trade-off in performance is a knob worth turning. As wireless devices continue to proliferate, resources will become scarcer that will result in greater benefits from an intelligent, reconfigurable radio that is capable of understanding the performance trade-offs to produce the desired QoS. A promising feature of the cognitive engine is that it demonstrates improved performance with the increase in the configurable radio parameters. As the radio systems become more complex, the cognitive engine not only scales but will produce more accurate and specific results to the problems.

7. ACKNOWLEDGEMENTS

This work was supported by the National Science Foundation under grants 9983463, DGE-9987586, and CNS-0519959 and by the National Institute of Justice, Office of Justice Programs, US Department of Justice under Award No. 2005-IJ-CX-K017. The opinions, findings, and conclusions or recommendations expressed are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or the Department of Justice.

8. REFERENCES

[1] GNU Radio, http://www.gnu.org/software/gnuradio/.

[2] P. Sutton, L. Doyle, and K. E. Nolan, "A Reconfigurable Platform for Cognitive Networks," in *IEEE Proc. CROWNCOM*, Mykonos, Greece, 2006.

[3] Ettus Research, LLC, http://www.ettus.com/.

[4] T. W. Rondeau, B. Le, C. J. Rieser, and C. W. Bostian, "Cognitive Radios with Genetic Algorithms: Intelligent Control of Software Defined Radios," in *Software Defined Radio Forum Technical Conference*, Phoenix, AZ., 2004, pp. C-3 - C-8.

[5] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, Pearson Education, Inc., Upper Saddle River, New Jersey, 2003.

[6] I. Gilboa and D. Schmeidler, *A Theory of Case-Based Decisions*, Cambridge University Press, Cambridge, 2001.

[7] J. Kolodner, *Case-Based Reasoning*, Morgan Kaufmann Pub., San Mateo, CA, 1993.

[8] B. Fette, *Cognitive Radio Technology*, Elsevier, New York, 2006.

[9] T. W. Rondeau, B. Le, D. Maldonado, D. Scaperoth, and C. W. Bostian, "Cognitive Radio Formulation and Implementation," in *IEEE Proc. CROWNCOM*, Mykonos, Greece, 2006, pp. 1-10.

[10] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, MA, 1989.

[11] D. Scaperoth, B. Le, T. W. Rondeau, D. Maldonado, C. W. Bostian, and S. Harrison, "Cognitive Radio Platform Development for Interoperability," in *MILCOM*, Washington, D.C., 2006.

[12] C. Hwang and A. Syeed, *Multiple Objective Decision Making - Methods and Applications*, Springer-Verlag, New York, 1979.

[13] C. M. Fonseca and P. J. Fleming, "Genetic Algorithms for multiobjective optimization: formulation, discussion, and generalization," in *Proc. Int. Conf. Genetic Algorithms*, 1993, pp. 416 – 423.

[14] OProfile, http://oprofile.sourceforge.net/news/.

[15] J. G. Proakis, *Digital Communications*, McGraw Hill, New York, 2000.