Hindawi Publishing Corporation
Journal of Computer Networks and Communications
Volume 2012, Article ID 549106, 15 pages
doi:10.1155/2012/549106

Research Article

Hybrid Experiential-Heuristic Cognitive Radio Engine Architecture and Implementation

Ashwin Amanna, Daniel Ali, David Gonzalez Fitch, and Jeffrey H. Reed

Bradley Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA 24061, USA

Correspondence should be addressed to Ashwin Amanna, aamanna@vt.edu

Received 20 November 2011; Accepted 23 January 2012

Academic Editor: Luca Ronga

The concept of cognitive radio (CR) focuses on devices that can sense their environment, adapt configuration parameters, and learn from past behaviors. Architectures tend towards simplified decision-making algorithms inspired by human cognition. Initial works defined cognitive engines (CEs) founded on heuristics, such as genetic algorithms (GAs), and case-based reasoning (CBR) experiential learning algorithms. This hybrid architecture enables both long-term learning, faster decisions based on past experience, and capability to still adapt to new environments. This paper details an autonomous implementation of a hybrid CBR-GA CE architecture on a universal serial radio peripheral (USRP) software-defined radio focused on link adaptation. Details include overall process flow, case base structure/retrieval method, estimation approach within the GA, and hardware-software lessons learned. Unique solutions to realizing the concept include mechanisms for combining vector distance and past fitness into an aggregate quantification of similarity. Over-the-air performance under several interference conditions is measured using signal-to-noise ratio, packet error rate, spectral efficiency, and throughput as observable metrics. Results indicate that the CE is successfully able to autonomously change transmit power, modulation/coding, and packet size to maintain the link while a non-cognitive approach loses connectivity. Solutions to existing shortcomings are proposed for improving case-base searching and performance estimation methods.

1. Introduction

Wireless communication devices and networks face outside influences that degrade performance and have potential to render links useless. New advances in the area of cognitive radio (CR), inspired by artificial intelligence integration with reconfigurable platforms, enable devices and networks to observe, make a decision and learn from past experience. Key problems faced by CR are how to effectively integrate both learning and decision onto software-defined radio (SDR) over-the-air platforms such that they can react to situations quickly and effectively.

Specifically this paper address the realization and implementation of a cognitive engine (CE) on an SDR platform for the purpose of link adaptation. The problems addressed include incorporating mechanisms for system observation, triggering the engagement of a CE, architecting the CE such that it can both make decision when faced with new situations, and learn from past experience.

Prior art has defined CE architectures based on heuristic decision making, such as GA, as well as experiential CBR. Earlier works have also proposed hybrid architectures that combine both. This paper builds upon previous work to fully implement a hybrid CBR-GA engine such that the CBR feeds into a GA. The CBR makes decisions if past experiences are available, otherwise the GA is entrusted to identify radio parameter settings if the situation is not similar enough to past experiences. The top cases of the CBR feed into the GA as initial parents to improve the starting search points of the GA. Other contributions include overall process flow, case base structure/retrieval method, performance estimation approach within the GA, and hardware/software lessons learned.

Unique solutions to realizing the concept include mechanisms for combining vector distance and past fitness into an aggregate quantification of similarity. Current limitations within the architecture and implementation are discussed.

Solutions are proposed for case base searching based on a unique indexing scheme. Additional shortcomings in estimation methods are discussed with proposed solutions utilizing blind channel estimator feedback to the transmitter.

The remainder of this paper is structured as follows. Section 2 introduces the foundations of GA and CBR cognitive radio architectures. Note that our use of the term *cognitive* is trivial at best given the transformational work in the fields of psychology, education, and computer science studying true cognitive science. This section focuses on cognitively-inspired simple heuristic and experiential decision-making algorithms for integration with an SDR platform. We will not discuss rule, policy, or bayesian reasoners. Section 3 details our CE architecture and process flow. This includes the CBR case structure, retrieval mechanisms, and GA integration and process flow. Section 5 discusses the specific hardware platform and detailed integration issues related with coordinating a software-based CE with a stand-alone SDR. Section 6 presents performance metrics as the system places a load of an over-the-air file transfer across a transmitter/receiver link in the presence of a third party interference source. Three specific environmental cases were considered. Section 7 identifies specific limitations and assumptions in our current architecture related to CBR search and retrieval and current estimation algorithms within the GA. Unique and compelling solutions are proposed for each to attack the limitations. Finally, Section 8 summarizes and discusses areas for future research.

2. Background

This section focuses on background related to CR architectures with focus on GA and CBR-based architectures. CR's origin stem from the growth of reconfigurable SDR platforms. This transition from purely hardware-based platforms was the enabling factor for enabling the integration of artificial intelligence to wireless communications. Mitola is widely credited for jump starting the field with the original thesis of a system that can intelligently respond to the needs of a user [2]. Today's CE architectures maintain this vision with designs oriented around observing system performance metrics, also known as meters, in order to make decisions about how to set configuration parameters of the SDR to support a defined goal.

The foundation of the process loop that our CE follows lies in the observe, orient, decide, act (OODA) loop [3]. The meters provide the trigger for engaging a CE and define improvements. Radio goals, such as minimizing energy, minimizing error, or maximizing throughput provide an orientation that the CE uses to drive decision making. Simplified decision algorithms identify new radio configuration parameters that ideally will improve performance such that the observable meters fall within a defined threshold. Finally, these new radio parameters are passed to the SDR for implementation and return to the start of the loop. Note that this process flow is reactionary in nature and sometimes considered flawed due to its inability to be proactive.

Early CE architectures focused on rule-based decisions providing fast reaction time yet would operate the same way regardless of the situation. Biologically inspired heuristic methods were seen as an alternative approach that enabled multiobjective performance definitions and capability to react to new situations [4]. GAs take an initial radio configuration and evolve it through an iterative combination of random mutations and crossover of features of strong candidate solutions. Disadvantages of the method include time required to reach convergence, strong dependency on the definitions of fitness, and requirement for estimation algorithms to rank viability of candidate solutions. The engine described here, limits the number of generations the GA is allowed in order identify a potential solution before conditions have changed too much. It is acknowledged that the solution from the GA, especially one from generation-limited operations, will not be optimal.

Implementation of a GA requires conversion of metrics and radio configurations onto a unified scale in order to combine them into a fitness function for assessing a solution's potential for success. This requires the use of utility functions that normalize metrics onto similar scale [5].

While GAs enabled a CE to adapt to new situations, the convergence time diminished its value in practical deployment given how fast wireless environments change. To address this shortcoming, researchers investigated combining a GA with other decision architectures. CBR possessed many of the desirable characteristics for CR. The fundamental algorithm is computationally straightforward, can make decisions faster than a GA and incorporates long-term learning. This experiential-based decision maker founds itself on the human decision-making principles that a solution to a problem can be found by utilizing or modifying solutions to past problems that are close in similarity [6]. While simple conceptually, the implementation requires mechanisms for case definitions, quantification of similarity, case search, and case addition. Hybrid combinations of CBR and heuristics were designed for use with 802.22 white space transmissions where the most similar cases from a library were then optimized using a hill climbing search and a GA [1, 7]. This engine uses the configuration of the most similar retrieved case if it falls within a defined similarity threshold, otherwise the top retrieved cases are used as part of the initial parent population of the GA.

3. Hybrid Architecture

3.1. General Process Flow. Figure 1 illustrates the general decision cycle of the cognitive architecture. The process flow includes five parts: (1) observation, (2) orientation, (3) decision, (4) action, and (5) learning. The first part of the decision cycle focuses on system observations. Typically, observations include meters and environmental conditions, such as noise level. This is implemented by feeding back collected meters from the receiver to the transmitter via a dedicated 802.11 link. Predefined thresholds for the performance determine when the engine moves out of the observation cycle into the orientation and decision cycles. Currently, the CE is triggered when the packet error rate (PER) goes over a certain threshold.

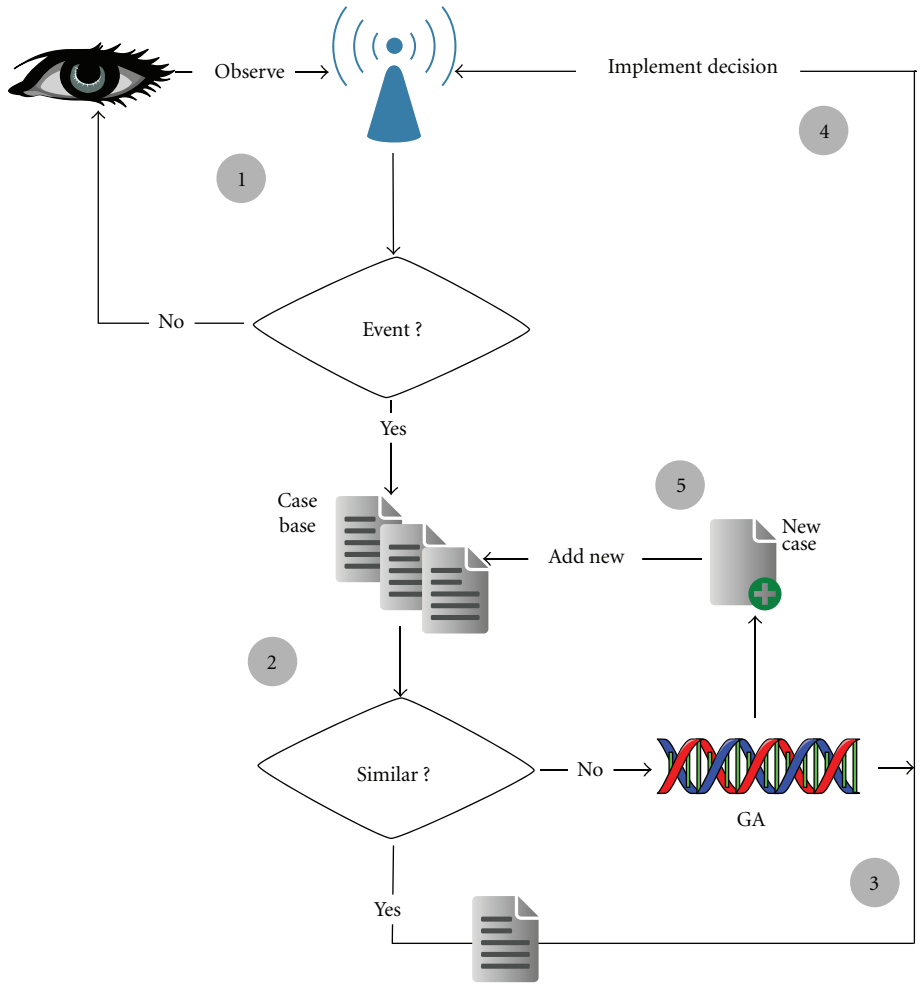

Figure 1: Hybrid CBR-GA cognitive engine process flow.

If an event has been detected, the engine moves onto an orientation phase where it determines which decision-making module best suits the situation. This engine is based on a hybrid architecture of a CBR and a GA. The goal is to first identify if a past decision is similar to the current situation. If similarity falls within a defined threshold, then the configuration settings from the most similar past case are selected and implemented on the radio. This is preferred, as the time-to-action for a CBR is typically much faster than the heuristic GA. However, if there is no past experience that falls within a defined similarity threshold, then the GA is engaged.

The GA is a heuristic search optimization algorithm described in Section 3.2. Once it identifies a solution, the decision is implemented by changing the configuration inputs of the SDR to match the solution. The last part of the decision cycle involves learning whether or not this solution is worth remembering later. This is accomplished by checking the system metrics for improvement after implementing the solution. If this new solution resulted in improved performance, then solution is added as a new case to the library. Details on the GA process are discussed next, followed by the CBR.

3.2. GA. GAs fall underneath the classification of heuristic search. This section briefly reviews their operation. The reader is referred to [8] for more details. Inspiration for the GA came from the evolutionary processes of organisms. Here, organisms or animals that possess genetic traits that are more favorable for a given situation have a higher chance for survival. These traits are passed on from parents to children with the goal of passing on the best traits from each parent. Furthermore, random mutation provided the capability to adapt over time to new situations. Adapting these concepts to cognitive radio requires defining performance goals into computational functions and encoding radio configuration parameters into a string. Fitness functions quantify the success of specific set of configuration parameters. For example, a function that was a factor of packet error, throughput, and spectral efficiency could define a radio's performance. A second set of functions, called utility functions, is implemented to normalize each performance measure onto similar scales.

These functions provide a computational equivalent to an organism's chances of survival. The combination of configuration parameters that leads to the largest value of

the fitness functions is most desired. Similarly, encoding the parameters into a form equivalent to a cellular chromosome enables manipulation using genetic concepts such as gene crossover and mutation. The iterative operation of the GA starts by identifying an initial population. Each individual in the population is a unique set of encoded configuration parameters that factor into an individual's fitness function. The weakest individuals are removed and the remaining individuals are used as parents for a new population. The individuals in this next generation are created by combining, or crossing over, the best traits from a set of parents. Additionally, during each generation, random mutations occur in some individuals. This process is repeated until a predefined maximum number of generations is replaced. Overall, the key configuration parameters of a GA that drive its operation are population size, cross over rate, mutation rate, and maximum generations.

The GA eventually converges towards a good, though not necessarily optimal solution. Like any random-based search, the initial seeds, or starting point of the search are a key aspect of reducing convergence time. The architecture presented here attempts to provide stronger starting points for the GA by using entries in the case history as initial seeds.

3.2.1. Utility Functions and Fitness. Utilities are metrics that give the system a way to measure the desirability of a configuration parameter or meter. They are normalized between 0 and 1, where 1 indicates most desirable and 0 least desirable. Each utility function will either be monotonically increasing or decreasing depending on whether the desirability of the parameter follows a higher-is-better or a lower-is-better goal. The general equation that was used to generate the utility functions are based on previous work [9]. The plots of each utility function can be seen in Figure 2, where the x-axis represents the range of the configuration parameter and meters, and the y-axis is the utility value. Note that for transmit power, on the x-axis, the range goes from 0 to 100. This is because here transmit power is represented as a percentage, where 0 corresponds to minimum output power and 100 represents maximum output power.

Fitness is a metric that combines all utilities into one number. It is also normalized between 0 and 1. Several different methods for calculating fitness were considered:

$$F = \frac{\sum(u_i \cdot w_i)}{N}, \tag{1}$$

$$F = \sqrt[N]{\prod(u_i^{w_i})}, \tag{2}$$

$$F = \frac{\sum(u_i^{w_i})}{N}. \tag{3}$$

Equation (1) shows the fitness calculations through a weighted sum, (2) shows the calculation through a weighted product, and (3) shows a summation of utilities raised to their weights, where u_i are the utilities, w_i are the weights associated to each individual utility and N is the total number of configuration parameters and meters. The purpose of the weights is to give a different significance to each utility for the overall fitness calculation, defining what tradeoffs make when deciding what configuration parameters to select.

Out of the three possibilities, the product fitness was used, because the value of the fitness will only be desirable (close to one) if all the utilities are desirable, and if one utility value is undesirable (close to zero) then the overall fitness will also be undesirable. Figure 3 shows an example of what the fitness space would be for two utilities, one with a weight of 0.8 and the other with a weight of 0.2, for each of the three methods. The product fitness approach is the only one that fulfills the goals. This means that if a configuration parameter or meter is especially poor, then it will get strongly penalized in the fitness calculation; so only a solution that has good combination of configuration parameters and meters will have a good fitness value.

3.2.2. GA Process Flow. At a high level view, the basic GA process flow is described as follows.

(1) Each row of the population is a chromosome defined as a combinations of potential genes. Each available gene corresponds to a unique combination of configuration parameter values that the radio could be set to. Typically each gene is encoded into a defined number of bits.

(2) The initial population is generated by seeding one-third of it with the most similar cases in the CBR if list if it has more than two cases stored; else it seeds it with the current case. The other two-thirds are randomly generated.

(3) Then, for each generation, the following process is followed, until a new population is formed:

  (a) the utilities and fitness are calculated for each individual in the population,

  (b) to generate the next population, parent chromosomes are randomly selected, where the ones with a higher fitness are more likely to be chosen, and they are crossed over,

  (c) Then each bit a gene is randomly mutated with a certain fixed probability.

(4) This process is repeated for a fixed number of generations.

3.2.3. Estimation Methods. For the GA to estimate the meters, the following process is carried out.

(1) First, the new SNR is estimated from the current SNR, the current transmit power, and the new transmit power, where the new SNR is the current SNR less the current transmission power added to the new transmission power in dB.

(2) To calculate throughput, we know that the system uses a constant symbol rate (R_S). Therefore, we can calculate the new throughput ($R_{U_{\text{New}}}$) from the symbol rate, the new t-error correcting code (n_{New}, k_{New}), where t is the number of bits it can correct, modulation order (MO_{New}) and payload length (PL_{New}),

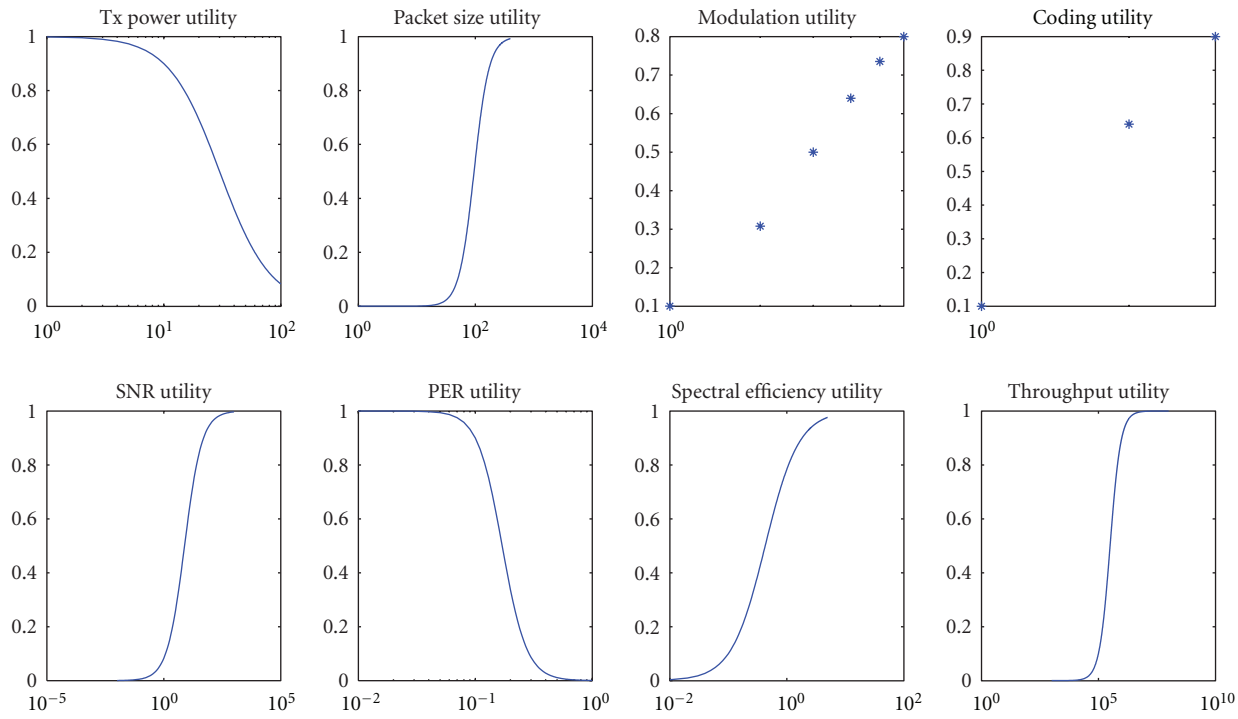

FIGURE 2: GA Utility functions— utility functions map the configuration parameters and meters onto a common scale between $(0, 1)$. A value of 0 is considered undesirable, while a value of 1 is considered most desirable. These utilities are adapted from previous work [1].

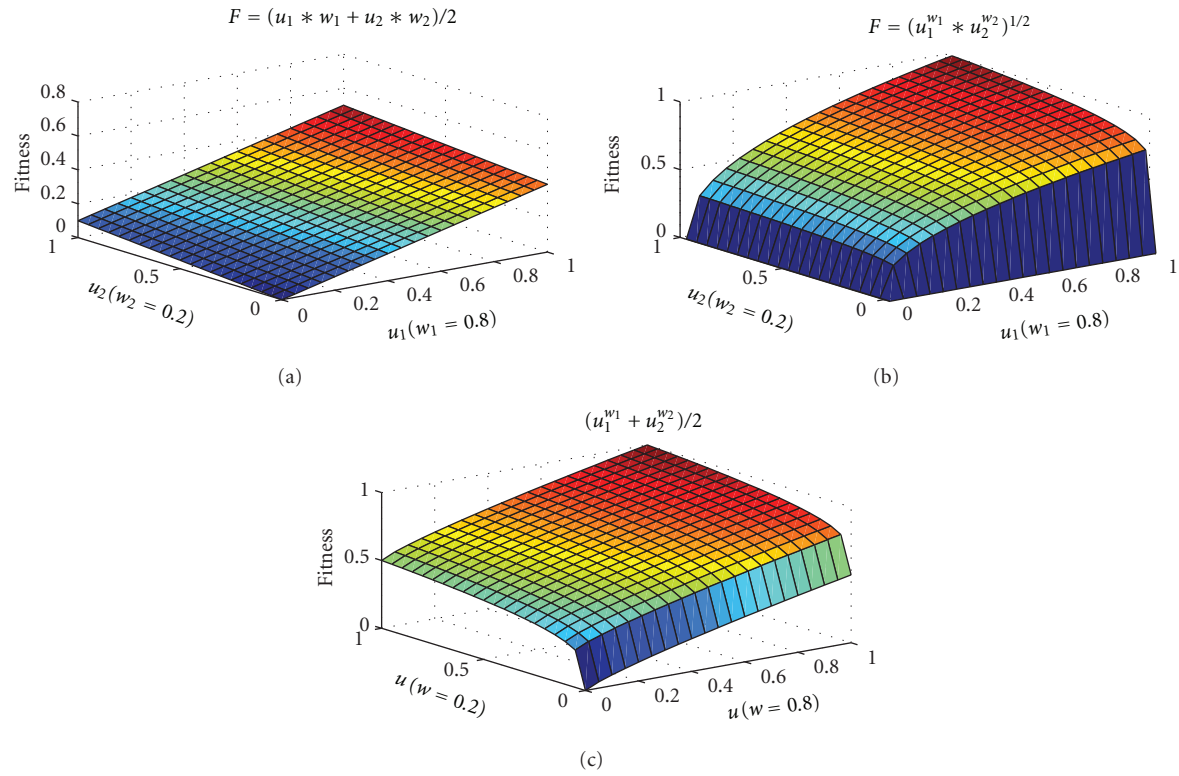

FIGURE 3: Fitness space— three methods for calculating overall fitness from utilities and weights were considered. These figures show the fitness with $w_1 = 0.8$ and $w_2 = 0.2$. The goal is for fitness to only be close to one, or highly desirable, when all of the utilities are also closer to 1. If anyone utility value's desirability drops, then the overall fitness value should drop as well. The product method met these criteria best.

the header length (HL), the number of subcarriers (NC), and the length of the cyclic prefix (CP). This is shown in (4):

$$R_{U_{\text{New}}} = R_S \cdot \frac{k_{\text{New}}}{n_{\text{New}}} \cdot \frac{\text{PL}_{\text{New}}}{\text{PL}_{\text{New}} + HL} \cdot \frac{\text{NC}}{\text{NC} + \text{CP}} \cdot \text{MO}_{\text{New}}. \quad (4)$$

(3) To calculate the BER for a t-error correcting code (n, k), refer to [10]. Assuming that the throughput for both the coded and uncoded data stream is the same, the relationship between the probability of error between the coded and uncoded data is shown in (5):

$$P_{e_{\text{Coded}}} = \binom{n-1}{t} \left[P_{e_{\text{Uncoded}}} \left(\frac{k}{n} \cdot \frac{E_b}{N_0} \right) \right]^{t+1}. \quad (5)$$

However, for the system's case, what remains constant is the symbol rate and not the throughput. Therefore, for coded data, the throughput (R_U) is calculated from the symbol rate (R_S) as described in step (3). Then a tilda symbol rate (R'_S) is calculated as if this throughput were that of uncoded data.

(4) Now we can calculate E_b/N_0 from the new SNR, the tilda symbol rate (R'_S), the new modulation order (M) and the bandwidth (BW) as shown in (6):

$$E_b/N_0 = \text{SNR}_N + 10 \log_{10} \left(\frac{\text{BW}}{R'_S \cdot \log_2(M)} \right). \quad (6)$$

(5) To estimate the new BER, the system uses the formulas from [16] for Additive White Gaussian Noise (AWGN) channels.

(6) The PER can be estimated from the BER and the PL and HL (in bytes) through (7):

$$\text{PER} = 1 - (1 - \text{BER})^{8(\text{PL}+\text{HL})}. \quad (7)$$

(7) Finally, the new spectral efficiency is estimated by simply dividing the estimated throughput by the bandwidth.

3.3. CBR. The CBR keeps a list where it stores previous situations encountered in the past and what solution it came up with for that particular situation. A vector distance is calculated that quantifies how close the current situation is to the past situations within the case-base. We define similarity as a combination of this distance and the effectiveness of the solution, known as fitness, when it was originally applied. The cases are ranked by similarity and are filtered by a defined similarity threshold. The top case that meets this threshold is applied to the radio front end. If none falls in this threshold, then the top cases are seeded into the GA as initial parents as described in Section 3.2.

3.3.1. Case Structure. For each case, the CBR stores the following.

(i) ID: Memory is preallocated for the CBR list. Each case is initialized to dummy values. The ID indicates if valid values are stored in that entry: 0 for when nothing is stored there and 1 for when a there is a case.

(ii) Old configuration parameters store the values of the parameters before they were changed.

(iii) Old meters store the values of the meters that triggered the CE.

(iv) New configuration parameters store the value of the parameters after they were changed.

(v) Utility stores the new utilities and fitness after the configuration parameters were changed and the meters measured.

3.3.2. Retrieval. To decide whether or not a previous case should be used for the current situation, two metrics are calculated: distance and similarity. In general, a case from the library can be considered a certain vector distance away from the current situation. While this alone is sufficient to identify past experience that could provide a solution to the current situation, more resolution in the retrieval is required.

(a) Distance is a metric that shows how close or far a current situation is from the old configuration parameters and meters stored in the CBR list. This value is normalized between 0 and 1, where 0 represents two identical cases, and 1 represents completely opposite cases (if the values of the configuration parameters and meters of the current case and of the stored case are at opposite sides of the range values). To calculate this metric, the euclidean distance was used in (8):

$$d = \sqrt{\frac{\sum \left((\text{CurCase}_i - CB_i) / (\text{Range e}_{\max_i} - \text{Range e}_{\min_i}) \right)^2}{N}}, \quad (8)$$

where CurCase_i is the value of the configuration parameters and meters of the current case, CB_i is the value of the old configuration parameters and old meters stored in the CBR list, $\text{Range e}_{\max_i}$ and $\text{Range e}_{\min_i}$ are the maximum and minimum values of each configuration parameter and meter, respectively, and N is the total number of configuration parameters and meters.

(b) Similarity is a metric that combines distance (how close the current case is to the stored ones) and new fitness (how good the stored solution is). Several different approaches for this combination were considered where smaller distances and higher fitnesses led to higher similarities. Equation (9) shows a product of distance and fitness, (10) shows the distance raised to the fitness, (11) shows the fitness raised to the distance, and (12) shows the inverse distance raised to the fitness

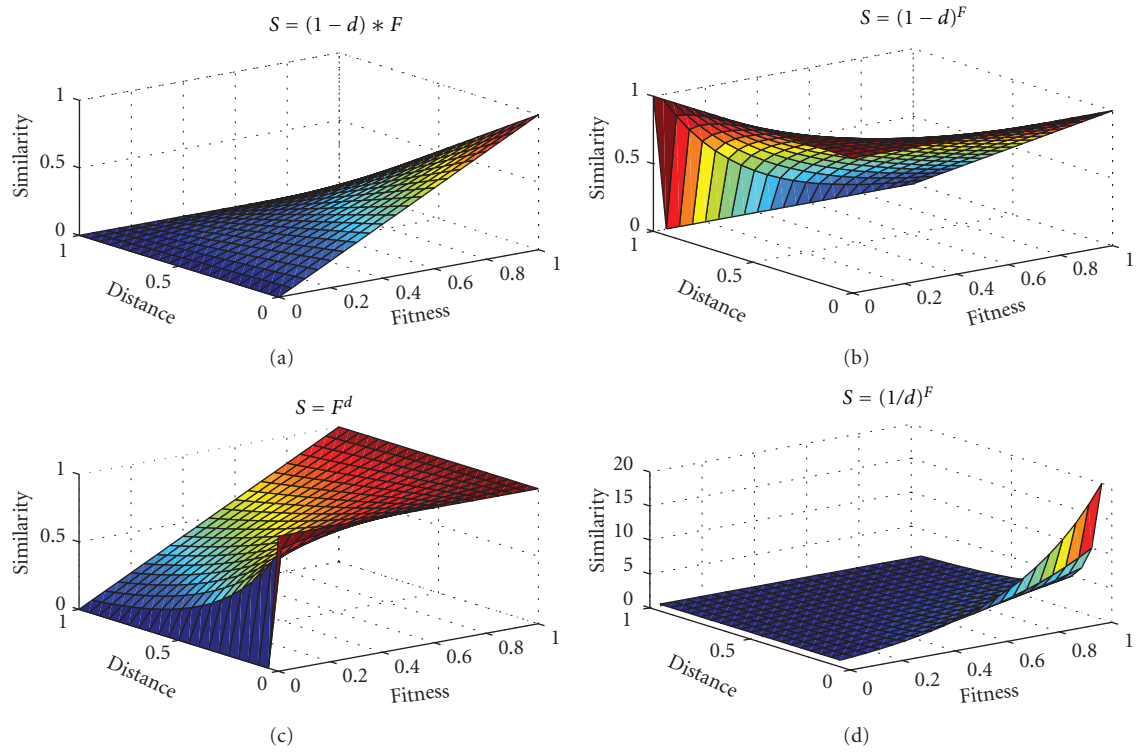

Figure 4: Similarity space— the similarity spaces of the four approaches of combining distance and fitness are shown. The goal is to achieve a high similarity when distance is small and fitness is high. Currently, the engine implements $S = (1 - d) * F$.

$$S = (1 - d) \cdot F, \tag{9}$$

$$S = (1 - d)^F, \tag{10}$$

$$S = F^d, \tag{11}$$

$$S = \left(\frac{1}{d}\right)^F, \tag{12}$$

where d and F are the distance and fitness metrics, respectively. In Figure 4 a comparison of the four possible similarity spaces are compared. Similarity (10) and (11) were discarded because the similarity metric should only have a high value when fitness is high and distance is low. Equation (12) was not used because it would present problems when distance was zero. It was also more complicated to normalize. Therefore, (9) was finally used for calculating similarity.

4. Configuration Parameters of the Cognitive Engine

The CE requires initialization of several elements before operations can begin. The GA requires definition of crossover rate, mutation rate, population size, and maximum generations. Similarly, the CBR requires definition of the the maximum case base size and similarity threshold for case retrieval. Finally, the defined PER threshold that will trigger engine engagement as well as the weightings of the configuration parameters used in calculating fitness are required. These are described below and listed in Table 1.

Table 1: Default configuration parameters of the CE.

Type	Parameter	Value
GA	Xover rate	0.7
	Mutation rate	0.01
	Population size	100
	Max generations	25
CBR	Case size	100
	Similarity threshold	100
Configuration parameter weights	PER	0.1
	Tx power	0.7
	Packet size	0.3
	Modulation	0.4
	Coding	0.5
Meter weights	SNR	0.5
	PER	0.9
	Spectral efficiency	0.3
	Throughput	0.3

(i) GA crossover rate: it is the probability that two parents, after being selected from the population in the GA, will crossover.

(ii) GA Mutation Rate: It is the probability that each bit of the GA population will get modified (changed from one to zero or from zero to one).

(iii) GA population size: it is the amount of chromosomes that the GA has in each generation.

(iv) GA Max generations: it is the maximum amount of iterations that the GA is allowed to make before coming up with a solution.

(v) Case-base size: its is the maximum amount of cases that will be stored.

(vi) Similarity threshold: it is a metric for the CE to decide whether to use the GA or the CBR. If the current case's similarity with any of the ones stored in the CBR is higher than this threshold, then it uses the CBR, and if it is lower, then it will use the GA.

(vii) Current PER threshold: if the measured PER is over this threshold, then the CE will get called.

(viii) Configuration parameter and meter weights: they are used for the calculation of fitness. The relative value of each weight with respect to the others will give each parameter more or less significance in the calculation of the value of fitness. This allows the GA to know how to compromise configuration parameter and meter values to try to find an optimum solution based on the goals.

5. System Model Implementation

The system is designed around the transfer of binary picture data from one node to the other under different environment scenarios. In order to accomplish the data transfer, a Universal Software Peripheral Device (USRP) was used in combination with the *liquid-DSP* software suite for software-based DSP [9]. The USRP is a flexible and affordable SDR platform often used in academic research. To extend this functionality, we implemented a feedback network on which to send control messages which collects real-time data. This section will describe, at a high level, the elements involved in the design of the parameter changes, the feedback collection, as well as certain design tradeoffs that must be made.

5.1. RF Hardware and SDR Interfacing. Ettus Research [11] is well known for producing the USRP which provides a flexible RF hardware front end on which to send and receive data. Among the different types of USRPs, there exists a distinction in the generations of these devices, each with a different bus speed for user data processing. For our system we chose to use the USRP 1 generation which is defined by the types of daughter cards supported along with its universal serial bus (USB) interface. Interfacing with the USB is a challenge and mechanisms for addressing this are discussed later. The USRP contains a single-core processor, however two threads from a multicore controlling laptop can share and interact with it. There is overhead for switching the USRP between threads as well as synchronization issues to consider. For this application, there is one dedicated thread from each USRP node and its accompanying laptop. In addition, there are other threads from the master controlling laptop that performance collection of meters, issues commands to start the receiver, and turns on or off the interference.

On top of the USRP board itself, a daughter card must be used to specifically define the RF parameters that it supports. Different daughter cards support different frequencies and bandwidths. For our system we chose the WRX daughter cards to support frequencies in the unlicensed TV bands. These higher frequencies are more susceptible to environmental conditions which provides a rich testing environment.

When implementing a CR system, one is limited by the amount of parameters supported by the DSP library. We feel that *liquid-DSP* [9] meets the requirements and is confirmed by [12] that it provides the most effective aspects of a communications system including a wide spectrum in each parameter. *liquid-DSP* provides many functionalities, however, we chose a subset of these to trade-off reducing the selection complexity of our algorithms while supplying a diverse solution set.

When our engine is used in combination with a lightweight DSP library, we are able to focus on accurate metric collection of the system. To facilitate ease of implementation, we used an 802.11 feedback network which is usually supported through an internal wireless network interface controller (NIC) in most recent laptops. By threading the processes of the radio applications, we are able to have the radio controller perform two essential tasks; data processing for the radio front end, as well as message parsing and taking the appropriate action. One thread is responsible for computing data to either go to or come from the USRP, while the other processes control messages to carry out the desired action. The most used commands for this system are meter collection, start-up and shut-down commands, parameter reconfiguration, and environmental control among others. Passing these messages around a light weight, custom, brokered architecture provides a reliable base for data collection, both on- and offline, in a multinode ad hoc network.

5.2. Software Control and Signal Processing. To interface with these messages at high level, a command structure was built around MathWork's MATLAB program. By defining MATLAB compatible functions, we make use of the lower level, C++ control code. Because C++ code cannot be used naturally by MATLAB, the use of the *MEX* interfaces was used to compile the system code into MATLAB usable functions. This way, complex matrix processing can be done efficiently in MATLAB, while low level socket code is still used to send and receive messages between separate programs. Different communication methods exist that send information between programs, such as file sharing, signals, and sockets. Socket communication was chosen for the ease of designing a single socket to be used in multiple instances, where the only redefinition needed would be changing the address and port number of the listening process. The network uses an ad hoc network, thus all nodes on the network have the potential to send information to one another, however, any given node only needs to send their messages to the *Broker*. This also provides the same interface for programs that are on the same computer, such as the transmitting program and the MATLAB control process.

The *Broker*'s main job is to accept messages from all nodes on the network and distribute them accordingly. This eliminates the need for an address included in every message as there are a relatively small number of messages that need to be sent throughout the system. Once these messages are initially defined, there is little need for extension, thus a more static approach to message passing. Using the *Broker* object, however, always for an easily extended network to add more nodes for further testing. Through the MATLAB/*MEX* interface, C++ code was used to interact with the *Broker* object as the CE. This is similar to other architectures such as CORBA, or the CROSS platform [13]. By using a properly layered and scalable control structure to handle back-end data processing, we are able to develop a high-level testing framework contained within MATLAB.

The initialization parameters for communications hardware are usually predefined and must be negotiated before the actual data transfer takes place. *liquid-DSP* offers support for real-time parameter augmentation in user payloads allowing us to focus on the CE in terms of optimizing data throughput at any point during the data transfer (bounded by at least every packet). This is possible by having a relatively short, robustly coded, and modulated header that contains the necessary information to demodulate and decode the user payload. Our system uses OFDM to transfer data which also incurs the need to consider the number of subcarriers and their frequency bandwidth allocations as listed in Table 5. This system is used instead of a single carrier signal due to its wide applicability in real world applications, such as long-term evolution as well as reporting more accurate network metrics for the type of noise we inject into the environment.

5.2.1. Observing the Meters. The meters provide the system with a means of classifying the transmitter's performance throughout the data transfer. By choosing appropriate meters we can aptly describe our system at any given point in time. The engine itself considers the following meters at the receiver node: received signal strength indicator (RSSI), signal-to-noise ratio (SNR), windowed packet error rate (PER), throughput, and spectral efficiency. Here, throughput and spectral efficiency are measured at the receiver and are used in contrast to the PER to measure performance.

5.2.2. Configurable Parameters. Based on each decision the engine makes, the engine needed an avenue to push new parameter settings to the radio front end. By adding the CE as a node on the ad hoc control network, we are able to effectively let the MATLAB engine communicate to the radios for metric collection, parameter adjustment, and other control messages. The engine considers parameters and their associated values listed in Table 2. We present just the minimum and maximum values here due to the discrete and static search space allowed in the system. Packet size and power are a continuous value within these ranges, while modulation and coding are discrete values. This distinction must be accounted for in the selection algorithms used, as our model has the possibility of coming up with a floating-point value for the discrete parameters. By properly segmenting a continuous search space, we can represent these results in a way that is transparent to mechanisms that rely on either a continuous (such as GA processing) or discrete (actually application of a new scheme to the radio) parameter.

Table 2: Ranges of reconfigurable SDR parameters.

Adjustable parameter	Min	Max
Transmission power	−20	0
Modulation	1 bps	6 bps
Coding	None	Reed-Solomon
Packet size	20	340

5.3. CE Triggers. The CE uses raw transmission data and collected metric data in a utilitarian style of abstraction. By considering the utility (i.e., usefulness) of the metric or configuration parameter value when making decisions we are able to consider not the raw value presented, but the usefulness of that value to the system. By measuring utility instead of the raw value themselves, we are optimizing each parameter's usefulness within the system, allowing for a more natural style of comparison. The engine, as of now, triggers on a predefined threshold of metric value, namely PER. For example, we implemented one trigger as a rise in PER above a rate of 10%. This can be changed to use its utility instead of a raw value to better represent the crossing of PER into an unusable area of operation. This is important to CE design and should be accounted for when considering case-base usage [9].

5.4. Metric Considerations. When collecting performance metrics, the system expects the data to reflect the current status of the transmitter's effectiveness for transmitting a file. In order to present a more instantaneous snapshot of the current situation, PER is presented on a 50 packet window. That is, for the last 50 received packets, the engine records the number of packets that failed the cyclic redundancy check (CRC). Given that a payload has failed this check, we can effectively discard the packet as incorrect data, recording the error. This is in contrast to the measured data rate, which is constantly averaged. Here, we allow the data rate to be averaged at the receiver as a measure of how effectively the transmitter is sending data. Thus, if the transmitter uses a high modulation, this will be reflected in the calculated data rate, however, if the power is too low, or it is simply too noisy an environment, then the PER should also reflect this in stark contrast to the amount of data being pushed through.

Measuring on an overall average, as well as a shorter window of considered metric samples has pros and cons as discussed above and are provided in a mix here for those reasons. Depending on the type of testing pursued, different methods for collecting performance meters are required. For example, a probing technique inspired by traditional design of experiments methodology required sending a specific number of packets repeatedly [14]. This is difficult in the engine's implementation due to the decoupling of the metric requests by the engine, and the amount of data, that is, an unspecified number of packets. By measuring the data

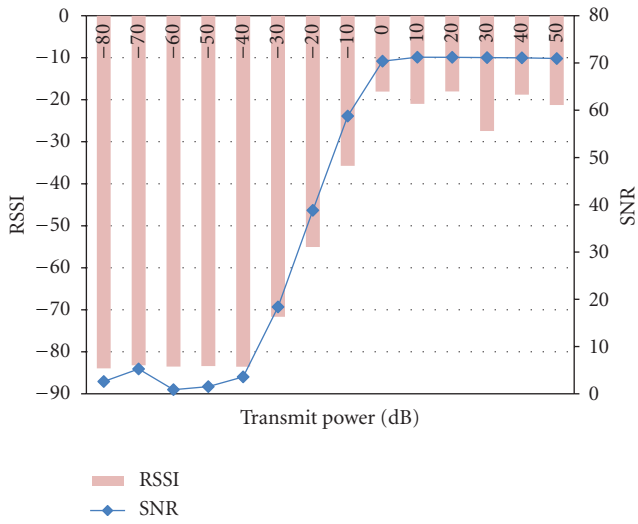

Figure 5: RSSI and SNR as the software gain is varied. RSSI is on the left axis and SNR is on the right axis. This illustrates the usable range of the transmit power parameter as between −40 dB and 0 dB.

sent over a specified number of packets on a whole, we can accurately provide results to the profiling mechanism. This is not intuitive for an instantaneous observation at any given time in the transmission in the case of the engine.

5.4.1. Configurable Parameter Considerations. Calibration of transmit power range is an important step prior to deployment. While the USRP allows one to set transmit power between −80 dB and 50 dB the radio does not operate well across this entire range. We need to define a more usable range that the CE can set the power to. A test was performed where transmit power gain was set and RSSI and SNR were measured during a data file transfer.

Because this range may be affected by many different things, including the environment itself, we thought it would be useful to test the effective range of powers available to the system within our context of a no-noise environment. The results, shown in Figure 5, illustrate that usable range is between −40 dB and 0 dB.

Note that there are actually two transmitter gains settable in the USRP. Typically one considers the software transmit gain setting as a tunable parameter of the USRP. There is also a hardware gain setting implemented by the Universal Hardware Drivers (UHD) alongside of the liquid interfaces. Due to the higher difficulty in setting the UHD gain setting, it is not a parameter that the CE has access to. It is set to a high level and all transmit power change are performed through the *liquid*-DSP interface.

Currently the engine selects from six defined modulation and three coding schemes. Packet size and transmit power have a much finer resolution. Overall, there are 56 potentially available modulation schemes in the *liquid* library. This limitation placed on available modulations was driven by the GA's need use of estimation models. The models used in the GA to estimate the fitness of potential solutions needed to account for all the available modulations. Therefore, to reduce complexity a select few modulations were used for each given bit per symbol range. These ranges are displayed in Table 3. A large number of coding schemes are also presented by *liquid*, but are not all used within the system. Two different codings are used for our GA's model, thus the system is limited to Hamming coding and no coding at all. Packet size is variable within the system to a predefined limit. Packet size is defined by header size and payload length. The payload length is user definable. We keep the size of the packets at least as long as the encoded header information, which is 33 bits and within the range of average packet size which usually does not exceed more than a few hundred bytes.

Table 3: Modulation schemes used for each bit per symbol setting.

Modulation depth (bps)	Scheme used
1	BPSK
2	QPSK
3	8-PSK
4	16-QAM
5	32-SQAM
6	64-QAM

5.5. Hardware Considerations. While there is great advantage to using a flexible USRP hardware board, there are still drawbacks in the interface. Previous work explored problems with the USRP 1′s USB interface and showed a poor performance benchmark for a USB connection. The USB interface limited the USRP's maximum output to 8 MSamples per second instead of the theoretical limitation of 128 MSamples per second [15]. The boards upper bound is limited by the digital to analog converters (DACs).

5.6. Experimental Test Environments. Data is sent over the air between two USRPs, sourcing data from a common picture file. A picture is used for demonstration purposes, but did not have to necessarily be used. Testing under different environments could possibly require a large amount of data thus we impose the requirement that the data sent should contain the payload length and the file offset in the header information. Should the header CRC fail, we cannot trust any data within the packet and is treated as a dropped packet. Given that more data is required, the transmitter starts sourcing the data from the beginning of the file, given that the end of file character is reached. This setup allows for variable payload sizes as well as a variable amount of packets to be sent.

Three different scenarios are created within our system to test it under different conditions. These signals are generated by varying the center frequency, the bandwidth, and overall power output of a third USRP. While the noise-generating node of this network is different in that it is a USRP 2, it generates junk data which we model as noise. These environments are modeled in three ways: no noise, a close by jamming signal, and a raise in the noise floor. The ambient environment lets the engine maximize performance as much as the environment will let it. The jamming signal uses a high peak power with low bandwidth to simulate a nearby signal causing interference, whereas the noise increase is simulated

Table 4: Interference environments.

Parameter	Narrow-band	Wide-band
Power (dB)	−10	−10
Bandwidth (kHz)	600	65
Center frequency (kHz)	910000	910150

Table 5: Static parameters within the transmission.

Parameter	Value
Header modulation	BPSK
Header coding	Hamming 128
Subcarriers	28
Cyclic prefix length	4
Bandwidth (kHz)	200
Hardware gain (dB)	0

through a wide-band, low powered signal across the center frequency we are transmitting.

Table 4 shows the differences in the two types of generated interference while Table 5 shows the configuration elements that we used for the transmitted signal.

6. Performance Results

6.1. Methodology and Results. The goal of the experimentation was to compare the performance of the CE against a noncognitive (no-CE) radio that is incapable of changing its initial configuration parameters. The no-CE configuration is equivalent to a traditional wireless devices that lack capability to adapt and learn. The wireless link is incapable of making a change to its initial configuration in reaction to interference.

It is acknowledged that adaptive communications does exist today, however most use rule-based decision making that follow the same course of action each time. A purely adaptive operation is easily exploited by malicious users. An anecdotal example is swatting a common house fly. Flies have adapted the capability to jump backwards before flying upwards in reaction to movements. This makes them difficult to swat, until one understands this behavior. One can exploit this adaptation by simply aiming the swatter slightly behind the fly. Similarly, if it is known that a radio simply increases transmit power in reaction to interference, it can still be easily jammed. However, a cognitive system that employed dynamic spectrum access may identify a new channel based on past experiences on which ones were most vacant.

The methodology consisted of first defining configuration parameter initial conditions, as listed in Table 6. Once defined, a data file was sent across the link in the presence of interference, wide-band interference, and narrow-band interference signals. For the tests, a total of 2000 packets are transmitted. However, because the packet size changes for the CE system, the exact amount of transmitted bits changes for each run.

The cognitive radio follows the decision cycle described in Section 3.1. By following this decision cycle, the CE identifies when packet error has fallen below a defined threshold.

Table 6: Configuration parameter initial settings.

	TX power (dBm)	Packet size	Modulation	Coding
CE	−20	400	64-QAM	No coding
No-CE	−15	300	32-QAM	No coding

Table 7: Mean values of the meters.

		SNR (dB)	PER	Spec. Eff.	Txput. (kbps)
Ambient	CE	20.8	0.0162	3.2	637.1
	No-CE	23.4	0.0111	3.6	718.9
Wide-band	CE	10.6	0.0147	1.0	191.4
	No-CE	5.2	1.0000	0.0	0.0
Narrow-band	CE	12.5	0.0076	1.8	367.8
	No-CE	7.5	1.0000	0.0	0.0

At this point, the engine engages either the CBR or the GA depending on the availability and similarity of past cases. The solution identified from either decision-making module is then implemented to the radio. An example of these decisions are shown in Figure 6 where transmit power, packet size, and modulation change. Results tabulated in Table 7 indicate that the cognitive engine is capable of mitigating interference conditions while the no-CE performance degrades.

6.1.1. Testing Environments

(i) Interferer off: the noise floor is simply that of the lab in which the test took place.

(ii) Wide-band noise floor increase: the overall noise floor, centered at the transmitted signal carrier frequency and with a bandwidth that is wider than the transmitted signal's, is raised.

(iii) Narrow-band noise spike: a high power noise spike with a much narrower bandwidth is inserted into the transmitted signal bandwidth.

To compare the CE system versus the no-CE system, it is assumed that the no-CE case is designed for the noise environment "Interferer off."

6.1.2. Configuration Parameter Initialization Values. The initial values of the transmitter configuration parameters (transmit power, packet size, modulation, and coding) need to be determined. For the case of the CE system, they will be set to values that are close to the most ideal case, that is, close to minimum power, close to maximum packet size, highest modulation order and no coding. For the case of the no-CE system, different tests were run with different combinations of configuration parameters to find a good set of them for the case of "Interferer off." This set of configuration parameters will also be used for the other types of environments to be tested. The initial parameter values for both systems can be seen on Table 7. For the CE system, the configuration parameter ranges are as the following.

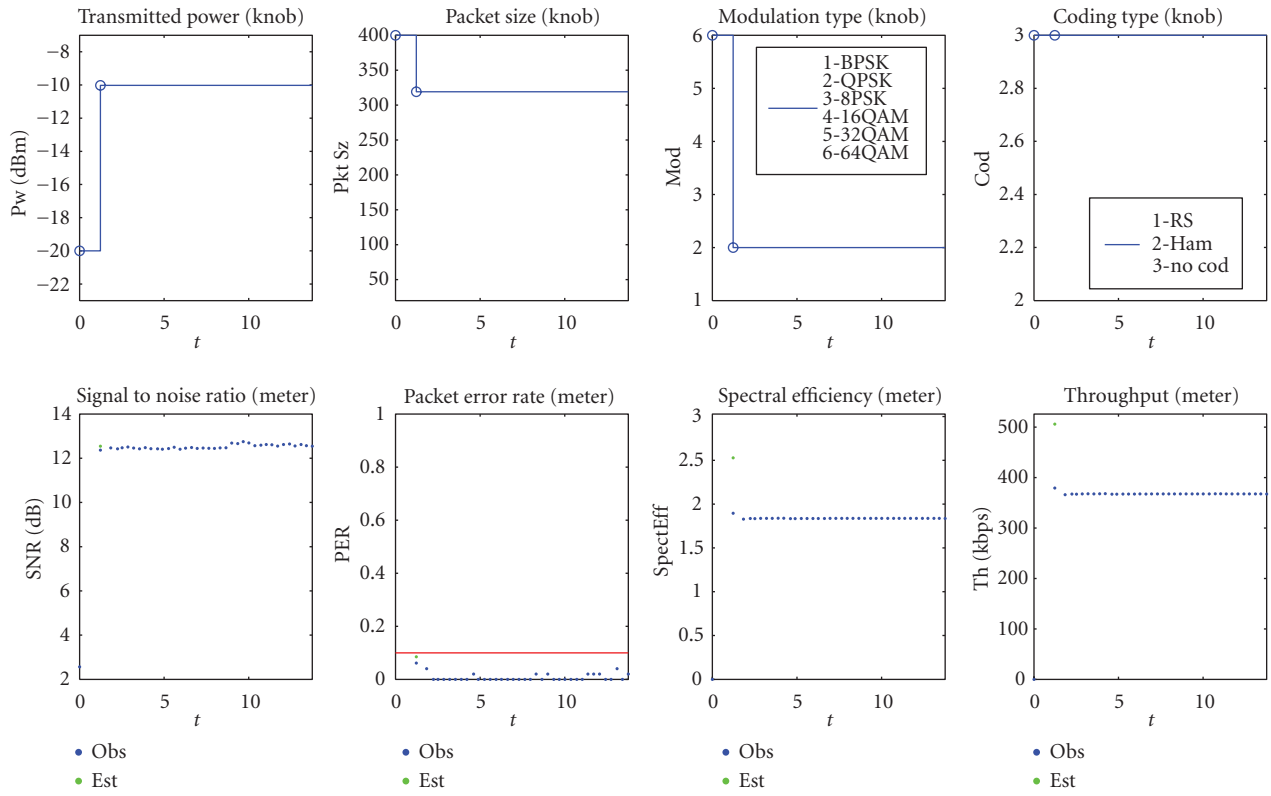

Figure 6: CE with narrow-band interference—in the presence of the narrow-band interference, the engine again raise transmit power and changed modulation to a more robust method. Packet size was able to remain higher than in the wide-band environment. Additionally, error coding was not required. This enabled a higher overall throughput to be achieved.

(i) Transmit power: (−20 dBm, −10 dBm).

(ii) Packet size: (20 bytes, 400 bytes).

(iii) Modulation: (BPSK, QPSK, 8-PSK, 16-QAM, 32-QAM, 64-QAM).

(iv) Coding: (hamming (7, 4), no coding).

Figure 6 shows how the configuration parameters and meters change during a run over time (in seconds) for the narrow-band interference environment. The results for the three environments, ambient, wide-band interference, and narrow-band interference are tabulated in Table 7. There are eight subplots in the figure.

(i) The top four show what the different values of the different configuration parameters are and what they are changed to when the CE gets called. For the case of no-CE, these values will remain constant over the whole run.

(ii) The bottom four show the values of the different meters at the time instances when they are measured. For this reason they are represented by dots. The blue dots are the meters the radio measured, while the green ones are what the GA estimated these values should be. On the PER graph, there is also a red line that represents the threshold for when the CE should get called. Therefore, if there is a blue dot in the PER graph that is over the red line, then the CE will get called (for the CE system).

On Table 7, the mean value of the meters for the CE system once it made its final decision versus the mean value of the meters over the entire run of the no-CE system are compared. As can be seen from the results, the performance of the CE system and the no-CE system for the case with the interferer off is similar. However, the CE system was able to find its solution on its own, while for the no-CE system it had to be designed and tested first. The no-CE system has a higher throughput and spectral efficiency at the cost of also having a higher transmission power than the CE system. For the other two scenarios (wide-band noise floor increase and narrow-band noise spike), the CE system was able to find a solution that allowed it to get data across the link, while the no-CE system could not operate under the new channel conditions, and therefore the PER was one. Note that the throughput for the no-CE system under the unfavorable channel conditions is 0 because the receiver could not synchronize with the received signal and could not measure it.

7. Current Limitations

7.1. Case-Base Searching. Case-base reasoning has been introduced in [16] for CR and has been used in different applications such as [15, 17] studies the impact that USRPs have on timing considerations for given networks

such as 802.11 and IEEE 802.15.4 sensor networks. The latency in these networks must be considered for a number of reasons, one of which is the clear to send/request to send protocol of 802.11. [15] shows that the USB transfer of information from the controlling computer introduces nonnegligible latency causing possible packet collisions. Similarly, given that a network has a relatively large number of nodes begin managed by a cognitive entity, an efficient cognition scheme is key. If the cognition for a wireless network introduces nonnegligible latencies, certain timing deadlines may not be met, thus causing problems within the system, such as the one described above. While this is not the only application, it is a simple example of why latency is an important issue and can be applied to a number of wireless systems that depend on computational deadlines to follow the protocol effectively.

By investigating further into the indexing schemes used for the case-base, we can improve performance of the searching algorithm used to find similar cases the engine has encountered in the past. Traditionally, the current case's similarity is calculated against each case in the case base, which can be done using the methods described in earlier sections. This scheme of similarity may be required for more structurally complex cases, but here, we can define a few simple, yet pivotal aspects of each case for fast access within the case-base. The approach being investigated eliminates the exponential dependency of the case-base size and the lookup time, allowing the case-base to grow large for more complex networks, and retain a faster access time throughout. The approach that is being investigated uses predefined thresholds for similarity relative to each parameter to eliminate the need for a similarity calculation, and uses these thresholds to index different cases within the data structure. By using the appropriate threshold for each parameter value, we can define what we determine to be similar enough in terms of distance on the vector's available search space. This is similar to [18] in that it separates the search space into binary tree decisions, however, we extend this to allow for more than two children and allow each node to index entries in the next dimension based on past cases seen. We also do not require any computation on deciding where to split the search space, as it is predefined before any case even enters the database.

To demonstrate the effectiveness of this indexing structure, Figure 7 demonstrates some preliminary results for average access times for both the traditional computation method along with a customize tree data structure to store cases. It is clear that traditional methods depended exponentially on the size of the case-base.

7.2. Estimation Methods. Currently, to estimate the BER for the GA, theoretical formulas for AWGN channels are used. However, it is well known that most real world environments do not follow this model, and that wireless signals will experience other effects such as fading. Therefore, the AWGN formulas will not accurately predict the performance of the system when changing its parameters to a different configuration.

The proposed solution to this problem is to implement a blind channel estimator, at the receiver, and to send this information back to the transmitter through the dedicated feedback channel. This new information would tell the transmitter what type of channel the system is in, so it could make better estimations of what the meters would be. this would enable the engine to find a more accurate solution for different types of coding.

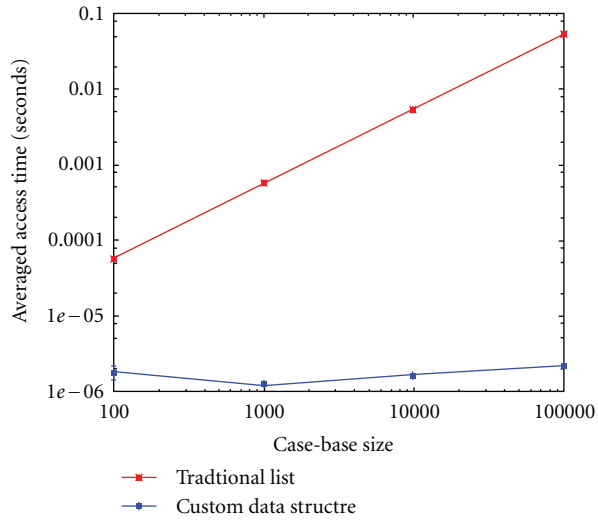

Figure 7: Average access time for the traditional storage approach versus a custom approach.

The designed system is essentially a way to perform link adaptation. In [19, section 2.2] an overview can be found of different techniques to adapt data rate and packet size. For data rate adaptation techniques, most of the described rely on some form of success rate or failure rate, a statistical approach to determine whether or not it should be changed. Other techniques use a measured SNR approach or a hybrid mechanism between the two. The described methods for adjusting packet size are based on estimating BER, using a Kalman filter or based on the success rate.

A method of adapting modulation and power is presented in [20]. It is a simple algorithm where if the target BER is not achieved, then the modulation order is progressively reduced. If the minimum modulation order still does not enable the BER to reach its desired target, then power is adjusted. It decides how much to increase the power by estimating the interference power and SINR, and then it increases the power to reach a target SINR. In [21] they adapt the modulation order of M-QAM for a Rayleigh block-fading channel. To do so, they estimate the channel; this channel estimation is not blind, so they have to use a training phase and a data phase. In [22–25], a CE is used to determine what MIMO technique and what modulation and coding to use based on feedback from the receiver sent back to the transmitter. However, their system is based on empirical observations, where a trial-and-error method is used and in which they balance exploration versus exploitation. This system would be different in two ways: an analytical model would be derived from the estimated channel information, and the GA would be used to find a close to optimal solution. To the best of our knowledge, the combination of a blind

channel estimator and a CE to find a close to optimal solution to an analytical model to fine-tune multiple configuration parameters has not yet been implemented.

8. Summary

This paper presented the implementation of a hybrid CBR-GA CR engine designed for link adaptation. The discussion included architecture, process flow, CBR similarity-based case retrieval, GA estimation methods, GA fitness definitions, and hardware/software lessons learned. The system was implemented on a USRP platform and tested in three interference environments. Performance results show that the engine is able to mitigate around interference when a non-CE fails. Limitations in the current architecture are discussed and new solutions to case searching and estimation methods were proposed.

We plan on extending this work in several ways to improve the engine's performance, such as expanding available transmission parameter support, using utility thresholds for engine triggering, simulating more accurate environments, as well as a more comprehensive study of proper case-base usage and a more accurate and dynamic channel model estimation for the GA. Most importantly, by implementing dynamic environments, we will be able to observe and tune the engine to react to dynamic spectrum access. While a frequency selection protocol has not been considered thus far in the system, we can at least expect to observe the engine's natural optimization. For example, if high spikes of power interfere with our transmission, by shortening packet length, the engine can observe better throughput in that only data sent at the time of the spike fails, instead of large packets that have error during the spikes, but good data otherwise. This provides one example of the engine reacting to the environment.

Acknowledgments

The research presented in this investigation was partially supported by the Federal Railroad Administration, Office of Research and Development, FRA Grant no. DTFR53-09-H-00021. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the view of the Federal Railroad Administration and/or US DOT. This work was also partially supported by the Institute for Critical Technology and Applied Science (ICTAS) of Virginia Tech.

References

[1] A. He, J. Gaeddert, K. Bae et al., "Development of a case-based reasoning cognitive engine for IEEE 802.22 wran applications," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 13, no. 2, pp. 37–48, 2009.

[2] J. Mitola, *Cognitive radio—an integrated agent architecture for software defined radio*, Ph.D. dissertation, Royal Institute of Technology (KTH), 2000.

[3] A. Amanna and J. H. Reed, "Survey of cognitive radio architectures," in *Proceedings of the IEEE SoutheastCon Conference: Energizing Our Future*, pp. 292–297, Charlotte, NC, USA, March 2010.

[4] 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 Tech, Blacksburg, Va, USA, 2004.

[5] Y. Zhao, J. Gaeddert, L. Morales, K. Bae, J.-S. Um, and J. H. Reed, "Development of radio environment map enabled case- and knowledge-based learning algorithms for IEEE 802.22 wran cognitive engines," in *Proceedings of the 2nd International Conference on Cognitive Radio Oriented Wireless Networks and Communications (CrownCom '07)*, pp. 44–49, Orlando, Fla, USA, August 2007.

[6] A. Aamodt and E. Plaza, "Case-based reasoning: foundational issues, methodological variations, and system approaches," *AI Communications*, vol. 7, no. 1, pp. 39–59, 1994.

[7] A. MacKenzie, J. Reed, P. Athanas et al., "Cognitive radio and networking research at virginia tech," *Proceedings of the IEEE*, vol. 97, no. 4, pp. 660–686, 2009.

[8] D. Goldberg, *Genetic Algorithms in Search Optimization, and Machine Learning*, Addison-Weslet Publishing, 1989.

[9] J. D. Gaeddert, *Facilitating wireless communications through intelligent resource management on software-defined radios in dynamic spectrum environments*, Ph.D. dissertation, Virginia Polytechnic Institute & State University, Blacksburg, Va, USA, 2011.

[10] B. P. Lathi, *Modern Digital and Analog Communication Systems*, Oxford University Press, 3rd edition, 1998.

[11] M. Ettus, "Ettus research," 2010, http://www.ettus.com.

[12] T. Newman and J. Evans, "Parameter sensitivity in cognitive radio adaptation engines," in *Proceedings of the 3rd IEEE Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN '08)*, pp. 1–5, Chicago, Ill, USA, October 2008.

[13] B. Hilburn, "Cognitive radio open source system," 2010, http://cornet.wireless.vt.edu/trac/wiki/Cross.

[14] A. Amanna, D. Ali, M. Gadhiok, M. J. Price, and J. H. Reed, "Statistical framework for parametric optimization of cognitive radio systems," in *Proceedings of the Software Defined Radio Forum Technical Conference and Product Exposition (SDR '11)*, 2011.

[15] T. Schmid, O. Sekkat, and M. B. Srivastava, "An experimental study of network performance impact of increased latency in software defined radios," in *Proceedings of the 2nd ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization (WinTECH '07)*, pp. 59–66, ACM, New York, NY, USA, 2007.

[16] T. W. Rondeau, *Application of artificial intelligence to wireless communications*, Ph.D. dissertation, Virginia Polytechnic Institute & State University, Blacksburg, Va, USA, 2007.

[17] A. He, J. Gaeddert, K. K. Bae et al., "Development of a case-based reasoning cognitive engine for ieee 802.22 wran applications," *SIGMOBILE Mobile Computing and Communications Review*, vol. 13, pp. 37–48, 2009.

[18] S. Wess, K. Dieter Althoff, and G. Derwand, "Using k-d trees to improve the retrieval step in case-based reasoning," in *Proceedings of the Selected papers from the 1st European Workshop on Topics in Case-Based Reasoning (EWCBR '93)*, S. Wess, K. Dieter Althoff, and M. M. Richter, Eds., pp. 167–181, Springer, London, UK, 1993.

[19] N. C. Tas, *Link adaptation in wireless networks: a cross-layer approach*, Ph.D. dissertation, University of Maryland, College Park, Md, USA, 2010.

[20] K. Jayanthi, *Some investigations on quality improvement using link adaptation techniques in cellular mobile networks*, Ph.D. dissertation, Pondicherry University, 2010.

[21] A. Soysal, S. Ulukus, and C. Clancy, "Channel estimation and adaptive m-qam in cognitive radio links," in *Proceedings of the IEEE International Conference on Communications (ICC '08)*, pp. 4043–4047, Beijing, China, May 2008.

[22] H. Volos, C. Phelps, and R. Buehrer, "Initial design of a cognitive for mimo systems," in *Proceedings of the SDR Forum Technical Conference and Product Exposition (SDR '07)*, 2007.

[23] H. Volos, C. Phelps, and R. Buehrer, "Physical layer cognitive engine for multi-antenna systems," in *Proceedings of the IEEE Military Communications Conference (MILCOM '08)*, pp. 1–7, San Diego, Calif, USA, November 2008.

[24] H. Volos and R. Buehrer, "Cognitive engine design for link adaptation: an application to multi-antenna systems," *IEEE Transactions on Wireless Communications*, vol. 9, no. 9, pp. 2902–2913, 2010.

[25] H. Volos and R. Buehrer, "Robust training of a link adaptation cognitive engine," in *Proceedings of the IEEE Military Communications Conference (MILCOM '10)*, pp. 1442–1447, San Jose, Calif, USA, November 2010.