Scalable Parameter Management using Casebased Reasoning for Cognitive Radio Applications

Daniel R. Ali

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Engineering

Jung-Min Park, Chair
Ashwin E Amanna
Thomas Charles Clancy

May 18, 2012
Blacksburg, Virginia

Keywords: Cognitive Radio, Casebase Reasoning, Data Structure

Scalable Parameter Management using Casebased Reasoning for Cognitive Radio Applications

Daniel R. Ali

(ABSTRACT)

Cognitive radios have applied various forms of artificial intelligence (AI) to wireless systems in order to solve the complex problems presented by proper link management, network traffic balance, and system efficiency. Casebased reasoning (CBR) has seen attention as a prospective avenue for storing and organizing past information in order to allow the cognitive engine to learn from previous experience. CBR uses past information and observed outcomes to form empirical relationships that may be difficult to model apriori. As wireless systems become more complex and more tightly time constrained, scalability becomes an apparent concern to store large amounts of information over multiple dimensions.

This thesis presents a renewed look at an abstract application of CBR to CR. By appropriately designing a case structure with useful information both to the cognitive entity as well as the underlying similarity relationships between cases, an accurate problem description can be developed and indexed. By separating the components of a case from the parameters that are meaningful to similarity, the situation can be quickly identified and queried given proper design. A data structure with this in mind is presented that orders cases in terms of general placement in Euclidean space, but does not require the discrete calculation of distance between the query case and all cases stored. By grouping possible similarity dimension values into distinct partitions called *similarity buckets*, a data structure is developed with constant ($\mathcal{O}(1)$) access time, which is an improvement of several orders of magnitude over traditional linear approaches ($\mathcal{O}(n)$).

Acknowledgments

I would like to acknowledge my committee chair and academic advisory Dr. Jung-Min Park for his guidance through my graduate research as well as provide me with invaluable opportunities that have advanced my academic and professional career. I would also like to acknowledge my committee Ashwin Amanna and Charles Clancy for serving on my council. Ashwin provided me with the guidance and opportunity through my graduate career and has been one of the best bosses I have ever had the pleasure of working for.

I would like to thank David Gonzalez as a friend and co-worker for invaluable critiques on my work as well as challenging me to become a better engineer. I would like to thank Joe Gaeddert for his work and contributions to wireless communications research which made my work possible. I would like to also acknowledge everyone that has worked on the FRA project, including Matt, Manik and Soumava.

Finally, I would like to acknowledge everyone in the MPRG, Wireless@VT, and in the Bradley department that has helped me through one of the most challenging and beneficial parts of my life. Thank you.

Dedication

This work is dedicated to my Mother, my Father, and my little brother, Dean. Thank you for a lifetime of love and support.

List of Abbreviations and Notations

$\boldsymbol{\beta}$	Performance Observables
$\boldsymbol{\delta}$	Similarity Dimension
$\boldsymbol{\phi}$	Uncontrollable Environment Descriptors
$\boldsymbol{\theta}$	Changeable System Parameters
$\boldsymbol{u}$	Utility Vector
$\boldsymbol{w}$	Weight Vector
η, σ	Utility Spread Parameters
p	Utility Goal Descriptor
q	Utility Function
u_{global}	Overall Fitness
k-NN	k-Nearest Neighbor
AI	Artificial Intelligence
ANN	Artificial Neural Networks
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
CBR	Case-based Reasoning
CE	Cognitive Engine
CR	Cognitive Radio
DSA	Dynamic Spectrum Access

DSP Digital Signal Processing

GA Genetic Algorithm

HMM Hidden Markov Models

LTE Long Term Evolution

OFDM Orthogonal Frequency-Division Multiplexing

QAM Quadrature Amplitude Modulation

SNIR Signal to Noise plus Interference Ratio

USRP Universal Software Radio Platform

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Since the advent of cognitive radio technology, practical applications use a number of different techniques to optimize wireless and networked communications. The need for scalable implementations are critical due to growing system complexities and stringent user requirements. Case-based reasoning has been identified as a candidate to provide or assist the main cognition cycle of CR technology by storing information about past experiences. Learning from past experiences is critical for CR technology and is found in most definitions. As this requirement is prevalent and CR systems grow in complexity, practical information management is critical to create realistic implementations.

1.2 Case-based Reasoning for Cognitive Radios

Within CR research, various methods have been used to realize the concept of Mitola's vision, which first introduced the concept of CR [31]. One important aspect prevalent in most research is that CR devices must learn from past experiences. CBR provides a well studied area of research that can satisfy this requirement in a modular sense. Other implementations such as artificial neural networks (ANN) or hidden Markov models (HMMs) provide a tighter coupling of past experiences with their search through the solution space. CBR can provide a decoupling of these concepts and provide a way for a cognitive engine (CE) to easily access past information while using its choice of cognitive optimization.

Traditional approaches of CBR in CR applications use a system of similarity to compare the current situation to past experiences. By calculating a weighted Euclidean distance between the features of the query case and all of the cases in the casebase, casebased cognitive radios use this metric to identify a similar case or set of cases [6, 21]. Once a case or set of cases

has been identified as similar, the information from the case(s) can be applied to the current situation.

As CR finds more niches to fill in the field of wireless communications, the internal cognition cycle is tasked with more responsibility in terms of the various aspects of the system it manages. These aspects and their various descriptions provide the cognitive engine with the information it needs to remember the state of the radio at the time the decision was made. As more information is required to describe the situation the CR is in, the number of possible configurations grows extremely rapidly. While not all areas of solution space need to be enumerated, a CR can undergo a number of situations quickly depending on the granularity of the system description and how often the cognitive cycle is invoked.

1.3 Contributions

CBR presents a well studied area of research that can be applied to CR technology to facilitate information gain from past experiences. While it may have disadvantages as a stand-alone application for the entire domain, it does present a reasonable avenue for information storage from a better suited cognitive entity, such as a genetic algorithm (GA) [43] or fuzzy logic [23]. These contributions have made a significant impact on CR presenting a modular way to store and retrieve past information. As CR technology is applied to various aspects of wireless and networked communications the need for proper information management of past experiences is critical in order to form a centralized cognitive entity whose operation does not hinder the wireless system.

This work revisits CBR as applied to CR technology, demonstrating a renewed look at information abstraction, case representations and similarity considerations. The main goal of this work is to revisit the case representation and indexing module of CBR to reduce the burden on the other modules. Typically, four modules are present in CBR [27]: case representation and indexing, case selection and retrieval, case evaluation and adaptation, and case database population and maintenance. Because it is assumed that the third module, case evaluation and adaptation, is performed by a more complex cognitive entity, the underlying casebase data structure can incorporate case representation, and retrieval to allow the CE more time to optimize the parameters of the case. By doing so this indirectly reduces the need for casebase maintenance as well as conservative population size constraints. By using existing approaches to representational information in wireless communications [15], an abstract framework is developed for case representation. By de-coupling the whole of a case's content and the exact parameters that a CE requires of the case to be similar, a more accurate representation of similarity can be defined for practical CE implementations. Furthermore, by bucketing these parameters a data structure for this approach is designed and investigated that provides little impact from accessing its information on the cognitive architecture. This data structure is compared to traditional approaches and shows performance increases by several orders of magnitude, especially as the casebase grows large. This is possible by

bucketing the parameters that are considered when searching for similar cases. The buckets are presented in an abstract fashion so that the specific application can design the most appropriate parameters for use in that system. Application of this data structure is simulated in a specific scenario to demonstrate at least some level of convergence within the scenario's problem space.

Using this approach allows cognitive entities to store vast amounts of useful information without the pitfalls of slow access times. I first introduce this concept of this data structure is first introduced in [6]. Because no other software libraries, besides common GNU C++ libraries, were used in the development of the data structure, it can be deployed easily without reliance on additional software packages.

Chapter 2

Cognitive Engine Design

2.1 Introduction

CR problem-solutions can be seen as a complex optimization problem with one or more different local maximum and a single global maxima. These maximum can change with even slight variations on the system. Environmental considerations make this inherent as no radio will experience a single, unchanging environment. The very nature of additive white Gaussian noise (AWGN) implies stochastic variation in the system. In real world applications, multi-path and fading are almost always apparent as well, presenting researchers with the difficult task of automating solutions for radios to work well in these environments. Using cognitive technology research has applied this learning technique that require not only new solutions to new problems, but also learning from past experiences. How these tasks are implemented is the core of CR research.

Joe Mitola is credited with the inception of this idea and saw CR as a way to understand and react to changing user needs [31]. His definition has evolved over time into a more communications oriented point of view. As others have stated the exact definition of CR is still up for debate and has been defined many times over [39]. Even still, much lower level details of the PHY and MAC layer are apparent in different applications and the definition used in this work will assume the definition presented in [39]:

> The goal is to build a flexible, reconfigurable radio that is guided by intelligent processing to sense its surroundings, learn from experience and knowledge, and adapt the communications system to improve the use of radio resources and provide desired quality of service.

In the context of this work, there are two relevant aspects of a CE that must be considered. First, a mechanism must be in place to generate possible solutions for the problem encountered by the radio. Secondly, a way to store these solutions, their respective performance,

and the operating conditions is focused upon to provide reliable information in a timely manner. There are numerous aspects to a CR and CE design, such as policy, inter-process communication, security, etc., I go over the relevant aspects of CE design to establish a backdrop to stage the presentation of the main contributions of this work.

Much research aims to quickly converge on an understanding of the environment a radio is in and using available transmission parameters, adapt itself to the global maxima [44, 6, 42, 22]. Using this view on CR, both link adaptation and proper dynamic spectrum access (DSA) can store information on the decisions it made in terms of adaptation it just requires proper identification of the parameters used. Parameter description is a crucial part of a CE design as it is the basis of integrating a well planned high level design into an applications specific one. This chapter's goal is to define, at a high level, the relevant aspects of a CE that will provide the information necessary to adapt the contributions of this work to a specific application.

2.2 Cognitive Engine Approaches

The CR problem domain is large and many implementations have proven to be successful for different applications [20]. In this section I review some of these approaches and highlight the advantages they provide. It is important to understand these approaches and the problems they solve order to know how CBR can be used effectively and efficiently. Because CBR is based on case similarity, it is also important to correlate how these approaches use past information to solve unseen or poorly solved problems. CBR facilitates a way for CR designs to store and effectively reuse information the radio has learned through its optimizations. The retaining and reuse of information is a critical aspect of CE and CR design and is seen in most definitions of CR [15, 39].

2.2.1 Artificial Neural Networks

ANNs are created by creating a set of nonlinear functions with adjustable parameters to simulate a network of neurons. Each neuron is comprised of a single function that takes multiple inputs to produce a single output. Their exists many ways to form and structure these networks, allowing for a multitude of applications [19]. ANNs are well suited for recognizing received stimuli as well as solution adaptation. This makes it well suited for CR applications. In CR research, ANNs have been developed for spectrum sensing applications [13, 48]. These applications used them to classify different wireless signals online, which is useful for dynamic device interoperability. Similar to the simulations performed in this work, ANNs have been used for parameter adaptation [46, 18].

ANNs provide advantages in several ways. They scale well conceptually, which is critical as CR and CE design becomes more complex with more of the network stack to manage.

They also can be created from just a few examples, relying on past experiences to create a complex network. This is a crucial aspect in that it learns by experience which is the main focus of this thesis. ANNs tightly couple the information learned with proposed solutions, which is different than the assumption presented here, which is that higher level cognition is its own entity with a separate set of data to contain its experience.

2.2.2 Hidden Markov Models

HMM is a useful tool to develop statistical relationships of random processes [37]. By modeling the states of the output of a system, various analytics can be applied to the states to learn more about the system in terms of the observed states. These states can be known or unknown a-priori and are considered "hidden". These hidden states are then discovered through observation and incorporated into the analysis of the system. HMMs have been applied to three different problem domains [37]; an evaluation or recognition problem, which is used to compute the probability that a specific observation sequence will be observed given the parameters of the model. Secondly, the decoding problem which finds the sequence of hidden states that best explains the observation sequence. Lastly, the training and learning problem, which uses an observation sequence to find the most likely set of state transitions and observational outcomes.

HMMs is useful to identify and classify stimuli and have been applied to channel modeling [40]. The advantage of this is that if a CR can deduce what device is transmitting in a certain frequency band purely on observed signal alone with no meaningful interaction, the CR then has some preliminary information on the device, its expected protocols and frequency occupancy priority, etc. Rondeau *et al.* used HMMs to model wireless channels during a CR transmission. In [24] Kim *et al.* used HMMs to identify cyclostationary features for primary signal detection. Gosh *et al.* formulated the problem of spectrum sensing using Markov models proving its existence amongst wireless applications [16]. This was done using real-time measured data. HMMs have also been used to describe spectrum occupancy prediction [5]. This provides useful information to CR to significantly increase signal to noise and interference ratio (SNIR) . HMMs build the set of states it considers based on the observations of the system outputs. In comparison to CBR, HMMs can see resemblance between the states it considers and the cases in CBR. HMMs focus on the transitions from states and can be useful in CBR to understand the situation transitions that cases store and base similarity on.

2.2.3 Rule Based Reasoning

Rule based reasoning uses an approach of applying current data to a set of rules described in an IF-THEN format [35]. The rules can be built into the system and executed in any order. Conversely, these rules can be learned autonomously and used dynamically as new

data is observed and more conclusions can be reached about the system. This type of system is generally split up into two functional modules, an inference engine and a rule base. The inference engine is what decides which actions to take based on the conclusions reached by sifting the observed data through the rule base. In [47] a rule based reasoning was used to develop an application for IEEE 802.22 applications, which achieves similar performance to a genetic algorithm approach but with less complexity. This is one of the major trade-offs for rule based reasoning in that it is much easier to implement and has less computational complexity, especially when compare to a GA approach. In [10] a RBR engine was developed using predicate calculus using a generic CR architecture to be used in the reasoning process.

Rule based reasoning stores the information it considers useful in its rule base. This is useful in that the cognitive entity in the system develops relationships and from those relationships rules can be formed that shape well suited solutions. The approach taken in this work is based mainly in storing discrete instances of parameter application. If each situation the radio encounters can be described by a set of parameters, there are advantages in these discrete applications have a measurable outcomes which may provide more precise information than a rule outcome may infer.

2.3 Genetic Algorithms

Genetic algorithms are discussed here at length to describe a commonly used way of generating new parameters for a given situation. GAs have been commonly used in conjunction with CBR [33, 6, 9], and are well suited for searching a large solution space to find a globally optimal solution. There are issues, however, as it can be complex and take more time than is allowed for some applications.

A biologically inspired CE design is presented through the works of Rieser *et al.* [38] which introduces two important aspects of a CR. First, a GA is introduced to traverse the search space of possible configuration parameters that is available to the radio. Newman introduced the concept of using past experience as a starting set for GA population adaptation [33] and is revisited in more detail to provide a better description of its integration [39]. Within the context of this work, the CBR is used to provide more then as a mere primer for the GA. Due to the reasons discussed below, GA usage in embedded radio system can cause detrimental impact on a CE's effectiveness, proving to be too complex [15]. For these reasons, the CBR should be relied upon whenever possible to provide a solution orders of magnitude faster than is possible with a GA. A CBR casebase can be seen as a cache of empirical measurements of real values applied to a specific situation.

Rieser's dissertation goes into detail on the application of GAs to CR, demonstrating the chromosome structure and the interaction between the different components of a GA [38]. GAs are used as an optimization heuristic to search a parameter set for an optimal set of values. This is inspired through biological functions like reproduction and evolution in

order to simulate a rapid progression of a populous through multiple generations. Mapping raw parameter values into acceptable genes inspired the idea of bucketing parameters in the casebase. By defining the parameters that are considered by the search algorithm into different chromosomes, the GA evolves its set of considered parents into a set that has what is generally accepted as a high probability of having an acceptably good solution. This is done through a few different techniques known as selection, crossover, and mutation. Selection creates a pool of candidate solutions from the general population to be crossed over. Crossover is the matching of parents to one another to produce offspring which will have a mix of genes sourcing from either parent. The offspring can then be further mutated to ensure the system doesn't converge to a local optima. Each member is evaluated via a utility function to determine which chromosome selections produced the best results. Those chromosomes deemed fit enough survive to the next generation to reproduce, while the others are discarded. An example of a chromosome is shown in Table 2.1 to some n entries [38].

	Gene			
Chromosome Number	Power	Frequency	Code Rate	Modulation
0	5 dBm	2.5 GHz	5/6	BPSK
		⋮		
n	0 dBm	2 GHz	1/2	QPSK

Table 2.1: Chromosome Example

In some research, the mutation aspect of chromosome augmentation requires specific patterns of bit flipping with certain probabilities [32]. This conflicts with the wide range of possible values that each gene may take on. For instance flipping a psuedo-random bit (there is some probability theory behind the bit position selection) in the string sequence "BPSK" will most likely result in an unintelligible string. The same applied to other types commonly found in computer languages such as floating point representations as well as signed integer types. A mapping is required to interpret these non-uniform binary representations and may add additional computational complexity to an already complex system of evolution. This is especially strenuous when required by the fitness function which must re-map each gene of each chromosome back to some "model space" for every evaluation of each member of an evolutionary population. Model space is considered the raw values that can be used in wireless estimation functions such as in theoretical bit error rate (BER) curves. In [6] I address this concept and discuss the mappings used for the system.

Furthermore, genetic algorithms are required to model the environment in which they are operating in. Because they are not able to empirically try every case that is contained within its population, these must be based in a theoretical evaluation and are charged with accurately providing a method of determining the best set of transmission parameters [32]. This is a difficult task however, as the functions used to accurately describe the inputs and output of a wireless system are not always static and can change dynamically over time [7]. Not only do these functions change, but are required to be applied to every chromosome

in each population to determine performance. This can be very taxing for even powerful computer given a large enough population size. The problem is compounded when considered in an embedded sense of CR design.

While the GA itself can present the problems described in this section, it in no way nullifies its well studied functionality to the system. While a GA is still accepted by many to provide a useful way to traverse a solution space [20], the importance of efficiency gains by using past experience should be considered when performing on-line optimization. This will generally provide better performance given that the database structure representing the past knowledge is quickly accessible and the information useful to the reasoning entity.

2.4 Case-based Reasoning

When CBR was first introduced to CR it was under the guise of Case-base Decision Theory (CBDT), which has a natural correlation to CBR [39]. Here we consider the application of CBDT to be synonymous with CBR. CBR incorporates a group of case manipulation mechanisms to drive its reasoning. These including case representation and indexing, case retrieval, case adaptation, and case-base maintenance [36]. In CR literature CBR is usually coupled with an adaptation mechanism to choose new parameters given the case-base is not sufficiently populated to produce a case with similar enough characteristics or a case with sufficiently acceptable performance. Multiple adaptation techniques were surveyed for IEEE 802.22 WRAN applications including a hill-climbing search, a GA, and a combination of the two [21]. CBR has also been coupled with fuzzy logic to determine the type of operating channel [23]. This separation shows a divide in the definition of CR in that the CBR aspect provides the mechanisms to retain and organize empirical knowledge versus adaptation and the ability to respond to situations it has yet to undergo. It is in this, that CBR can be used in a diversity of scenarios that require, or can make use of, a modular system component that retains and organizes information from past experiences.

The term "Case-based reasoning" is an umbrella terms used to encompass a few different forms of artificial intelligence (AI) . It is important to classify which specific application of CBR is best suited for CR, as it has many applications. Exemplar-based reasoning is a classification problem in which many exemplars are classified to represent the class's description. An unclassified exemplar presents the problem and must be classified. Instance based learning uses a highly syntactic CBR-approach, commonly using feature vectors, with discrete values as opposed to a generalize knowledge base. This remains to be seen as a classification problem, in that it attempts to identify clusters within of similar cases within its space [4]. Memory-based reasoning is a focus group of CBR and views the collection of cases as a large memory and the reasoning processes as accessing and searching this memory. This work relates to this field in that it focuses on memory organization and a specific retrieval algorithm for a static case representation. In the typical CBR domain, cases vary with their structure and have a degree of complexity and thus richness of information. This

domain also encompasses some form of case modification or adaptation of retrieved cases. In this work, the complexity of the cases is static and the adaptation of case information is a job left to a higher level of the cognition within the engine. Analogy based reasoning uses cases from a different field of view to solve cases in a completely separate domain. An example of a singular case in this view would be the use of biological evolution as the basis for a optimization heuristic algorithms, such as that discussed in Section 2.3. However, this approach may be difficult to implement any kind of on-line approach for an area as specific as cognitive radio.

While each approach has merit and can be applied to CR applications, this work views the problem in the *memory-based reasoning* domain. Traditionally, memory-based reasoning research has focuses on using parallel processing and parallel memories [25] to improve performance. By adding faster hardware implementations systems are able to solve problems faster, however, I revisit this subject with two different advantages at my disposal. This work is presented in terms of proper memory organization using modern day computational power, reducing the requirements of the data structure in terms of size and access time. I also redefine similarity for our specific situation in that each feature vector of a CR's case can be partitioned in terms of each similarity dimension. This eliminates the need for a similarity calculation to quantify a discrete distance between cases. Section 3.3 discussed this is greater detail.

CBR is based in the idea that every case contains information related to the situation that the reasoning entity underwent at some point in time. While there are many ways to represent and implement information, three common factors are seen in each case that a CBR stores; a problem, an action, and a result. This section introduces the idea of CBR to CR and reviews the literature of its latest application. Chapter 3 provides an in depth analysis on the relevant system components of CBR and how they apply to our system.

2.5 General Parameter Description

In a simplified view, a radio's transmission parameters are at the heart of its ability to control aspects of the system to improve system efficiency and overall throughput. As CR research grows to more complex applications, the number of parameters considered grow as well. This is the case with LTE network technology as CR technology is used for bandwidth allocation strategies [2]. The relationship between adjustable parameter inputs (commonly known as *knobs*), environmental observables, and resulting performance metrics (commonly known as *meters*) is explained in detail in [15]. I review these definitions to provide the basis of understanding of how the CBR component within a CE can be designed from an abstract stand point. These allow an avenue to quantify a numerical understanding of the capabilities of a radio, its operating environment, and the measure of its performance both in traditional terms (throughput, BER, spectral efficiency, etc.) and in a goal-oriented environment (parameter utility, combined overall utility, adequate case relevance etc.).

CRs and CEs alike have multiple parameters available for them to augment and change to provide a dynamic performance gain. These parameters are assumed to be reactive in nature in that they are goal oriented and performance metrics have the ability to dictate whether there is a "problem" or not. How a radio can define its problem is discussed in more detail later. These changeable parameters are considered as a vector $\boldsymbol{\theta} = \{\theta_0, \theta_1, ..., \theta_{N_\theta - 1}\}$, containing N_θ elements in the system. Environmental observables are considered to be uncontrollable parameters that describe the conditions the radio is operating in. This can range from path loss to number of users in the system to application requirements. This is defined by the vector $\boldsymbol{\phi} = \{\phi_0, \phi_1, ..., \phi_{N_\phi - 1}\}$ similarly containing N_ϕ possible elements. A wireless transmission's performance can be viewed in different ways, such as spectral efficiency, throughput, overall system throughput, bit errors, signal strength, etc. These performance parameters are considered as $\boldsymbol{\beta} = \{\beta_0, \beta_1, ..., \beta_{N_\beta - 1}\}$ consisting of N_β parameters.

These vectors of parameter interact in different ways to produce the observable measurement statistics. Abstractly one can view the environment, hardware considerations, or any reasonable factor that may impact the performance measures of the radio as a black box represented by a function f. Formally,

$$\boldsymbol{\beta} = f(\boldsymbol{\theta}|\boldsymbol{\phi}) \tag{2.1}$$

By wholly understanding f, a radio will then have the knowledge to know which $\boldsymbol{\theta}$ parameters will produce the "best" possible $\boldsymbol{\beta}$ parameters. This is the primary problem with CR in that f is extremely difficult to understand, especially in real time and autonomously. Communications theory demonstrates a mathematical representation of understanding for this function, but may not always perform as expected in implementation. The advantage of CBR is the ability to store sufficient empirical information on f such that it does not need to adapt a case or create one anew based on a reasoning entity's solution generation.

As a simple example, consider the measurement of RSSI as β_0, transmission power as θ_0, ϕ_0 as path loss, and that $N_\phi = N_\theta = N_\beta = 1$. Typical RSSI graphs demonstrating a sample of the type of fading (as shown in Figure 2.1) shows the plot of $\boldsymbol{\beta}$ over time.

Figure 2.1 presents an overly simplified view of the characteristics of a wireless channel. More complicated estimation techniques can be used to identify statical understandings about the channel (e.g. Rician or Rayleigh fading estimations). This is especially relevant when considering the channel against the properties of the waveform produced at the transmitter. Knowing how the waveform built at the transmitter interacts with its environment is crucial in understanding how to best build a waveform. Comparing coherence time and coherence bandwidth to symbol time and bandwidth used provides more details about the type of channel. These characteristics can extend $\boldsymbol{\phi}$ and provide better understandings of the environment to the reasoning engine. Figure 2.1 presents a simple simulation over an orthogonal frequency-division multiplexing (OFDM) in a Rician Fading channel.

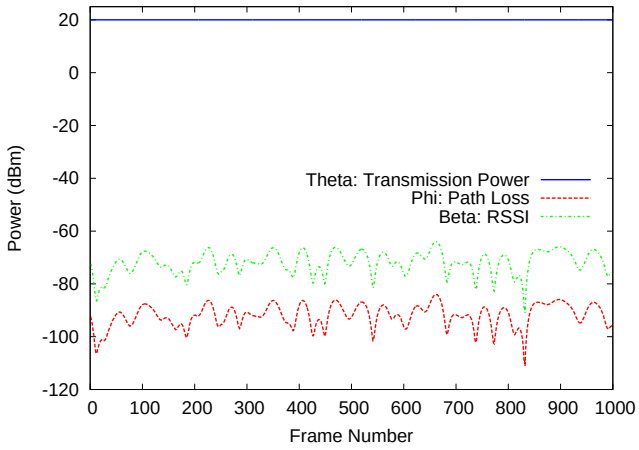


Figure 2.1: Plotting a simple f

2.6 Utility: Measuring Parameter Usefulness

Each parameter considered in the system may or may not map to a space that the reasoning engine can find useful. $\boldsymbol{\phi}$ is meant to provide a set of parameters that are measured by the CR and provide useful information, but the actual values of these metrics provide no additional information beyond what the channel was actually like during that point in operation. This is opposed to a controllable parameter like transmission power. Transmission power can provide a double edge sword in terms of utility as it can offer both better signal strength as well as long battery life if used efficiently. Both signal strength and battery life source their sample information off the raw dBm values that this metric provides. This section's aim is to go over how an appropriate mapping of raw values to a common usefulness metric, known as utility, can provide a good backdrop for the set of goals of a CR system.

2.6.1 Utility

[15] provides a meaningful utility function described by three main parameters. p , which is actually made up of the descriptor $\dot{x}$ and an input x, η, and σ. $\dot{x}$ describes the upper goal that the raw value of the parameter (x) should map to. η is the threshold parameter and σ is the spread parameter. η and σ can be chosen to be any arbitrary value, but are chosen such that once the upper goal is reached in the parameter space, a utility of 95 % will be obtained.

$$q(x, \dot{x}; \eta, \sigma) = \frac{1}{2}\left\{1 + \tanh\left[\log\left(\frac{x}{\dot{x}}\right) - \eta\right]\sigma\right\} \tag{2.2}$$

This is considered an "upper" goal in that it reaches the higher utility value. These mappings

are described formally in Equation 2.2, where the hyperbolic tangent function is $\tanh(z) = (e^z - e^{-z})/(e^z + e^{-z})$ and $p = x/\dot{x}$. The values of η and σ are described when $q(\dot{x}) = q_0$ and $q(p\dot{x}) = q_p$ such that

$$\sigma = \frac{1}{2\log p}\left[\log\left(\frac{1}{q_0} - 1\right) + \log\left(\frac{q_p}{1 - q_p}\right)\right] \tag{2.3}$$

and

$$\eta = \frac{1}{2\sigma}\log\left(\frac{1}{q_0} - 1\right) \tag{2.4}$$

This function produces a monotonically increasing utility function that provides a means of mapping raw parameter values to a measure of usefulness to the system. Figure 2.2 shows this curve for a generic logarithmic parameter. Note that q is bounded $q \in (0, 1)$, providing a common grounds for comparison. It is important to note that this mapping is extremely important to a CR in that it provides a relative ground of comparison to the system. Comparison of solutions based simply in parameter space are not sufficient. Equation 2.2 provides a common equation which can be applied across all utilities while still preserving the unique attributes of the parameter it is describing. This is done by specifying p for each parameter.

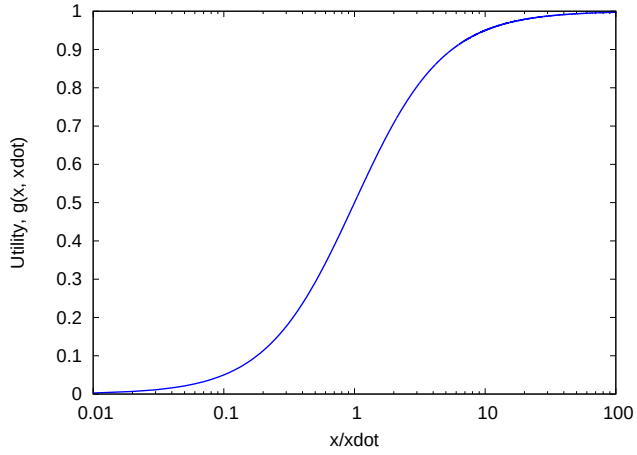


Figure 2.2: Generic Utility Function

This function has seen use in CR applications [47] and [21] and is used here for several reasons as described in [15]. As describe before Equation 2.2 is monotonically increasing which improves convergence on most search algorithms. This prevents gradient based search algorithms from hitting a plateau. Because 2.2 is a ratio of the metric x to its goal $\dot{x}$, units are abstracted away from the underlying parameter space it is describing. Finally,

an unattainable upper bound allows engines and reasoners alike to always search for better performance. The diminishing marginal return on the utility however, provides smaller returns the further above its upper bound it is. This is useful when trying to optimize a global utility.

This generic utility function describes a single utility, however, there will be a number of utilities in the system that need to be measured. Thus, the vector $\boldsymbol{u} = \{u_0, u_1, ..., u_{N_{\boldsymbol{u}}-1}\}$ is used to describe the value of h for $N_{\boldsymbol{u}}$ utilities. Although a simple search algorithm was used in this work, this description of utility provides a convenient mapping making it easier for solution space searching algorithms to plug into the framework described here for CBR. The use of the subspace mappings is also helpful in visualizing the relationships between $\boldsymbol{\phi}$, $\boldsymbol{\theta}$, $\boldsymbol{\beta}$, $\boldsymbol{u}$ and u_{global}. These parameters are the basis for the case description presented here.

2.6.2 Fitness

In CR applications, a natural design trade-off is usually inherent in the different $\boldsymbol{\theta}$ parameters of a CR system. Power, again, is used as an example of having two different and conflicting utilities. These utilities being signal strength and battery consumption. Not only might higher power levels drain the battery but cause harmful interference known as the saturation effect in a single link scenario. In a networked scenario, this effect is still possible to each link as well as cause harmful interference to other radio links within range. CR implementations need to balance this trade-off effectively and fairly. In this example a weight can be introduced for each utility, giving more importance to reducing power due to compounding effects if too much power is in the air. These weights can then be incorporated into a fitness function to appropriately reflect the relative weighted values of each utility. They are assumed as a vector in $\boldsymbol{w} = \{w_0, w_1, ..., w_{N_{\boldsymbol{w}}-1}\}$ of $N_{\boldsymbol{w}} = N_{\boldsymbol{u}}$ weights. Each weight must be designed according to the goals that the CR aims to achieve. These goals are assumed static and unchanging once deployed.

Fitness is considered as a global utility u_{global} which quantizes the performance of the radio into a singular value. By combining the underlying utilities, the radio has a convenient way of gaging how well the solution it generated did. [32] introduces a weighted utility function described in Equation 2.5 viz

$$u_{global}(\boldsymbol{u}|\boldsymbol{w}) = \sum_{i=0}^{N_{\boldsymbol{w}}-1} w_i u_i \tag{2.5}$$

with the constraints that

$$w_i \geq 0, \forall i = 0, 1, ..., N_{\boldsymbol{w}} \tag{2.6}$$

and

$$\sum_{i=0}^{N_{\boldsymbol{w}}-1} w_i = 1 \tag{2.7}$$

While this is an intuitive approach to consolidating utilities, it has a flaw. Given that each utility was to have some measure of importance in the system, and some utility was reduced to 0, u_{global} would not adequately reflect this. Even if that utility were given plenty of weight. For instance, if throughput was considered a utility and it were reduced to zero, if the other utilities considered by the system are doing moderately well, then the fitness would reflect a moderate fitness, whereas zero throughput should map to zero fitness as no connectivity presents no practical usefulness to the system.

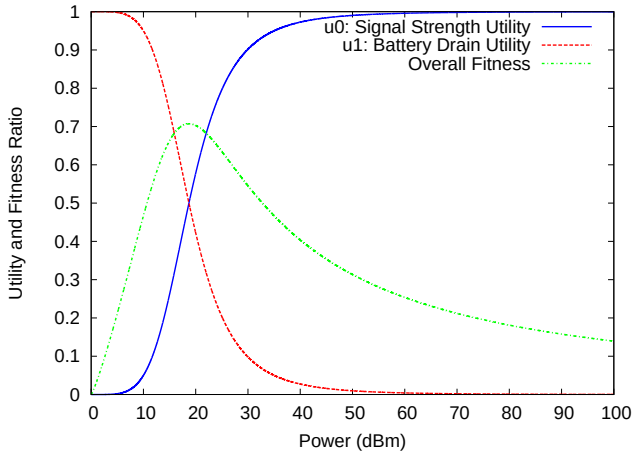
This work uses a weighted production function instead of a weighted summation function as described in [15]. This equation is shown in Equation 2.8 which is on the same bound as $q(\boldsymbol{u})$ such that $u_{global} \in (0, 1)$.

$$u_{global}(\boldsymbol{u}|\boldsymbol{w}) = \left[\prod_{i=0}^{N_{\boldsymbol{w}}-1} h(u_i)^{w_i}\right]^{1/N_{\boldsymbol{u}}} \tag{2.8}$$

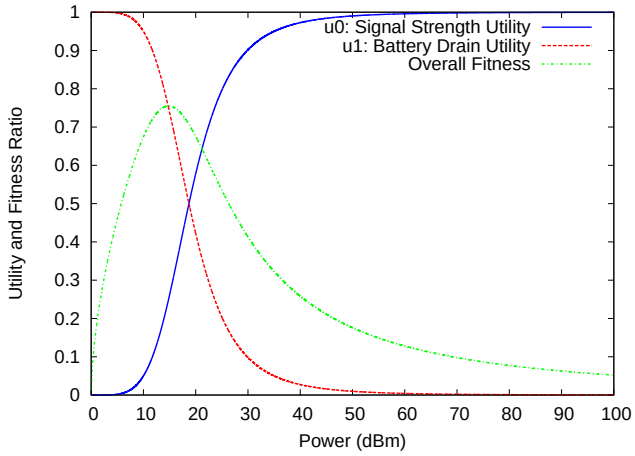
To demonstrate the impact weight can have on this fitness function a situation is developed of two utilities. Using our previous example consider u_0 is Signal Strength and u_1 is Battery Usage. Allow $q_0 = 0.95$ and $q_p = 0.05$. For signal strength $p_0 = 10/35$ and to equally pit battery power $p_1 = 35/10$. In Figure 2.3a we can see that when the weight is evenly split between both parameters at $w_0 = w_1 = 0.5$, an optimal fitness value can be reached of $u_{global} = 0.707$ at $x = 18.75$. If we wanted to skew this to favor battery life over signal strength for the reasons stated above, we simply set the weights differently such that $w_0 = 0.75$ and $w_1 = 0.25$. The result of this skew is shown similarly in Figure 2.3b, reaching a slightly higher fitness of around $u_{global} = 0.755$ at its peak at at $x = 14.81$.

An interesting aspect that may not be intuitive is that this fitness presents a function that does not have natural peak performance at one. The view on this is that since $h(x)$ at least approaches both bounds that the fitness function should do the same. This is not true. In the scenario presented in Figure 2.3, both utilities are pitted equally against each other and have their upper bounds at some distance equal in the power domain to their distance. If the battery drain utility is allowed to be more generous in terms of its goals and is shifted a fitness increase can be seen in Figure 2.4. Formally, this graph is the same as Figure 2.3a, however, $p_1 = 45/20$. This gives a result of $u_{global} = 0.894$ and $x = 23.815$

What this reflects is that Equation 2.8 shows us a true representation of trade-off between utilities. Scaling this fitness value to one can be done, however, provides no advantage in terms of radio performance. A global fitness maxima must be known a-priori to appropriately



(a) Utilities and Fitness of Equal Weight



(b) Utilities and Fitness of Skewed Weight

Figure 2.3: Weight skew

scale this function and is not required as all fitness functions are assumed static as the measure of usefulness each utility exhibits does not change. Variable and dynamic utility function are outside of the scope of this work, but provide an interesting area of research that may offer even more dynamics in terms of radio performance metrics.

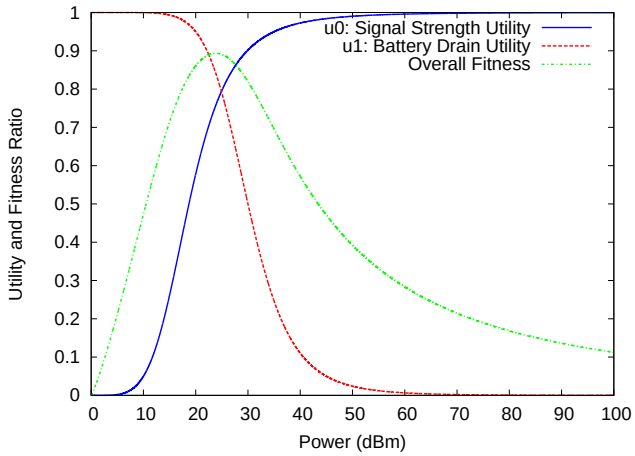


Figure 2.4: Generous Goal Fitness

Chapter 3

Parameter Management

3.1 Introduction

Traditionally, much CBR research has studied how a reasoning entity would use the information in the cases to classify the case under a specific classification description. Using this description it could then find other cases with similar attributes and use the information for these cases to be applied to the current situation [26]. In the CR domain, cognitive entities must have adequate domain knowledge in order to function properly as AI entities. Without at least knowing the possible variables of the system it is trying to optimize, it cannot adequately perform its duties. Knowing that a cognitive entity will at least have knowledge of the parameters involved, an adequate storage mechanism can be created to structure and index the information of the experiences it under goes. As cognitive systems become more and more complex, more parameters are introduced to the CE to interrupt different aspects of the systems. Systems like LTE and WiMAX can make use of a multitude of information to adequately optimize the adjustable parameters of the system [23, 1]. In this section, the case design used for the casebase is introduced, as well as describing the aspects of each element in the case. One way a casebase could be generally integrated with a reasoning entity is also reviewed. Finally the concept of similarity is introduced to give the reader an understanding of the actual organization of the data structure. Finally, the data structure is explained at length using the definitions created by this chapter.

3.2 Case Design

Within CBR, case design is critical to adequately representing useful information. As mentioned in Section 2.4 a case generally contains three critical pieces of information for use within the system; a problem, an action, and the respective results. There are numerous

Table 3.1: Case Representation

Case Contents	CBDT Attribute
$\hat{u}$	Problem
$\hat{\theta}$	Action
$\boldsymbol{u}$	Result
$\boldsymbol{\delta}$	Similarity
$\boldsymbol{\epsilon}$	Misc.

ways to represent the case's information and the organization of the casebase hinges on this design. Table 3.1 shows the organization of a case at a high level. What this case represents is a problem $\hat{u}$ that describes some set of utilities that have dropped below their designated goal or reached some undesirable levels. The lower limits of acceptable performance can defined by adequately defining p for Equation 2.3.

There are three classical representation of cases; object-oriented representation, predicate representation and relational representation [36]. Object-oriented representation compacts the data into a singular object, which may contain a set of feature vectors or contain a set of other objects. Similarity calculations can be difficult with this representation if the structure of these cases varies from case to case. Predicate representation presents cases purely in a rule based IF-THEN indexing system. Given that certain feature vectors or objects are available within the case, or they are a specific value, then they are selected based on some similarity gradient. This has some similarity to Rule-Based Reasoning and could be seen as an instance of it. Relational representation uses a search algorithm based on one or more feature vector instance values. By searching the casebase based on a range for a specific attribute, or a set of ranges for multiple attributes, a set of cases can be produced which match this description.

This approach uses a combination of these representations with two caveats, that the structure of the case is static and the calculation of similarity must be based on a static feature vector set, defined here as a *similarity dimension*. Each member of this similarity dimension must be describable by a minimum, a maximum and an interval gradient. This is represented in the case in Table 3.1 as $\boldsymbol{\delta} = \{\delta_0, \delta_1, ..., \delta_{\boldsymbol{\delta}-1}\}$.

Through the description of the processes involved here, we can view the flow of interaction between the engine and the casebase as a series of queries and responses. This is shown at a high level in Figure 3.1. Each part of the case is populated at a different point in this flow graph and demonstrates where in the reasoning process the casebase has control of the case and the reasoner has control over the contents of the case. This also demonstrates the separation between the two in that this work focuses mainly in the implementation of the casebase space, but is designed with its reasoning entity negotiations in mind.

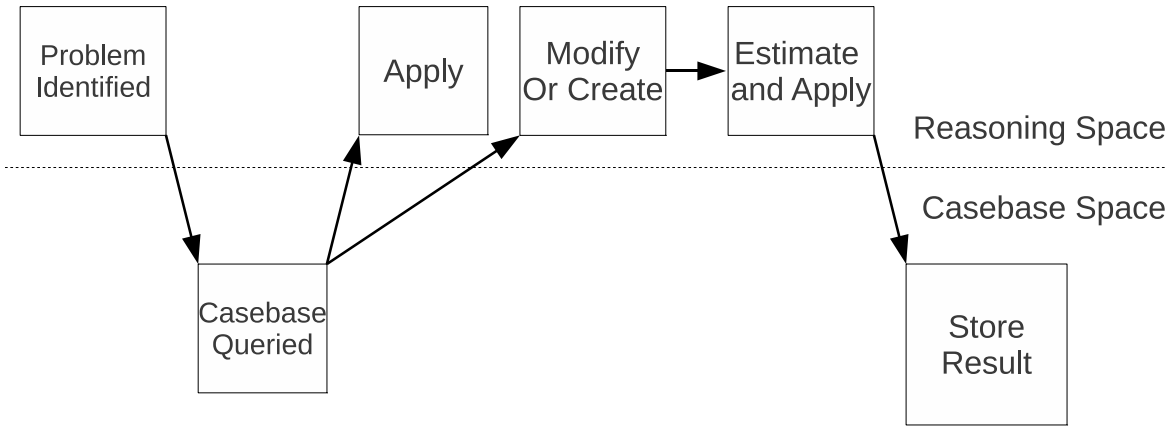


Figure 3.1: Interaction Flow Graph of Casebase and Reasoning Entity

3.2.1 Engine Invocation and Problem Description

At it's inception, CR research has been based around the OODA loop presented originally by Mitola [31]. This has generally been accepted as a reasonable loop for a CR to take, but is defined at a high level for abstraction. Different application implement this loop as is needed by their application [7, 28, 39, 38]. This work assumes some form of this loop is used in a general CR architecture and is discussed here to further the reader's understand of a clear problem definition within a case. The definition of the case is not described specifically to maintain abstraction, but is presented in a specific form in Chapter 4. From a high level point of view, this loop is considered invoked when something in the system changes, usually for the worse. The change in the system is assumed to have an impact on the utilities considered. Specifically, the designer should plan for possible problems that the radio can encounter, so that a measure of usefulness can be implemented to know exactly what the problem is. For example, in a single wireless link, throughput seems like an obvious utility to include in the reasoning engine. Signal strength, however, might be included to provide a means for the radio to know what is wrong with its transmission. While throughput degrades with signal strength, the CR cognition loop will not know how to adequately correct for this unless it knows its signal strength is empirically weak due to underestimating the path loss of the channel or simply not using enough power.

Thus we can assume that at least in some form of utility, we can identify the problem when a problem arises. To accurately describe this problem, a specific set of utilities, $\hat{u} \subseteq \boldsymbol{u}$ is used to describe the problem. This invokes the need for cognition and happens first in Figure 3.1.

There are other possibilities for the OODA loop (or some reasoning in general) to kick

in. The most apparent is the drive for the engine to optimize itself beyond satisfactory performance, even if there is no utility below acceptable measures. This is beyond the scope of this work and adds much more complexity for minimal return, which is reflected in the utility functions described in Section 2.6.1, given that the parameters for $\dot{x}$, q_p, q_0 and p are chosen for acceptable performance for each utility. Alternatively these goals can be chosen such that the fitness functions reflects the best possible values at their upper goal but with two drawbacks. Generally, each utility cannot be obtained at their maximum. Such is the case with signal strength and battery power consumption. Furthermore, the advantage of diminishing returns is lost and reflects only a steep gradient to reach its upper bound. Another reason why an OODA loop would be invoked might be because of a simple time interval. Regular initiation of a CE could provide benefit in analyzing ongoing metrics to predict possible deep fades if the channel's coherence time was small enough. The reasoning used in these simulations is time-interval based as it is all that is required to show acceptable performance of the casebase.

3.2.2 Case Elements

Now that the problem has been has been identified, the proper actions can be taken to correct its cause. The reasoning entity has controllable parameters available to them, which are assumed described in $\boldsymbol{\theta}$. To adequately correct for the problem utilities, $\hat{u}$, the engine will change some subset of parameters that it has control over, which is defined as $\hat{\theta} \subseteq \boldsymbol{\theta}$. How these changes are made are left up to the reasoning entity as the best changes depend on the application. This is the situation that a new case is created entirely. The reasoner should first query the casebase to find a set of similar cases related to its current problem. A set of similar cases can provide two different types of information to the reasoner, encompassing two types of reasoning as discussed in Section 2.4, instance-based reasoning and exemplar-based reasoning. The difference here is the problem of classifying cases is not an issue, merely a problem of properly indexing them. It is assumed that the designer has enough knowledge of the system to at least identify the parameters that are influence and have influence on the system. The relationship between these need not be defined for the casebase, as enough instances within a specific scenario will produce a good result or result set. This set can provide a general set of knowledge for some instance samples of similarity features, as well as ones that the reasoner can simply re-apply, with no higher level cognition involved. This would be, of course, reapplied if the reasoner deemed it performed well enough to be reapplied.

The result of the case is the resulting set of utilities after the application of $\hat{\theta}$ is performed. Within the CBR cycle, described in [3] and in [21] a case estimator is used to predict how well a case is going to perform. This is done in the reasoning space and may or may not be done given that the reasoner is going to simply reapply a previous case or alternatively, modify or create a new case altogether. This distinction is illustrated in Figure 3.1. The estimation that is performed can provide the reasoner with useful information as it can compare its

generated solution to past solutions to ensure the best one is chosen. This is where a proper exploitation vs. exploration trade-off algorithm would best be suited between a reasoner and a casebase, such as in [44].

It should be noted that $\boldsymbol{\epsilon}$ provides the case with any other relevant information that may not necessarily be included in either of the vectors mentioned Section 3.2.2 or Section 3.2.1. This includes non-relevant, yet non-trivial information the reasoner may use for other high level reasoning including possible time stamp information, case identification numbers and policy considerations (what policies were in effect during this case's operation). Specific policy instances in effect may be used here or within the similarity dimension as it places constraints on what solutions the reasoner may be allowed to use.

3.3 Similarity

The final aspect of the case representation is $\boldsymbol{\delta} = \{\delta_0, \delta_1, ..., \delta_{N_{\boldsymbol{\delta}}-1}\}$, a similarity dimension vector of $N_{\boldsymbol{\delta}}$ entries. This dimension of the case is the cornerstone of this casebase's indexing scheme as it provides the relative similarity details of the case. This structure is assumed to be specific information, mapped from the information of the case ($\hat{u}$, $\hat{\theta}$, $\boldsymbol{u}$ or $\boldsymbol{\epsilon}$) into the similarity dimension. This deviates from CBR literature in considering the information on which to index the case is wholly separate from the case contents itself. While there is a separate vector of information than the rest of the case, the contents of $\boldsymbol{\delta}$ are still generated from the case information. This is based on the fact that not all information in the case is relevant to describing the aspects that must be similar in order for a case to provide useful information to the reasoning engine. By designing the case such that some or all information is relevant to describe the similar aspects of the situation in which the case was generated, a reasoner can know in which problem domain it is before even assessing what it should do.

3.3.1 Concept Definition

Similarity provides a way for a case-based reasoner to adequately quantize how the relevance of the information of cases in the casebase to a particular case that is currently trying to be solved. In traditional CBR research, similarity has taken on two forms; *surface similarity* and *structural similarity* [30]. Structural similarity is based in the fact that each case has a different set of feature vectors, structures and objects contained within itself. The exact structure of the case can vary from case to case and can be computationally complex when considered for an entire casebase. Surface similarity is calculated using the features of the case comparatively, which represents the traditional approach of similarity for static feature vector cases. For two cases, e_q and e_p of n feature vectors, and a similarity weight vector of $w_i \forall i = 0, 1, ..., n$, weighted Euclidean distance is shown in Equation 3.1 has been a mainstay in CBR distance calculation [36]. When all weights are equal to one, this degrades into a

measure of Euclidean distance. d^1_{pq}.

$$d^{(w)}_{pq} = d^{(w)}(e_p, e_q) = \left[\sum_{j=1}^{n} w_j^2 (x_{pj} - x_{qj})^2\right]^{1/2} \tag{3.1}$$

Equation 3.1 represents the heart of a k-Nearest Neighbor (k-NN) algorithm. By comparing all cases to the current case, a k-NN algorithm can be used to find the absolute most relevant case or cases by classifying which problem set the case belongs to. Some implementations in CR literature have used a ranking system to derive the highest set of possible cases using this calculation [6]. I used CBR differently in this classification in that generally, CBR first identifies the class of case via some method (e.g. - k-NN) and uses this classification to find which cases performed the best amongst those in that class. In [6] a casebase was used to store the information and used Equation 3.1 to identify similar cases. This is not necessarily a classification problem, which is what most classical CBR research attempts to solve, but does provide a studied quantification of how to compare the relevance of two cases.

3.3.2 Advantages and Disadvantages

This weighted Euclidean distance provides advantages in that it can find the closest neighbors near some given point in Euclidean space and factors in weights to make some features more important than others. This same weighting scheme may be used in the case definition in Table 3.1, but there are two negative drawbacks to using this kind of calculation. First, it is unwieldy for large casebases that contain many cases. The access time is always relative to the computational power behind it, however, with CR embedded designs are prevalent and this computation power should be considered limited. As a casebase's size increases, the access time of a list data structure increases linearly as well when all elements of the list have to be visited. This is on the order of $\mathcal{O}(n)$, where n is the number of cases stored. Secondly, using this kind of similarity calculation presents problems like that seen in Equation 2.5. Because Equation 3.1 is based in the real number space, $\mathbb{R}$, features that may range from several orders of magnitude larger than others may overshadow smaller, yet equally important differences. For example, if some utility was used as a similarity metric in $\boldsymbol{\delta}$ and is bounded as defined in [15] as $g(x) \in (0, 1)$, then some other similarity measure such as packet size which may range from just a few bytes to several thousand could easily overshadow this. Consider a case with two feature vectors x_0 and x_1 which represent fitness and packet size respectively, with weights $w_0 = 0.5$ and $w_1 = 0.5$. When the Euclidean distance is plotted, Figure 3.2 shows this disadvantage with an almost linear scale in packet size's difference, while fitness difference has almost no impact.

Even when the weights are heavily skewed at $w_0 = 0.99$ and $w_1 = 0.01$, the same trend is still prevalent in Figure 3.3

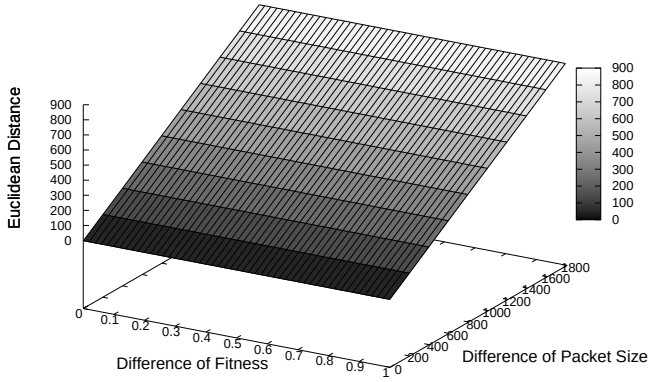


Figure 3.2: Euclidean Distance Example

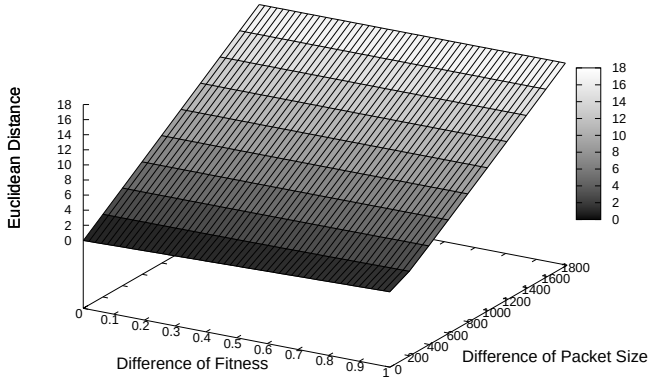


Figure 3.3: Weighted Euclidean Distance Example

The reason this does not work for our application and why it has worked in so many other domains is due to the range of dimensions we are working with as well as an inherent difference in the problem that traditional CBR usage has been applied to. The dimensions used in traditional CBR usage assumes that the general scale of each dimension is relative to each other in terms of the differences presented. Using the weights to scale these is inefficient and a misuse of what the weights themselves are trying to accomplish for the resulting metric. Weights are used to describe the importance of the metric in terms of similarity, not to appropriately scale the curve of its feature vector. Other work has created scale invariant techniques such as the Mahalanobis distance [29]. This work grows out of the original CBR problem of classification and is not targeted to simply identifying cases on a single vector. The problem still remains, however, is that as a casebase grows, its access time grows linearly as well if every case in the casebase needs to be compared against the

query case.

3.4 Organization by Similarity

There are two main ways to consider similarity in a CBR system. One is structural similarity, which is the basis of a classification problem [45]. By successfully grouping similar cases, a reasoner can use this knowledge to identify the domain the current case lies within and identify the different experiences it has gone through within this domain to find information relevant to a good solution. This deals mainly with what attribute-value pairs are available within the current case and the set of cases that had the same pair(s) available to it as well [36]. Once this domain is identified, there are even more techniques that use this information to further refine the information those cases provide it. The other way to obtain a similarity is to simply calculate it using a distance metric between two cases. For the CR domain, the problem with these two measures are described in Section 3.3.

Using the case definition described in Table 3.1, we can now describe how a data structure can be used to organize this information. Recall the similarity dimension vector, $\boldsymbol{\delta}$. Instead of assuming that each member of $\boldsymbol{\delta}$ is bounded on the real number line, in CR applications we can statically define the parameters required by the CR to maintain the goals defined in its objectives. Not only the utilities, but any parameter contained within the case can be translated into this similarity dimension by understanding the properties of the dimension itself. For example, when considering an abstract piece of information that would be useful in a single wireless link, dimensions of path loss, distance, fading, extreme temperature would all contribute in terms of a similar situation that the radio would currently be experiencing. In a networked aspect, a base station would want to store similar data but relate it to a geographic location to identify an area of unusually bad reception for its users. It could also associate data with the user itself. The number of users, network load, and dynamic network configurations would all contribute to describing the situation a CR is in when it encounters a problem. As CR technology moves up the network stack in terms of what it has control over and must consider the case dimensions can become quite large. As dimensions are added to a case's description of its situation, the possible quantifiable situations grow exponentially. Consider a simple example of a similarity dimension as 4 vectors, each able to take on one of 10 possible values. This results in $10^4 = 10000$ possible combinations for a simple scenario. Allowing the dimension to take on any continuous value rapidly increases complexity. Furthermore, adding more features to $\boldsymbol{\delta}$ exponentially increases these possibilities at an alarming rate. With such a large combination of solution space the reasoning of a CE is tasked with the large task of finding optimal solutions for any situation. In parallel with this task, managing such a large number of possibilities can be daunting if not designed properly.

In order to manage such a large solution space, consider each dimension of $\boldsymbol{\delta}$. For a CR the dimensions of these vectors, $\delta_i \forall i = 0, 1, ..., N_{\boldsymbol{\delta}} - 1$ can be described in terms of a minimum, a

maximum, and a granularity. The constraint for this description is that for some dimension i, $\max \delta_i - \min \delta_i \geq 2g_i$ to maintain at least more than one possible interval. The minimum and maximum need not be the absolute of the metric but just a practical measure of what the CR will experience in its life cycle. For example if temperature were a dimension on this similarity vector the absolute minimum theoretical value of 0 Kelvin does not need to be considered the minimum due to the fact that no radio will ever experience this temperature. There are reasons that all of these ranges would be included, one being for the benefit of the reasoning entity. Radios that communicate from the dead cold of space experience different temperature ranges than those along the equator and each range should include a practical set of values.

The grain of each dimension is defined here as $\boldsymbol{g} = \{g_0, g_1, ..., g_{N_\delta - 1}\}$. $\boldsymbol{g}$ is meant to provide an interval of value that the CR reasoning engine can consider points in that interval to be similar. This will vary from dimension to dimension but the only requirement of the range is that it is even divisible by the difference of extremes on δ_i, viz

$$(\max \delta_i - \min \delta_i) \mod g_i = 0 \, \forall \, i = 0, 1, ..., N_{\boldsymbol{\delta}} - 1 \tag{3.2}$$

The constraint presented in Equation 3.2 creates intervals on the dimensions values space and is defined here as *similarity buckets*. To demonstrate this concept consider an example where $\boldsymbol{\delta}$ contains two dimensions, transmission gain and receiver gain, δ_0 and δ_1 respectively. For the sake of simplicity assume the following configuration in Table 3.2.

δ_i	Max	Min	Grain
δ_0 (dBm)	24	16	2
δ_1 (dBm)	34	26	2

Table 3.2: $\boldsymbol{\delta}$ Configuration for δ_0 and δ_1

From Table 3.2 the grain of each dimension allows for a 2 dBm swing to be considered similar. This may be large for a communication system, especially if it was trying to optimize at very fine levels but is presented coarsely here for similarity. If the CR had operated in three different scenarios using different, but similar values each time we could get possible values as below, in Table 3.3.

Case Number	$\hat{\delta_0}$	$\hat{\delta_1}$
1	21.4	30.7
2	20.6	31.5
3	20.8	31.3

Table 3.3: Sample Cases for a Simple $\boldsymbol{\delta}$

Plotting these samples on a 2-dimension plot would give us cases that were stored together as shown in Figure 3.4.

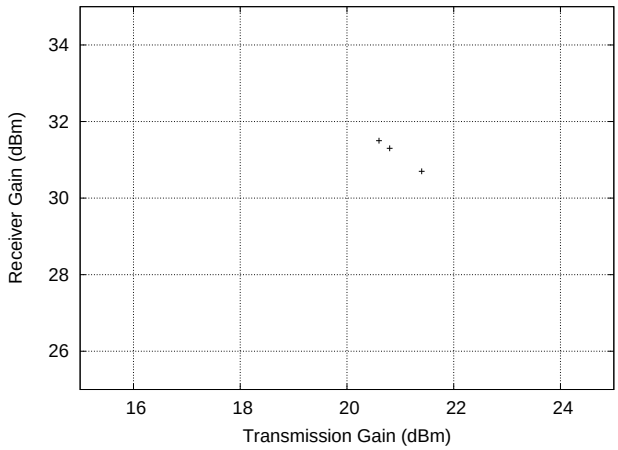


Figure 3.4: 2D Plot of Case Samples

This demonstrates the bucketing feature of using g_i for each respective dimension of $\boldsymbol{\delta}$. By partitioning the dimension into what can be considered similar values of each dimension, we can store cases into the same $N_{\boldsymbol{\delta}}$ dimensional hypercube. We can see this organization at a high level in an abstract data structure as shown in Figure 3.5. Note that even though these cases are stored in the same bucket, no specific information is lost in the actual case.

Considering that each dimension of $\boldsymbol{\delta}$ is represented by an array of pointers that represent a set of slots that classify every possible value of δ_i, we can see the structure of two levels of dimensions in Figure 3.5. Each node of this structure contains a single array of possible dimension buckets and every value in the array is a pointer to either NULL or has value. When an array element has value, it can either be a pointer to another node in the next level down of the tree, which means another array of possible values, or a pointer to the container that holds each cases' information individually. The former is commonly referred to as a node or arch and the latter a leaf or terminal node in a treed data structure [12]. $N_{\boldsymbol{\delta}}$ dictates how many levels of indirection this structure can have.

These levels of nodes when used together form a m-way tree structure. An m-way tree can be seen in graph theory as a rooted tree structure in which each child can only have at most m children [12]. The difference in this data structure is that at each level, each node can contain at most the number of buckets as defined by the parameters of the dimension. The number of children, or buckets, is calculated via Equation 3.3. Due to the constraint on these descriptors in Equation 3.2, the number of children is always a positive integer value.

$$NumBuckets = \frac{\max \delta_i - \min \delta_i}{g_i} \tag{3.3}$$

Using Equation 3.3 we can see that the maximum number of buckets available for the example shown in Figure 3.4 is $NumBuckets = 4$. These nodes build up a tree structure with the

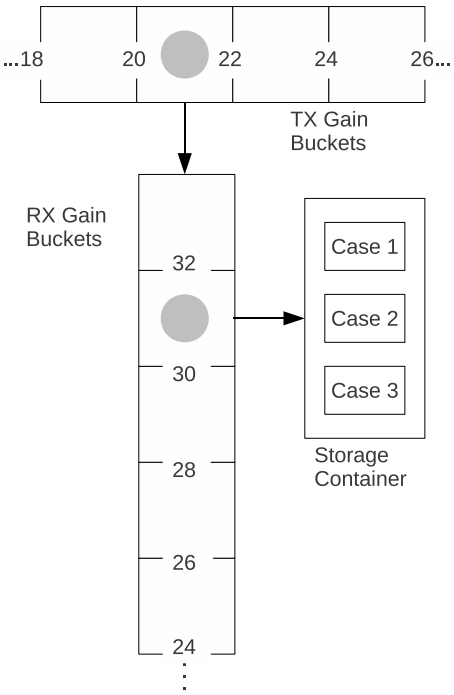


Figure 3.5: Data Structure Organization of Case Samples

first dimension, δ_0 being the root node and containing pointers to possible children. For a singular case, we can see that for some $N_{\boldsymbol{\delta}}$ dimension similarity vector, the levels of the tree correspond to each dimension in Figure 3.6.

Each one of the cases is organized via their exact value of each dimension. Depending on the value of that dimension, an index is calculated for which interval on the dimension that particular value would be found. This index calculation provides an easy way to group similar values in a bucketed fashion. In order to calculate the index offset for the insertion and accessing algorithm Equation 3.4 is used to find the proper index value for the precise dimension value, $\hat{\delta}_i$.

$$i(\hat{\delta}_i; g_i) = \left\lfloor \frac{\hat{\delta}_i - \min \delta_i}{g_i} \right\rfloor \tag{3.4}$$

Tree structures have been seen before within the context of CBR [45, 41]. These trees have been used for classification and are similar to the approach taken in this research but has several differences. In [45] the most important difference is the use of a binary tree. By dynamically creating the buckets on arbitrary bounds, the case attribute space is split when two sets of case(s) needed to be separately distinguished. This approach assumes that enough knowledge about the system is available to predefine the situation. [41] also uses

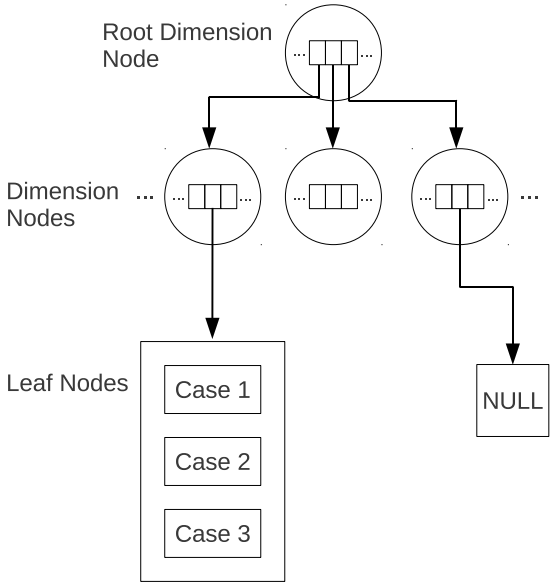


Figure 3.6: Tree Data Structure

trees to separate cases into qualitative and quantitative divisions, but also creates indexes dynamically. For our application a simpler approach can be used as long as each dimension of similarity can be understood in context of a minimum, maximum and extreme. CBR enables CR to store and index practical information from past experiences. Classification of these cases in this approach are held to simple strict bounds that are defined before implementation.

The advantage of this algorithm is based in it's iteration through $\boldsymbol{\delta}$. For a case that is being queried to the casebase, the underlying data structure needs only to traverse and calculate the correct index for each dimension of $\boldsymbol{\delta}$. This is seen in the access algorithm presented in Algorithm 2. The insertion and access algorithm are similar except that the insertion algorithm, in Algorithm 1, new $DimensionNode$ and new $LeafNode$s must be created in the appropriate place. Algorithm 2 simply assumes that no cases have matched the query case. This is opposed to traversing the entire casebase and iteratively comparing each case in the casebase to the current query case. Theoretically these computations orders differ from $\mathcal{O}(n)$ to $\mathcal{O}(1)$.

Algorithm 1 Insertion Algorithm using Case e_i

$Node \leftarrow RootNode$
$Q \leftarrow e_i$ {Gather similarity vector from query case}
$\{\delta_0, \delta_1, ..., \delta_{N_{\boldsymbol{\delta}}-1}\} \leftarrow \boldsymbol{\delta}$ {Dimension definitions}
$\{g_0, g_1, ..., g_{N_{\boldsymbol{\delta}}-1}\} \leftarrow \boldsymbol{g}$ {Grain definitions}
$index \leftarrow 0$
for $i = 0$ to $N_{\boldsymbol{\delta}} - 1$ **do**
  if $!Node$ **then**
    $Node \leftarrow$ new $DimensionNode$
  end if
  $index \leftarrow \lfloor (Q_i - \min \delta_i)/g_i \rfloor$
  $Node \leftarrow Node.Children[index]$
end for
if $!Node$ **then**
  $Node \leftarrow$ new $LeafNode$
end if
$Node.Cases.pushBack(e_i)$
return

Algorithm 2 Query Access Algorithm using Case e_q

$Node \leftarrow RootNode$
if $!Node$ **then**
  return $NULL$
end if
$Q \leftarrow e_q$ {Gather similarity vector from query case}
$\{\delta_0, \delta_1, ..., \delta_{N_{\boldsymbol{\delta}}-1}\} \leftarrow \boldsymbol{\delta}$ {Dimension definitions}
$\{g_0, g_1, ..., g_{N_{\boldsymbol{\delta}}-1}\} \leftarrow \boldsymbol{g}$ {Grain definitions}
$index \leftarrow 0$
for $i = 0$ to $N_{\boldsymbol{\delta}} - 1$ **do**
  $index \leftarrow \lfloor (Q_i - \min \delta_i)/g_i \rfloor$
  $Node \leftarrow Node.Children[index]$
  if $!Node$ **then**
    return $NULL$
  end if
end for
return $Node.Cases$

Chapter 4

Simulation Results

4.1 Introduction

Section 3 explains the structure and organization of cases for a CBR in use by a CR application. By appropriately defining the general framework this data structure can be used in, values can be applied and performance can be verified. This chapter aims to simulate this implementation under several scenarios to analyze the performance of the structure. This chapter aims to verify the validity of the claims made in terms of a performance improvement over traditionally accepted methods of similarity calculation in CBR applications. By first showing the implementation in terms of extreme cases, for example 100 dimensions of similarity contained within $\boldsymbol{\delta}$, the data structure is still expected to perform relatively quickly, especially as the casebase grows in terms of number of cases. Secondly, the structure is combined with a simple reasoner as a case generation mechanism. Even with a random population of cases throughout the simulation two things are prevalent; access time is greatly reduced as compared to traditional CBR approaches and Euclidean distance assumptions of the cases returned are generally preserved even though there is some loss in precision when considering a similarity measurement over the entire casebase using an calculation like Equation 3.1.

Each simulation was performed using open source C++ code and a light-weight digital signal processing (DSP) library known as *Liquid-DSP* [14]. *Liquid-DSP* provides a useful set of functions that aide in simulating wireless links. Using theoretical approaches, it enables the simulator to perform a wide variety of wireless testing, supporting a multitude of popular wireless techniques including automatic gain control, equalization, forward error correction schemes, examples to demonstrate the usage of each, as well as tests to prove their functionality. The basis of this simulation was forged out of an OFDM link example between two nodes. OFDM is chosen for its prevalence in current Long Term Evolution (LTE) technologies [1]. The static simulation parameters are shown in Table 4.2. These

must be defined in the scenario to determine the resulting data that is presented in Section 4.3.2.

4.2 Testing Scalability

Part of the contribution of this thesis is to design a data structure well suited to handling many cases within a CBR framework. Restrictions are not placed on the cases in terms of information they can store to describe the situation the case was created for. In order to test the various scenarios that this data structure can handle, the practical storage capacity is verified to demonstrate the preservation of access time through a large information set. This is especially important as the assumption is traditional CBR approaches are ill-suited to handle more complex scenarios that CR technology will encounter.

4.2.1 Scenario

Using the following metrics in Table 4.1, a simulation is created to insert a variable number of cases into a casebase followed by a retrieval of all cases. Before the casebase can be created, $\boldsymbol{\delta}$ must be design as it dictates the height of the similarity tree. By creating a variable number of cases, inserting them into the casebase and then testing the retrieval time of random case access, we can observe the performance of the similarity tree against a traditional list approach.

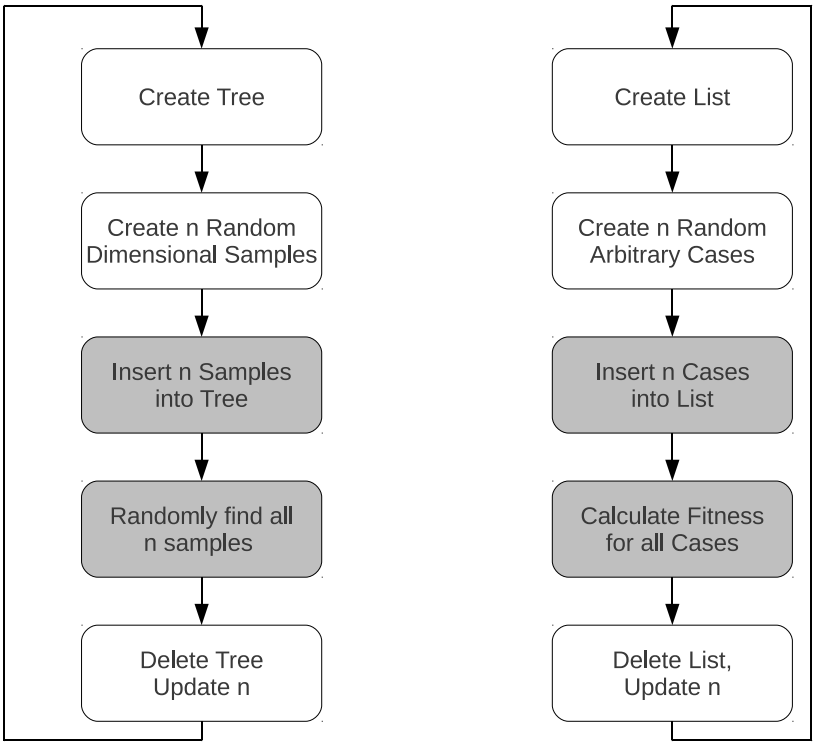
Metric	Value
$N_{\boldsymbol{\delta}}$	100
$\max \delta_i \ \forall i = 0, 1, ..., N_{\boldsymbol{\delta}} - 1$	100
$\min \delta_i \ \forall i = 0, 1, ..., N_{\boldsymbol{\delta}} - 1$	0
$g_i \ \forall i = 0, 1, ..., N_{\boldsymbol{\delta}} - 1$	1

Table 4.1: Configuration for Similarity Tree Simulation

Each data structure was initialized to empty storage with no contents. This can make a difference if reallocation is used in the casebase's data structure as reallocation can significantly slow down insertion time, but also increase access time. An example of this kind of dynamic storage is STL's `vector` implementation [11]. A simple experiment was conducted using a variable number of cases. For each simulation event loop, n represented the number of cases inserted and then searched for. Figure 4.1 shows this loop for both kinds of simulations tested. Each gray event block in Figure 4.1 shows which events are measured in terms of timing. Insertion time of both data structures is measured as well as access time.

How each data structure is tested varies slightly to better simulate how it would be used in a practical sense. Figure 4.1b shows the event process for testing the linear data structure.

First an empty list is created, then some n number of cases is inserted into the casebase list. Once the list is populated, an iterator traverses the list performing a similarity calculation over $N_{\boldsymbol{\delta}}$ dimensions for each case. In this scenario, traditional reasoning entities must calculate the distance between the query case and every case in the casebase. Some kind of ranking system would be used to rank the top most similar cases. Inefficient or overly extraneous ranking systems are not taken into account here and are assumed to take a relatively small amount of time.



(a) Tree Simulation (b) List Simulation

Figure 4.1: Event Flow for Simulations

A similar method was used to test the similarity tree. First, an empty tree was constructed and used to insert n random dimensional samples. Because the tree itself is configured on the similarity dimension of the cases, a single arbitrary case was used to insert into the tree using a different similarity dimension. The similarity dimension was varied to simulate an evenly diverse case set. This shows how the case representation abstracts away from a tight dependence on the contents of the case. Each case is used to create a path through the tree by populating the dimensional vector. Once all cases are inserted, each case is then searched

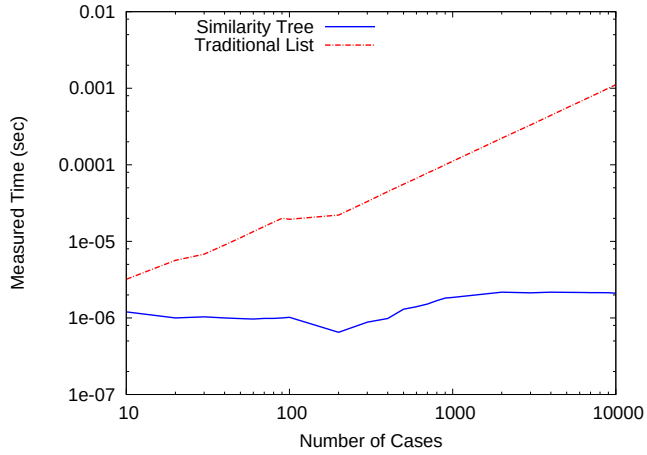
for in the tree. The average time required to access each case is recorded. Once all cases are found, the tree is deleted and n updated.

4.2.2 Results

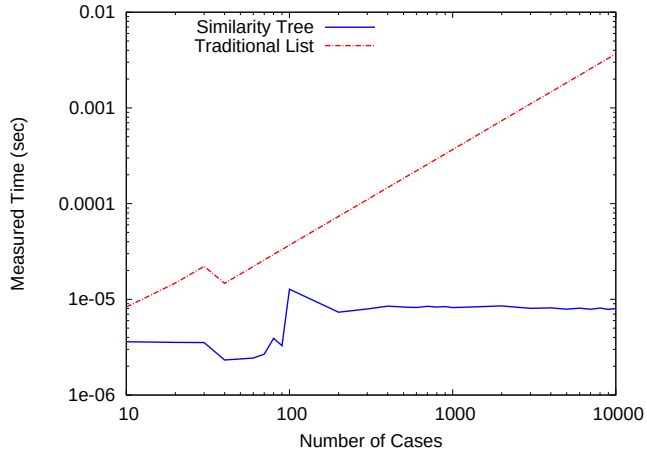
Using the event flows described above, performance metrics were collected to verify the performance gains of this the bucketed similarity tree approach over a traditional list methodology. The initial simulations show that a similarity based tree approach provides a much faster access time than a traditional list approach. This is especially true as the casebase grows in size. Figure 4.2a shows the average access time for each data structure which reflects a much faster access time for the similarity tree when compared to a traditional list approach. Figure 4.2a represents these access times at $N_{\boldsymbol{\delta}} = 25$.

By increasing the number of similarity dimensions in the case, the similarity tree requires more processing time. This does not add additional order complexity to the list data structure as it uses the same information to calculate distance, however, Figure 4.2b shows that some additional computation power is required to compute more dimensions for each case. Specifically Figure 4.2b represents the respective access times when $N_{\boldsymbol{\delta}} = 100$. Figure 4.2b shows us that even with an increase in the number of similarity dimensions, the buckets similarity tree out-performs the linear data structure.

In these initial tests we can see that the similarity tree provides more advantages in terms of access times for larger casebase sizes. Bucketing the parameters of $\boldsymbol{\delta}$ provides us with a generalized guideline to quantize the scenario each case was created in. While the access time is improved, it is important to verify that this data structure creates a useful way to organize information as well as a fast was to access that information.



(a) Average Access Time for $N_{\boldsymbol{\delta}} = 25$



(b) Average Access Time for $N_{\boldsymbol{\delta}} = 100$

Figure 4.2: Simulation Access Times

4.3 Bucket Convergence

In Section 4.2 I showed how the similarity tree compares to traditional methods of organization when it came to scalability and access time. In this section the casebase is used on-line, with a simple reasoning entity to create and store information that the reasoner can use to classify the kind of channel it is operating in. The scenario is not necessarily how a real reasoner would use the casebase nor a representational of a traditional reasoning mechanism, but instead is used to show solution convergence within the dimensional problem sets. Because I have changed the way information is quantified as similar, it is important to verify that the information provides useful information relevant to the problem at hand.

When each bucket is created via $\boldsymbol{\delta}$, a situation is created that inherently defines the general

vicinity of the problem in the problem space. Once the problem is identified, there exist an optimization problem that the reasoning entity must perform to find an optimal solution to this problem. In this scenario, the convergence time of the reasoning engine is not considered but the fact that it converges even if from random trials is important to ensure that the underlying data structure provides a useful organization of information for the cognitive engine.

4.3.1 Scenarios

In this scenario the casebase similarity tree is tested under a simple reasoning entity. This entity creates cases at random to not only try and find an optimal solutions to a static environment, but also to build information about the specific problem in terms of whether a solution is good or bad. Using *Liquid-DSP* and the similarity tree tested in Section 4.2, a new test was created for a simple wireless environment. This parameters of this environment is extended later, but is tested here to ensure convergence of the solution sets that are retrieved by the casebase tree. The reasoning entity in this scenario is allowed to generate new cases randomly and test for similar ones in the tree. The reasoner queries the similarity tree and retrieves a set, if any, that are deemed similar in terms of the definitions presented in $\boldsymbol{\delta}$. Being a simple reasoner, amongst the cases presented by the similarity tree, the one with the highest fitness is selected. We examine the parameters of the case to identify trends within the buckets defined, specifically transmission power. The representation of the static parameters are given in Table 4.2. Modulation, forward error correction, and packet length will become variable in later simulations but kept static here.

Table 4.2: Static Transmission Parameters

Configuration Element	Value
Frequency	910 MHz
Modulation	QAM, 8 b/s
Forward Error Correction	None
Packet Length	1024
No. of Sub-carriers	64
Cyclic Prefix Length	16 bits
No. of S0 Training Symbols	3 symbols
Cyclic Redundancy Check	16 bits
Free Space Path Loss	20.0 dBm
Noise Floor	-90 dBm

To demonstrate the casebase's ability to appropriately store and reach a steady state, only transmission gain was varied randomly between it's possible values. Using power as the only similarity dimension and giving it such a large grain allowed for two independent scenarios to be created; one that operated in a transmission gain higher than -40 dB and the other

when operating at lower than -40 dB. By allowing the reasoner to choose cases that simply contained a higher fitness, a basic reasoner is created. Naturally, the highest fitness possible provides the best solution to the simulated wireless transmission. By biasing the weights to choose transmission gains that conserved battery life, but preserved a small amount of throughput, we can observe that even though the casebase is being populated randomly, at some point convergence occurs. Table 4.3 shows the configuration of this experiment in terms of the parameters described in Section 2.5.

Table 4.3: Simulation Parameter Description

	Metric	Value Description
$\boldsymbol{\theta}$	Transmission Gain	-90 thru 10 dBm
$\boldsymbol{\beta}$	Throughput	b/s
$\boldsymbol{\phi}$	Path Loss	-40 thru +10 dB

Table 4.4: $\boldsymbol{\delta}$ Description

Similarity Dimension	min	max	Grain
Transmission Gain	-90 dBm	10 dBm	50 dBm

Two other simulations were performed with different weights. By exchanging the fitness weights in terms of signal strength and power consumption, we are able to observe the production of cases that converge to powers that are as low as possible in each of the different buckets. The last simulation evenly distributed the weights to show power convergence within each bucket for some arbitrarily fit transmission gain.

Table 4.5: Signal Strength Emphasized Utility Description

Utility	$\dot{\beta}$	p	Weight
Power Consumption	-65 dBm	-65 / -15 dBm	0.01
Signal Strength	-15 dBm	-15 / -65 dBm	0.97
Throughput	800 kb/s	100 / 800 kb/s	0.01
Robustness	13	13 / 2 Coding	0.01

Table 4.6: Battery Conservative Utility Description

Utility	$\dot{\beta}$	p	Weight
Power Consumption	-65 dBm	-65 / -15 dBm	0.97
Signal Strength	-15 dBm	-15 / -65 dBm	0.01
Throughput	800 kb/s	100 / 800 kb/s	0.01
Robustness	13	13 / 2 Coding	0.01

Table 4.7: Evenly Distributed Utility Description

Utility	$\dot{\beta}$	p	Weight
Power Consumption	-65 dBm	-65 / -15 dBm	0.25
Signal Strength	-15 dBm	-15 / -65 dBm	0.25
Throughput	800 kb/s	100 / 800 kb/s	0.25
Robustness	13	13 / 2 Coding	0.25

4.3.2 Results

The reasoner is allowed to create cases and apply them to the simulation. As the reasoner creates cases, it stores them in the casebase. The casebase is queried before every case creation to find similar cases. As is expected, the casebase eventually produces cases that converge to a near optimal transmission gain value. In the first simulation we observe that by maximizing the transmission gain in each individual scenario, a better fitness is returned in Figure 4.3. Figure 4.3 shows the convergence of the casebase to return cases with higher overall power. As opposed to lower powers which eventually give no throughput, higher SNRs are not bounded by the same thing. In a real scenario, given that the transmitter had enough power the receiver would be saturated with power and would again be unable to receive packets.

In the next scenario, battery power was held over higher regards than any other entry in $\boldsymbol{\theta}$ that the radio had control over, as shown in Table 4.6. This contributed to the lower powers having more fitness in each bucket. For each bucket, (-90 thru -40 dBm and -40 thru 10 dBm) the lower the sampled power, the higher the fitness. This is due to the battery conservation bias shown in Figure 4.5. It is interesting to note that even though lower power is weighted for a higher fitness, the system does not use the lowest power possible. This is most likely due to the fitness incorporating throughput as well. Given that the SNR of the received signal was low enough to not synchronize for packet reception, the throughput dropped to zero, reducing the overall fitness to zero, preventing a power of zero being held at the highest regard.

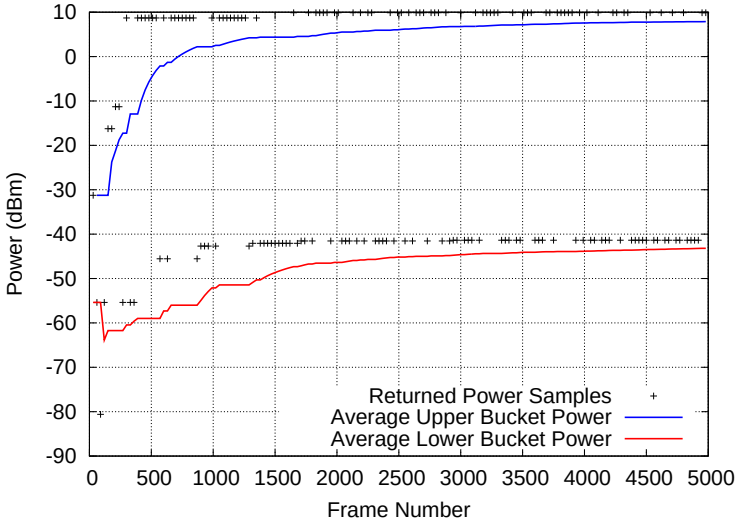


Figure 4.3: Liberal Power Decisions

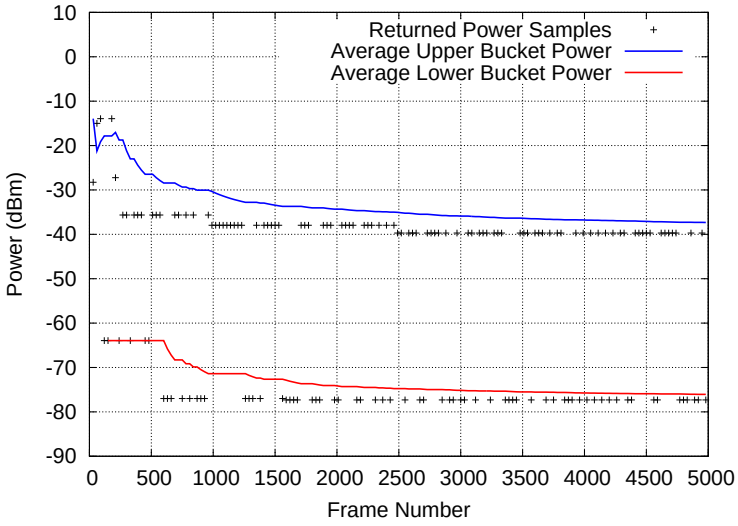


Figure 4.4: Conservative Power Decisions

While biasing shows us that a convergence to an optimal fitness is helpful, it is not very practical. In a more realistic scenario, the weights of the utilities would be on a much more even keel. The configuration in Table 4.7 shows an even weight distribution amongst each utility. While the weights used in each scenario sum to one evenly, this is not a constraint of the utility functions. A summation of one simply gives the reader a better frame of reference for the relative weight distribution.

Figure 4.5 shows us a convergence of an evenly distributed utility weight. The casebase confirms that the highest utility power is given around -48 dBm, which is near the middle

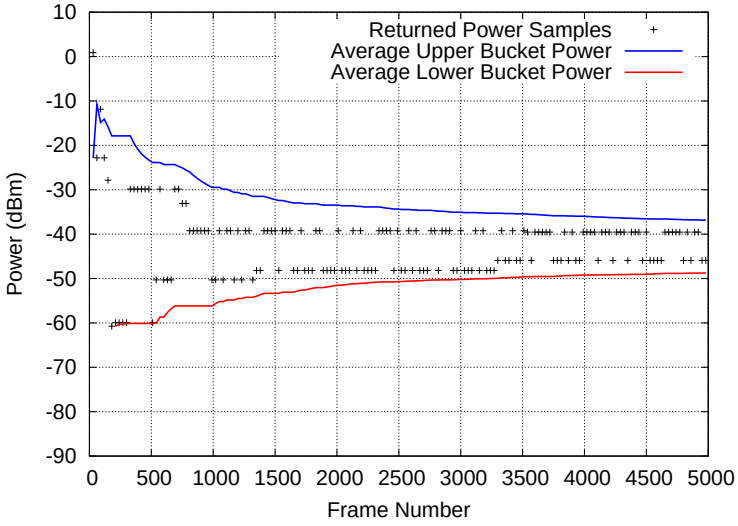


Figure 4.5: Evenly Weighted Power Decisions

Table 4.8: Updated $\boldsymbol{\delta}$ for Finer Grain

Similarity Dimension	min	max	Grain
Transmission Gain	-90 dBm	10 dBm	10 dBm

of the possible values the transmitter is able to use. This is slightly conservative of battery power however, and can be typical of wireless systems that once a given SNR is reached, the probability of synchronization is relatively high with a diminishing return when increasing power. This makes sense in that the system would tend to slightly more conservative power cases as throughput is not noticeably affected between the middle of -40 dBm and the settling point of -48 dBm.

If the number of scenarios is increased by increasing the granularity of the power, shown in Table 4.8, we are presented with 10 scenarios in which the reasoner can specifically classify the problem with. While these scenarios are simple and artificial they still demonstrate two things; the casebase's able to accurately split up the problem space by identifying the problem as it happens and it is able to converge within the defined situation given that the situation has been seen enough in the past to have enough information to solve it.

The results in Figure 4.6 show that the casebase eventually contains the required information to produce the best fit for that power setting. It is interesting to note within this results that two global convergences appear. Most buckets try and optimizing their power to a transmission gain settling around what is seen in Figure 4.5, however, a contending fitness is produced within the scenario of the -80 to -70 dB bucket. This set of cases shows similar if not better improvement around -80 dBm. This demonstrates the trade-off of throughput and battery consumption vs. throughput and signal strength. This can be seen as two

local maximum in the entirety of the system, however, in real applications these would be considered local in their own bucket space.

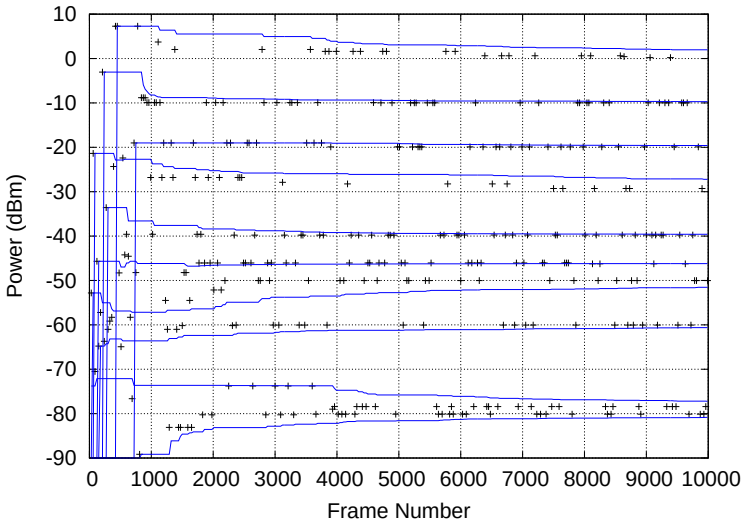


Figure 4.6: Finer Grain Bucket Splitting

4.4 Dynamic Environments

In Section 4.3 I demonstrated the ability of the casebase to at least converge on a certain fitness within the defined similarity buckets for a simple case. In this section the simulations work with a time-variant environment to demonstrate the bucketed similarity tree's usage. The first of which is that with a different $\boldsymbol{\delta}$, a similar situation is redefined. By adding in a new dimension that varies over time, a true measure of stochasticity is introduced to the reasoner in that its own controllable settings, specifically described in $\boldsymbol{\theta}$, do not dictate the actions it will make next. It is entirely possible for real world situations to incorporate the information from $\boldsymbol{\theta}$ but should only be included if it is useful for the reasoner.

4.4.1 Scenarios

A scenario similar to Section 4.3 with a few key differences. First, the weight of the utilities is redefined for more realistic scenario, which places emphasis on throughput and battery conservation. Signal strength is included to provide a trade-off between the two utilities that transmission gain can offer. It can be said that signal strength provides diminishing returns and can be represented fairly well through the measure of throughput. Robustness is not given great consideration as this simulation is designed to demonstrate the convergence of optimal power levels in a time varying channel. $\boldsymbol{\delta}$ is defined in Table 4.9 which represents the similarity buckets of path loss in the channel.

Table 4.9: $\boldsymbol{\delta}$ for Time-Varying Channel

Similarity Dimension	min	max	Grain
Path Loss	-10 dBm	40 dBm	1 dBm

The utilities used for this scenario are defined in Table 4.10. This places greater emphasis on throughput and battery conservation over signal strength and robustness. Signal strength is downsized due to the diminishing returns it gives once a certain SNR threshold is reached. Robustness is not considered yet here as it is left static of the duration of this simulation. During the simulation, the path loss of the system is varied in a square wave. The square wave oscillated between 0 dB and 30 dB of loss in the system over a period of 200 packets. Each time the system identified the amount of path loss in the system, it was classified into its respective bucket with an associated fitness. This simulation continuously tried cases and saved their results and before each case was generated by the reasoner, the casebase was queried for a case in that situation the same path loss and requested a power with the best fitness. The returned case is not applied to speed up the convergence time, but evaluated to see what would be available to the cognitive entity.

This can also be demonstrated through a sine wave as well to demonstrate a more dynamic environment. To be more specific with the scenario encountered, we can redefine $\boldsymbol{\delta}$ to also

Table 4.10: Time Varying Utility Description

Utility	$\dot{\beta}$	p	Weight
Power Consumption	-65 dBm	-65 / -15 dBm	0.23
Signal Strength	-15 dBm	-15 / -65 dBm	0.00999
Throughput	800 kb/s	100 / 800 kb/s	0.53
Robustness	13	13 / 2 Coding	0.23

be more specific about the levels of path-loss and make the intervals of δ_0 to be 5 dB with no coding and a modulation scheme of QAM with 4 bits per symbol. To allow more cases to be tried for each of the buckets, the number of frames was extended to 2500 frames. Figure 4.8 shows the casebase responding adequately to the shifting path-loss.

4.4.2 Results

The result of this scenario produced good results, showing that the system adequately classified the scenario it experienced and converged on an optimal power for each level of path loss. Figure 4.7 shows the casebase returning the powers of the cases with the highest fitness, having converged on a power for both scenarios around packet 400 for random attempts.

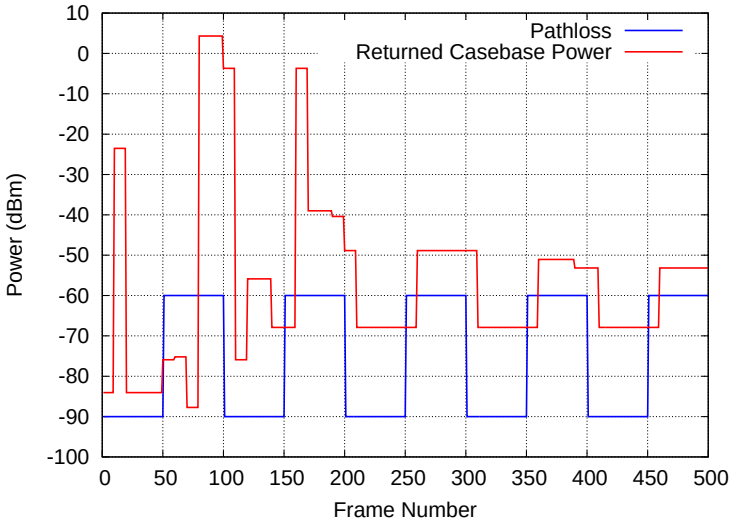


Figure 4.7: Time Varying Path Loss Scenario

There are a couple things to note about the results presented in Figure 4.8. The trends of the returned cases show a small delay between the fit choices returned by the casebase and the given pathloss. This is due to the reactive nature of a simple Reasoner. Proactive reasoners have shown promise to improve engine and overall system performance [8] and can be applied here. The other thing to note is that while these cases are returned with good

fitness, they are not applied. Again, this is so we can quickly see convergence of a random $\boldsymbol{\theta}$ generation as applied to each similar situation. The next scenario will lift this limitation and create a more realistic simulation that applies past cases to the transmission as well as create new ones.

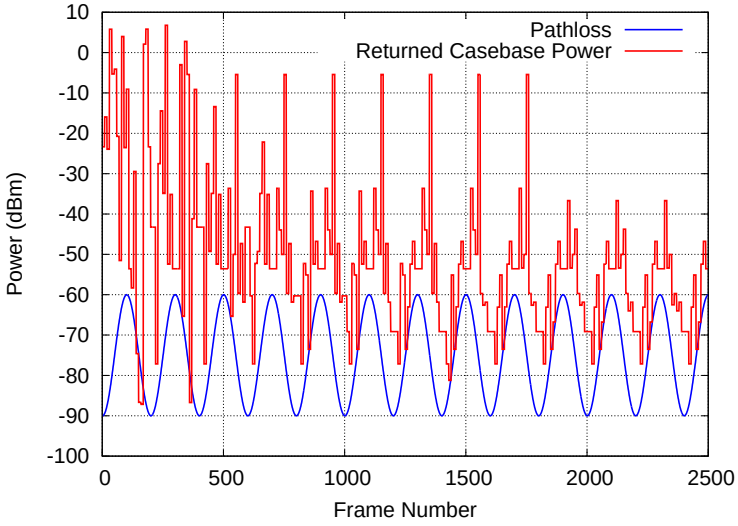


Figure 4.8: Varying Path Loss as a Sine Wave

4.5 Additional Parameters

In this simulation, the reasoner was able to modify not only transmission power, but modulation, coding and packet size. Similarity is also extended to include more than just path loss and a larger number of frames were simulated. Extending the simulation to include these parameters introduces different aspects that make the simulation more realistic but also more complex. Again a random reasoner is used to generate cases, but because cases were being queried and applied, additional steps were required to allow a trade-off between exploration versus exploitation to at least some extent. While the trade-off used in this simulation is not optimal, it allows for enough convergence to happen showing the power of the casebase's bucketing strategy. This can provide useful information to a reasoning entity to perform the optimization of the system.

4.5.1 Scenarios

In this scenario, a simulation was performed similarly to those described before in Section 4.4. The length of the simulation was increased to 10000 packets as even more parameters have been introduced to the system. Again, the speed of convergence is not the focus, but the

convergence within a bucket is what is under consideration. The parameters of the system and similarity configuration is presented in Table 4.11.

Table 4.11: Additional Parameter Simulation Configuration

Metric	Value Description
θ	
Transmission Gain	-90 thru 10 dBm
Modulation	BPSK, QPSK, QAM8 - QAM256
Coding	None, Convolution V27 P23-P78, Convolution V28 P23-P78
Packet Size	512 - 2048 bytes
β	
Throughput	b/s
ϕ	
Path Loss	0 thru 40 dB
Throughput	0 thru 2000

δ			
Similarity Dimension	min	max	Grain
Path Loss	0 dB	40 dB	1 dB
Throughput	0	2000 kbps	200 kbps

The final set of utilities used are described in Table 4.12. These are chosen to minimize power consumption and maximize throughput. Because more parameters are introduced, the weights change to reflect what a real system would aim to optimize. In [34] five utilities were chosen to be optimized; maximum throughput, minimum BER, minimum power consumption, minimum interference, and maximum spectral efficiency. The reason so few utilities are used here is because most utilities in [34] would be redundant in this system. Throughput is calculated using average good data over some time interval. This throughput reflects the BER in the system and does not need to be double counted here. Interference deals more with a networked scenario and because this is a single link transmission, is not considered here. This, however, does present an interesting utility as inter-node interference is a major limiting factor for wireless networks [17]. Spectral efficiency is a scalar measurement of the throughput of the network. Theoretical capacity can be increased, for example, by increasing the bits per symbol of the modulation scheme or by reducing the FEC rate. However, practical adjustments such as these are reflected in the throughput measurement and would essentially be double counted in this system. For this reason, they are not included.

This simulation also performs an on-line trial of generated cases, as well as cases that are returned from the casebase. The simulation uses measures of 10 frames to measure the

Table 4.12: Utility Configuration

Utility	$\dot{\beta}$	p	Weight
Power Consumption	-65 dBm	-65 / -15 dBm	0.49
Signal Strength	-15 dBm	-15 / -65 dBm	0.01
Throughput	800 kb/s	100 / 800 kb/s	0.49
Robustness	13	13 / 2 Coding	0.01

statistics of $\boldsymbol{\phi}$ and $\boldsymbol{\beta}$ and thus $\boldsymbol{\delta}$. To introduce some exploration every 11th measurement (every 110th packet) uses a randomly generated $\boldsymbol{\theta}$. During the other measurements the casebase is queried with the measured parameters of $\boldsymbol{\delta}$. Given that a case with a fitness greater than the current case is returned from the query, the returned case is applied. The additional parameters are also recorded to find the best case fit.

4.5.2 Results

In Figure 4.9 the current power of the system, the current path loss, and the resulting fitness is plotted. Because of the additional parameters, we can see that the graph does not converge as quickly as the previous simulations. This is expected, however, as more parameters mean that the cognitive entity much search through a larger solution space. As the bucket size decreases due to finer granularity, the case then becomes more descriptive about the situation it was in. There is a trade-off to this, however, that shows in Figure 4.9 in that the convergence time for the reasoning entity is increased when randomly searching the solution space. Again, a better reasoner can be used to find better solutions more quickly. Generally, near the end of the run fitness is maintained for the most part and is degraded only when a random case is forced upon the simulation.

The effect on throughput due to the variation of pathloss can be seen in Figure 4.10. Eventually the trend of the throughput finds a cyclic performance degradation which could provide useful information to a reasoning entity trying to classify the kind of interference it is experiencing. The case with the casebase here, however, shows us that the casebase can maintain a fit throughput even in a varying environment. When the throughput degrades to little to no connectivity, this is because a current case is experiencing a new level of pathloss and must be adjusted. Preemptive these fades was not employed here but can be used to reduce the amount of connection blackout. Random application of poor $\boldsymbol{\theta}$ selections can also cause a drop in connectivity.

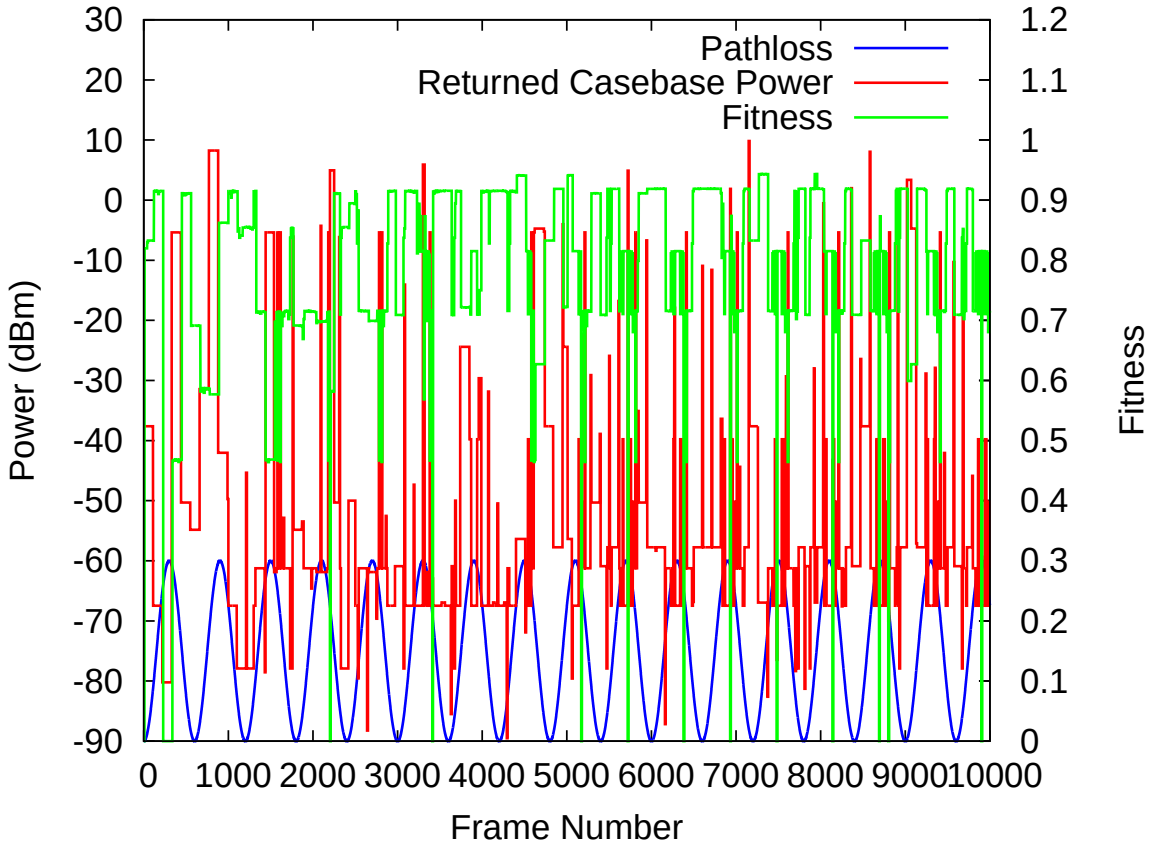


Figure 4.9: Power, Pathloss and Fitness of a Wireless Link

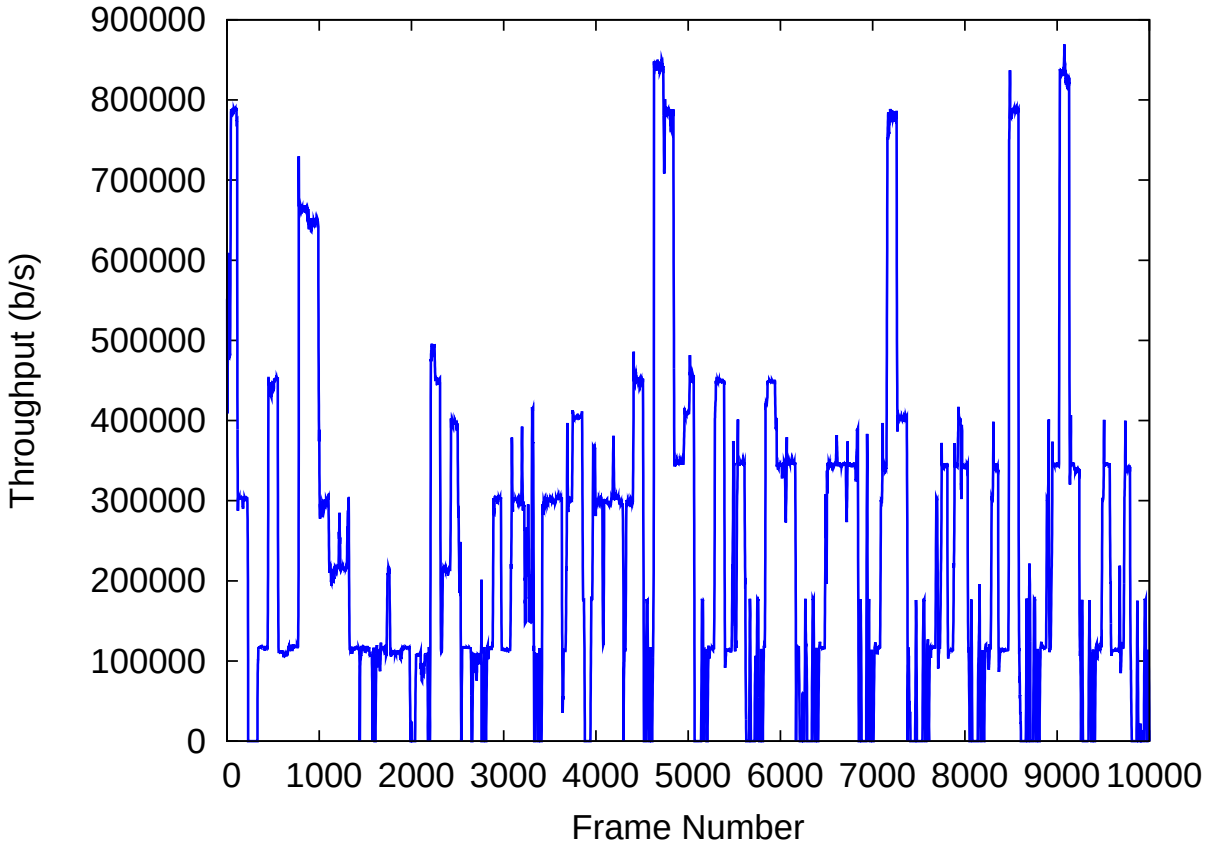


Figure 4.10: Throughput of a Wireless Link Simulation

Chapter 5

Concluding Remarks and Future Work

This work presents a renewed look at an abstract look of CBR for CR applications. The ideas presented in this work show that while casebase maintenance is important, the casebase does not need to be aggressively pruned to maintain acceptable performance. By simplifying the requirements of similarity, a constant access time can be maintained and more information can be stored for the cognitive entity to use. This can be useful when considering the a Pareto front that may be changing. One interesting note to make is that more parameters such as frequency, bandwidth, or sub-carriers could be used in $\boldsymbol{\theta}$ but were not. This is because there are not something that can be changed through some header information received at the receiver. As there are certain properties of the waveform required for the receiver to synchronize and understand the signal it is using, some protocol is needed to inform the receiver before the change takes place. Others, like power, modulation, coding and packet size can be observed by the receiver through static and robust header information. Liquid-USRP, an extension of Liquid-DSP to the universal software radio platform (USRP) , uses a robust set of parameters for the header information which contain the information to decode the rest of the samples for the packet payload. This provides a useful trade-off for efficient usage of spectrum as well as dynamic reconfigurations.

To extend this work, a dynamic bucket size would be useful for many applications. For instance, over a specific threshold of power, receivers are saturated. Given that the transmitter knew where this power was in terms of its own gain setting, it could bucket any power greater than this threshold into a similarity bucket. The same can be said for other parameters and would require a different set of Equations 3.3 and 3.4 for each dimension of $\boldsymbol{\delta}$. This could provide greater clarity for similar situation in that there need not be a constant of $g_i \ \forall \ i = 0, ..., N_{\boldsymbol{\delta}} - 1$. Given that an implementation was already in place and additional parameters need to be added to the similarity vector, the root node can be recreated with a set of children, have only one entry for all cases previously created. This is assuming that all

cases used the same parameter or at least were in the same similarity bucket when they were created. The casebase would need to be re-created from scratch given that this was not true. This should not be very often however, as the CE as the parameters are assumed to be well designed before implementation and are generally not considered dynamically changeable.

Bibliography

[1] E-UTRA: LTE Physical Layer; General Description, 2012.

[2] E-UTRA and E-UTRAN Overall Description, 2012.

[3] Agnar Aamodt and Enric Plaza. Case-based reasoning; foundational issues, methodological variations, and system approaches. *AI COMMUNICATIONS*, 7(1):39–59, 1994.

[4] David W. Aha, Dennis Kibler, and Marc K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991. 10.1007/BF00153759.

[5] I.A. Akbar and W.H. Tranter. Dynamic spectrum allocation in cognitive radio using hidden markov models: Poisson distributed case. In *SoutheastCon, 2007. Proceedings. IEEE*, pages 196 –201, march 2007.

[6] A. Amanna, D. Ali, M. Gadhiok, M. Price, and J.H. Reed. Hybrid experiential-heuristic cognitive radio engine architecture and implementation. *Hindawi Journal of Computer Networks and Communications: Special Issue on Trends and Applications of Cognitive Radio*, Jan. 2012.

[7] A. Amanna, M. Gadhiok, M.J. Price, J.H. Reed, W.P. Siriwongpairat, and T.K. Himsoon. Railway cognitive radio. *Vehicular Technology Magazine, IEEE*, 5(3):82 –89, sept. 2010.

[8] A. Amanna and J.H. Reed. Survey of cognitive radio architectures. In *IEEE SoutheastCon 2010 (SoutheastCon), Proceedings of the*, pages 292 –297, march 2010.

[9] Ashwin Amanna, Matthew J. Price, Soumava Bera, Manik Gadhiok, and Jeffrey H. Reed. Cognitive engine architecture for railway communications. *ASME Conference Proceedings*, 2010(44069):61–66, 2010.

[10] C. Clancy, J. Hecker, E. Stuntebeck, and T. O'Shea. Applications of machine learning to cognitive radio networks. *Wireless Communications, IEEE*, 14(4):47 –52, august 2007.

[11] Silicon Graphics International Corp. Standard template library. http://www.sgi.com/tech/stl/, 2011.

[12] Adam Drozdek. *Data Structures and Algorithms in C++*. Thomson Learning, Inc., 3rd edition, 2005.

[13] A. Fehske, J. Gaeddert, and J.H. Reed. A new approach to signal classification using spectral correlation and neural networks. In *New Frontiers in Dynamic Spectrum Access Networks, 2005. DySPAN 2005. 2005 First IEEE International Symposium on*, pages 144 –150, nov. 2005.

[14] J. Gaeddert. Liquid-dsp. https://github.com/jgaeddert/liquid-dsp, 2012.

[15] Joseph D. Gaeddert. *Facilitating Wireless Communications through Intelligent Resource Management on Software-Defined Radios in Dynamic Spectrum Environments*. PhD thesis, Virginia Polytechnic Institute & State University, Blacksburg, VA, January 2011.

[16] C. Ghosh, C. Cordeiro, D.P. Agrawal, and M.B. Rao. Markov chain existence and hidden markov models in spectrum sensing. In *Pervasive Computing and Communications, 2009. PerCom 2009. IEEE International Conference on*, pages 1 –6, march 2009.

[17] P. Gupta and P.R. Kumar. The capacity of wireless networks. *Information Theory, IEEE Transactions on*, 46(2):388 –404, mar 2000.

[18] M. Hasegawa, Ha Nguyen Tran, G. Miyamoto, Y. Murata, and S. Kato. Distributed optimization based on neurodynamics for cognitive wireless clouds. In *Personal, Indoor and Mobile Radio Communications, 2007. PIMRC 2007. IEEE 18th International Symposium on*, pages 1 –5, sept. 2007.

[19] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999.

[20] A. He, Kyung Kyoon Bae, T.R. Newman, J. Gaeddert, Kyouwoong Kim, R. Menon, L. Morales-Tirado, J.J. Neel, Youping Zhao, J.H. Reed, and W.H. Tranter. A survey of artificial intelligence for cognitive radios. *Vehicular Technology, IEEE Transactions on*, 59(4):1578 –1592, may 2010.

[21] An He, Joseph Gaeddert, Kyung Kyoon Bae, Timothy R. Newman, Jeffrey H. Reed, Lizdabel Morales, and Chang-Hyun Park. Development of a case-based reasoning cognitive engine for ieee 802.22 wran applications. *SIGMOBILE Mob. Comput. Commun. Rev.*, 13:37–48, September 2009.

[22] Honglin Hu, Jian Zhang, Xiaoying Zheng, Yang Yang, and Ping Wu. Self-configuration and self-optimization for lte networks. *Communications Magazine, IEEE*, 48(2):94 –100, february 2010.

[23] M. Khedr and H. Shatila. Cogmax- a cognitive radio approach for wimax systems. In *Computer Systems and Applications, 2009. AICCSA 2009. IEEE/ACS International Conference on*, pages 550 –554, may 2009.

[24] Kyouwoong Kim, I.A. Akbar, K.K. Bae, Jung sun Urn, C.M. Spooner, and J.H. Reed. Cyclostationary approaches to signal detection and classification in cognitive radio. In *New Frontiers in Dynamic Spectrum Access Networks, 2007. DySPAN 2007. 2nd IEEE International Symposium on*, pages 212 –215, april 2007.

[25] Hiroaki Kitano. Challenges of massive parallelism. In *Proceedings of the 13th international joint conference on Artifical intelligence - Volume 1*, IJCAI'93, pages 813–834, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[26] Janet L. Kolodner. An introduction to case-based reasoning. *Artificial Intelligence Review*, 6(1):3–34, March 1992.

[27] Janet L. Kolodner and David B. Leake. A Tutorial Introduction to Case-Based Reasoning. In David B. Leake, editor, *Case-Based Reasoning: Experiences, Lessons, & Future Directions*, chapter 2, pages 3–65. AAAI Press/MIT Press, Menlo Park, CA/Cambridge, MA, 1996.

[28] Bin Le, Thomas W. Rondeau, and Charles W. Bostian. Cognitive radio realities. *Wireless Communications and Mobile Computing*, 7(9):1037–1048, 2007.

[29] P C Mahalanobis. On the generalized distance in statistics. *Proceedings of the National Institute of Science Calcutta*, 2(1):49–55, 1936.

[30] Ramon Lopez De Mantaras, Derek Bridge, and David Mcsherry. Case-based reasoning: an overview. *AI Communications*, 10:21–29, 1997.

[31] III Mitola, J. and Jr. Maguire, G.Q. Cognitive radio: making software radios more personal. *Personal Communications, IEEE*, 6(4):13 –18, aug 1999.

[32] Tim R. Newman, Brett A. Barker, Alexander M. Wyglinski, Arvin Agah, Joseph B. Evans, and Gary J. Minden. Cognitive engine implementation for wireless multicarrier transceivers. *Wirel. Commun. Mob. Comput.*, 7(9):1129–1142, November 2007.

[33] Timothy R. Newman, Rakesh Rajbanshi, Alexander M. Wyglinski, Joseph B. Evans, and Gary J. Minden. Population adaptation for genetic algorithm-based cognitive radios. In *Cognitive Radio Oriented Wireless Networks and Communications, 2007. CrownCom 2007. 2nd International Conference on*, pages 279 –284, aug. 2007.

[34] T.R. Newman and J.B. Evans. Parameter sensitivity in cognitive radio adaptation engines. In *New Frontiers in Dynamic Spectrum Access Networks, 2008. DySPAN 2008. 3rd IEEE Symposium on*, pages 1 –5, oct. 2008.

[35] J. Paine. Expert systems. http://www.j-paine.org/students/lectures/lect3/lect3.html.

[36] S.K. Pal and S.C.K. Shiu. *Foundations of soft case-based reasoning.* Wiley series on intelligent systems. John Wiley & Sons, 2004.

[37] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257 –286, feb 1989.

[38] Christian James Rieser. *Biologically Inspired Cognitive Radio Engine Model Utilizing Distributed Genetic Algorithms for Secure and Robust Wireless Communications and Networking*. PhD thesis, Virginia Tech, Blacksburg, VA, 2004.

[39] Thomas W. Rondeau. *Application of Artificial Intelligence to Wireless Communications*. PhD thesis, Virginia Tech, Blacksburg, VA, 2007.

[40] T.W. Rondeau, C.J. Rieser, T.M. Gallagher, and C.W. Bostian. Online modeling of wireless channels with hidden markov models and channel impulse responses for cognitive radios. In *Microwave Symposium Digest, 2004 IEEE MTT-S International*, volume 2, pages 739 – 742 Vol.2, june 2004.

[41] Richard H. Stottler, Andrea L. Henke, and James A. King. Rapid retrieval algorithms for case-based reasoning. In *Proceedings of the 11th international joint conference on Artificial intelligence - Volume 1*, pages 233–237, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

[42] Ryan W. Thomas, Daniel H. Friend, Luiz A. Dasilva, and Allen B. Mackenzie. Cognitive networks: adaptation and learning to achieve end-to-end performance objectives. *Communications Magazine, IEEE*, 44(12):51 –57, dec. 2006.

[43] Thomas W. Rondeau (virginia Tech and Usa Trondeau@vt. Edu. Cognitive radios with genetic algorithms: Intelligent control of software defined radios.

[44] H.I. Volos and R.M. Buehrer. Cognitive engine design for link adaptation: An application to multi-antenna systems. *Wireless Communications, IEEE Transactions on*, 9(9):2902 –2913, september 2010.

[45] Stefan Wess, Klaus dieter Althoff, and Guido Derwand. Using k-d trees to improve the retrieval step in case-based reasoning. In *Stefan Wess, Klaus-Dieter Althoff, & M. M. Richter*, pages 167–181. Springer-Verlag, 1993.

[46] Zhenyu Zhang and Xiaoyao Xie. Intelligent cognitive radio: Research on learning and evaluation of cr based on neural network. In *Information and Communications Technology, 2007. ICICT 2007. ITI 5th International Conference on*, pages 33 –37, dec. 2007.

[47] Youping Zhao, Joseph Gaeddert, Lizdabel Morales, Kyung Bae, Jung-Sun Um, and Jeffrey H. Reed. Development of radio environment map enabled case- and knowledge-based learning algorithms for ieee 802.22 wran cognitive engines. In *Cognitive Radio Oriented Wireless Networks and Communications, 2007. CrownCom 2007. 2nd International Conference on*, pages 44 –49, aug. 2007.

[48] Xiang-Lin Zhu, Yuan-An Liu, Wei-Wen Weng, and Dong-Ming Yuan. Channel sensing algorithm based on neural networks for cognitive wireless mesh networks. In *Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08. 4th International Conference on*, pages 1 –4, oct. 2008.