Ex Ante Approaches for Security, Privacy, and Enforcement in Spectrum Sharing

Behnam Bahrak

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

Doctor of Philosophy
in
Electrical Engineering

Jung-Min "Jerry" Park, Chair
Carl Dietrich
Cameron Patterson
Sandeep Shukla
Anil Vullikanti

November 22, 2013
Blacksburg, Virginia

Keywords: Cognitive Radio, Dynamic Spectrum Access, Spectrum Enforcement, Policy Reasoning, Spectrum Learning, Geolocation Databases, Privacy-preserving Techniques.

Ex Ante Approaches for Security, Privacy, and Enforcement in Spectrum Sharing

Behnam Bahrak

ABSTRACT

Cognitive radios (CRs) are devices that are capable of sensing the spectrum and using its free portions in an opportunistic manner. The free spectrum portions are referred to as white spaces or spectrum holes. It is widely believed that CRs are one of the key enabling technologies for realizing a new regulatory spectrum management paradigm, viz. dynamic spectrum access (DSA) . CRs often employ software-defined radio (SDR) platforms that are capable of executing artificial intelligence (AI) algorithms to reconfigure their transmission/reception (TX/RX) parameters to communicate efficiently while avoiding interference with licensed (a.k.a. primary or incumbent) users and unlicensed (a.k.a. secondary or cognitive) users.

When different stakeholders share a common resource, such as the case in spectrum sharing, security, privacy, and enforcement become critical considerations that affect the welfare of all stakeholders. Recent advances in radio spectrum access technologies, such as CRs, have made spectrum sharing a viable option for significantly improving spectrum utilization efficiency. However, those technologies have also contributed to exacerbating the difficult problems of security, privacy and enforcement. In this dissertation, we review some of the critical security and privacy threats that impact spectrum sharing. We also discuss ex ante (preventive) approaches which mitigate the security and privacy threats and help spectrum enforcement.

Dedication

I dedicate this dissertation to my wonderful family and many great friends. A special feeling of gratitude to my loving parents, whose support and words of encouragement have carried me to the finish line.

Acknowledgments

I would like to express my gratitude for many people who helped me during my graduate study at Virginia Tech. First and foremost, I would like to sincerely thank my Ph.D. advisor, Dr. Jung-Min "Jerry" Park, for his patience, generous support, and invaluable guidance in my pursuit of academic excellence. I will never forget his endeavors to make my graduate study productive.

Besides my advisor, I would like to thank the rest of my doctoral advisory committee: Dr. Carl Dietrich, Dr. Cameron Patterson, Dr. Sandeep Shukla, and Dr Anil Vullikanti, for their insightful comments, and constructive criticisms.

In my daily work I have been blessed with a friendly and cheerful group of fellow students in ARIAS, CESCA and Wireless@VT. I take this opportunity to thank my friends at these groups: Amol Deshpande, Swati Kanaujia, Jatin Thakkar, Maxwell Whitekar, Kaigui Bian, Daniel Ali, Bo Gao, Hao Wu, Vireshwar Kumar, Amr Nabil, Seungmo Kim, Abid Ullah, and Sudeep Bhattarai, for their contribution to my research experience and personal life at Virginia Tech. I enjoyed working with you and have learned a lot from you. I wish you all the best of luck in your life endeavors.

I must thank all my friends at Blacksburg, particularly my roommates Ali and Omid, for making the past five years of my life memorable.

Contents

List of Figures

List of Tables

List of Abbreviations

AE	Atomic Expression
AI	Artificial Intelligence
BRESAP	Binary decision diagram-based REasoner for Spectrum Access Policies
CR	Cognitive Radio
CSMAC	Commerce Spectrum Management Advisory Committee
DARPA	Defense Advanced Research Projects Agency
DL	Description Logic
DoD	Department of Defense
DSA	Dynamic Spectrum Access
FCC	Federal Communications Commission
FEC	Forward Error Correction
FIA	Fine-grained Integration Algebra
FOL	First Order Logic
FSK	Frequency Shift Keying
GDB	Geolocation Database
GMSK	Gaussian Minimum Shift Keying
HMM	Hidden Markov Model
LBS	Location-Based Service
LP	Location Privacy

MAC	Medium Access Control
MTBDD	Multi Terminal Binary Decision Diagram
MTP	Maximum Transmission Power
NPRM	Notice of Proposed Rule Making
NSF	National Science Foundation
OWL	Web Ontology Language
PCC	Policy Conformance Component
PL	Performance Loss
PLOT	Primary users Location Obfuscation Technique
PR	Policy Reasoner
PU	Primary User
Qos	Quality of Service
RDF	Resource Description Framework
RF	Radio Frequency
RL	Reinforcement Learning
RL	Reinforcement Learning
ROBDD	Reduced Ordered Binary Decision Diagram
RX	Reception
SDLC	Synchronous Data Link Control
SDR	Software Defined Radio
SPRT	Sequential Probability Ratio Test
SSR	System Strategy Reasoner
SU	Secondary User
SWRL	Semantic Web Rule Language
TP	Transmission Period

TX	Transmission
USRP	Universal Software Radio Peripheral
W3C	World Wide Web Consortium
WM	Working Memory
WME	Working Memory Element
WSD	White Space Device
XG	neXt Generation
XML	eXtensible Markup Language
XSD	XML Schema Definition

Chapter 1

Introduction

With the rapid deployment of new wireless devices and applications, we are facing with a growing demand for wireless radio spectrum. However, the fixed spectrum assignment policy becomes a bottleneck for more efficient spectrum utilization, under which a great portion of the licensed spectrum is severely under-utilized. The inefficient usage of the limited spectrum resources urged the spectrum regulators like the Federal Communications Commission (FCC) to review their policy and start to seek for innovative communication technology that can exploit the wireless spectrum in a more intelligent and flexible way.

Cognitive radios (CRs) are devices that are capable of sensing the spectrum and using its free portions in an opportunistic manner. The free spectrum portions are referred to as white spaces or spectrum holes. It is widely believed that CRs are one of the key technologies that can address the spectrum scarcity problem. It is expected that they will play an important role in maximizing spectrum utilization and help satisfy the QoS requirements of a number of important communications applications from emergency first responders public safety networks to military tactical networks. CRs are able to perform onboard, real-time optimization of several TX/RX parameters such as transmission frequency, bandwidth, power, modulation type, etc. CRs face new security threats and challenges that have arisen due to their unique cognitive characteristics. To unleash the full potential of CRs, we have

to consider a number of key security and privacy issues that are specific to CRs.

In the spectrum sharing paradigm, a heterogeneous mix of wireless systems of differing access priorities, Quality of Service (QoS) requirements, and transmission characteristics need to coexist without causing harmful interference to each other. When different stakeholders share a common resource (such as the case in spectrum sharing), security, privacy, and enforcement become critical considerations that are essential to the welfare of all stakeholders. Security and enforcement are especially paramount considerations related to the recent calls in the United States for sharing of federal government (including military) spectrum with non-government systems.

In this dissertation, we investigate some of the critical security and privacy threats in dynamic spectrum access and spectrum sharing. We also discuss threat countermeasures and spectrum rule enforcement. We can classify the enforcement techniques in the context of two distinct approaches—*ex ante* and *ex post* enforcement. The former represents actions that are designed to "prevent" or reduce the likelihood of a potentially harmful interference event or a privacy violation, while the latter denotes "punitive" measures designed to punish malicious or selfish behavior after a potentially harmful interference event has occurred. We focus on ex ante approaches, and propose three preventive measures for security, privacy and enforcement in spectrum sharing.

1.1 Preventive Measures for Rogue Transmissions

Enforcing spectrum access control in legacy radios (e.g., cellular phones) is relatively straightforward since the spectrum access policies are an inseparable part of the radio's firmware and platform. Making controlled changes to a legacy radio's transmission behavior would require an adversary to have very specialized expertise in the radio's firmware and hardware, and would also require specialized equipment. Unfortunately, manipulating the transmission behavior of software defined radios (SDRs) and CRs is easier. The reconfigurability of a

SDR/CR makes it vulnerable to unauthorized modification. Such modifications can result in harmful interference. Illegally modified radios can even be used to launch very sophisticated jamming attacks, as shown in [1].

One approach for enforcing spectrum access control in spectrum sharing is to employ *policy-based* CRs. Policy-based CRs cope with evolving spectrum access policies and constantly changing application requirements by decoupling the policies from device-specific implementations and optimizations. These radios can invoke situation-appropriate adaptive actions based on policy specifications and the current spectrum environment [2]. Enforcing spectrum access policies by mandating the use of policy-based CRs is one effective approach for mitigating rogue transmissions.

In order to regulate and enforce proper transmission behavior, policy-based CRs need mechanisms to enforce spectrum access policies. Most of these mechanisms are carried out by specialized software modules called policy conformance components (PCCs) [3]. To enforce spectrum policies, the policies themselves first need to be interpreted, and then a CR's transmission strategies need to be evaluated against those policies to determine the legality of the transmission strategies. Within a policy-based CR, the aforementioned tasks are carried out in real time by a software module called the *policy reasoner*.

In this dissertation, we propose two novel policy reasoners (PRs) that can be used to enforce policy conformance and prevent rogue transmission in a policy-based CR. Our first PR is called BRESAP (Binary decision diagram-based REasoner for Spectrum Access Policies), which is a *rule-based policy reasoner*. Rule-based policies use logic programming techniques to encode the axioms and rules in a straightforward way [4]. Using rule-based spectrum policies simplifies the design of the policy reasoner because the reasoning complexity is sufficiently low in most applications to meet the real-time processing requirements of the radio. However, rule-based policies have a number of critical drawbacks. The most serious drawbacks are policy management overhead and limited interoperability. With rule-based policies, complex spectrum policies are difficult to specify and manage. Moreover, rule-

based policies do not support the sharing of the policy structure among different policy authors (i.e., regulation authorities), and thus limits interoperability of the policy-based radios across different regulatory policy domains. To overcome the limitations of rule-based spectrum policies, there is a growing interest in using *ontology-based policies* for prescribing spectrum access rules [5]. In fact, the IEEE 1900.5 Standard, Standard for Policy Language Requirements and System Architectures for Dynamic Spectrum Access Systems, published in 2012, prescribes the use of ontology-based policy language for managing the functionality and behavior of dynamic spectrum access networks. The second PR that we introduce in this dissertation is an ontology-based policy reasoner that can be used to enforce ontology-based spectrum access policies in a policy-based cognitive radio.

1.2 Preventive Measures for Belief Manipulations

Learning is an important feature of a CR which consists of evaluating the outcome of the decisions which have been made, and gathering knowledge to be exploited in future decision-making process. While learning can make wireless devices to be more flexible, it can cause a security threat. A radio that can learn has the potential to be taught by malicious entities in an adversarial environment. This kind of threat which is called a *belief manipulation attack* may have a long-lasting impact on the cognitive radio network [6].

Considering the computing limitations and energy constraints of a battery-powered CR, a CR may not be able to perform full-spectrum sensing (i.e., sense all available spectrum bands) because of its prohibitive cost. Therefore, a spectrum sensing policy at the medium access control (MAC) layer is needed to decide which set of channels to sense. The *channel selection problem* is the problem of designing such a sensing policy. The optimal channel selection strategy for an unlicensed user (i.e., secondary user) is based on the availability statistics of the channels. The availability of the channels is determined by the presence/absence of primary user signals in those channels. The channels' availability statistics are initially un-

known to a secondary user and need to be learned using sensing samples. *Spectrum learning* is the process of learning the spectrum statistics (i.e., primary user occupancy information), which is crucial to enable CRs to sense/interpret their spectrum environment and make intelligent decisions to achieve efficient communication. Although spectrum learning is beneficial for CRs, it can be used by adversaries to perform a belief manipulation attack.

In order to design an optimal sensing policy for a CR, we need to address the critical tradeoff, that the cognitive engine faces in each timeslot, between transmission ("exploitation") on the channel that has the highest expected reward (e.g., throughput) and channel sensing ("exploration") to get more information about the expected rewards of the other channels. The exploitation vs. exploration tradeoff problem, such as the one just described, is usually formulated as a *Reinforcement Learning* (RL) problem. In this dissertation, we analyze the security of channel selection algorithms that are based on reinforcement learning and propose mitigation techniques that make these algorithms more robust against belief manipulation attacks and reduce the probability of harmful interference.

1.3 Preventive Measures for Privacy Violations

The FCC ruling on TV white spaces proposes relying on a database of the incumbents' spectrum usage information as the primary means of determining white space availability at any white space device (WSD) [7]. The database is required to house an up-to-date repository of incumbents including television stations, and in certain cases, wireless microphones, and use this information to determine white space availability at a white space device's location. It has been shown that sensing-only devices do not generally utilize spectrum as efficiently as geolocation enabled devices, due to the large margins in incumbent detection thresholds that must be built into sensing-only devices [8]. Geolocation enabled devices have knowledge of the specific interference protection requirements of each licensed incumbent, which allows varying levels of protection to be applied, and thus maximize utilization of the spectrum.

Although using geolocation databases (GDBs) for spectrum sharing has many advantages, it poses a potentially serious privacy problem. For instance, SUs, through seemingly innocuous queries to the database, can determine the *types*, *locations*, and *times of operation* of incumbent systems operating in a given region of interest—we refer to this as the *operational privacy* of the incumbents. When the incumbent systems are commercial systems, such as the case in TV spectrum, this is not a problem. However, when the incumbents are federal government, possibly military systems, then the information revealed by the databases may result in a serious breach of operational privacy. Moreover, there is the possibility that SUs can obtain knowledge beyond that revealed directly by the database's query replies by using sophisticated inference techniques—we refer to this as a *database inference attack*. The operational privacy of primary users is an especially critical concern related to the recent calls in the United States for sharing of federal government (including military) spectrum in the 3.5 GHz band with non-government systems.

The problem of operational privacy of PUs cannot be addressed by tightly controlling access to the database, since all SUs need access to the GDB to enable spectrum sharing. A more viable approach is to "obfuscate" the information revealed by the database in an intelligent manner such that a certain level of privacy is assured while supporting efficient use of the spectrum.

In this dissertation, we focus on *location privacy* of PUs and show that a group of malicious queriers can infer the location of the incumbent systems even though the database's responses to the queries do not directly reveal the locations. We also propose a countermeasure called PLOT (Primary user Location Obfuscation Technique) that use perturbative masking method (a.k.a. randomization method) to preserve location privacy of PUs and can be employed by a GDB to make an appropriate tradeoff between protecting the incumbents' location privacy and enabling efficient spectrum utilization for SUs' network. In addition to PLOT, we provide several other privacy preserving techniques that may be used to prevent inferring the locations of PUs.

1.4 Dissertation Organization

The rest of this document is organized as follows. In Chapter 2, we introduce a rule-based policy reasoner that enforces trustworthy transmission behavior in CRs. In order to address some of the shortcomings of this rule-based policy reasoner, in chapter 3, we describe the design of an ontology-based policy reasoner that allows using more descriptive policies. Chapter 4 discusses the security issues of spectrum learning, followed by our techniques for preserving the location privacy of primary users in geolocation database-driven cognitive radio networks in Chapter 5. Finally, we conclude this dissertation and provide directions for future work in Chapter 6.

Chapter 2

Spectrum Access Policy Reasoning for Policy-based Cognitive Radios

In current radios, regulators define static spectrum policies for application-specific radios and the manufacturers must embed these policies into their radios as part of their firmware. These radios would need to be certified before sale to ensure that they are policy conformant. This approach has a number of shortcomings. In such radios, the policies and the radio platform are inseparable and any change in the policies requires reimplementation and reaccreditation of the radio.

In order to fully reap the potential benefits of dynamic spectrum access, radios need to be able to cope with evolving spectrum access policies and constantly changing application requirements [9]. Policy-based spectrum access addresses these challenges by decoupling the policies from device-specific implementations and optimizations. In addition to facilitating dynamic policy changes, policy-based spectrum access reduces certification effort, increases the cognitive radio's ability to adapt its behavior to rapidly changing spectrum availability conditions, and enables extensible DSA with respect to enabling the radios to be regulated by flexible and expressive regulatory policies [10].

The cognitive radio architecture developed under the auspices of DARPA's neXt Generation (XG) communication program [11] adopts a policy-based radio framework that enables the radios to dynamically load policies without the need to recompile any software on the radio. This ability enables radios to quickly adapt to rapidly changing situations. Each XG radio is equipped with a set of Policy Conformance Components (PCCs) that is responsible for ensuring that the radio's behavior conforms to the currently active policies and does not cause harmful interference [3]. The *policy reasoner* (*PR*) is the main inference component of the PCCs. The *system strategy reasoner* (*SSR*) is the component that controls the hardware, gathers sensory information, formulates transmission strategies, and acts as an interface for transmitting and receiving signals. The SSR interacts with the PR to determine the available spectrum access opportunities that conform to the currently active set of policies. Specifically, the SSR formulates a transmission strategy, based on collected sensory information and its current state, and sends this information to the PR in the form of a transmission request. The PR evaluates the transmission request against the policies to verify whether the transmission strategy conforms to the policies. If some of the configuration states in the transmission request are underspecified—which renders the transmission request invalid—then the PR computes the missing parameters and returns them to the SSR along with a transmission denial message. The missing parameters computed by the PR are referred to as *opportunity constraints*. We provide more details on how the PR and the SSR interact in Section 2.1.1.

The core reasoning problem in the policy-based cognitive radio is to infer from a given set of policies and a set of transmission strategies of a transmission request, whether a transmission request is valid or invalid; in addition, if the latter is true, determine the opportunity constraints. Efforts to solve this problem are ongoing. In [10], the authors propose a strategy of using a combination of reasoning techniques that includes forward reasoning, partial evaluation, backward chaining, conditional equational rewriting, and constraint propagation. We propose to tackle the problem by proposing a novel method for processing and reasoning about policies and transmission requests.

In this chapter, we propose a new policy reasoner—named *BRESAP* (*Binary decision diagram-based REasoner for Spectrum Access Policies*)—that utilizes Multi Terminal Binary Decision Diagrams (MTBDDs) to carry out policy reasoning. The PR uses a set of efficient graph-theoretic algorithms to translate policies into MTBDDs, merge policies into a single meta-policy, and compute opportunity constraints. We demonstrate that policies can be processed efficiently by *reframing the policy reasoning problem as a graph-based Boolean function manipulation problem.* The proposed policy reasoner employs graph-based algorithms to compute opportunity constraints for under-specified or invalid transmission requests.

2.1 Technical Background

In this section we provide some technical background that facilitates the understanding of this chapter's technical aspects.

2.1.1 XG Radio Architecture

DARPA's XG program is a technology development project sponsored by its Strategic Technology Office, with the goal of developing both the enabling technologies and system concepts to dynamically redistribute allocated spectrum along with novel waveforms [11]. An XG radio has four main components [12]:

- Sensors: An XG radio needs sensors to sense its environment and discover available spectrum.
- Radio Frequency (RF) transceiver: The RF component of an XG radio is used to transmit and receive various signals.
- System Strategy Reasoner (SSR): The SSR component controls the hardware, gathers

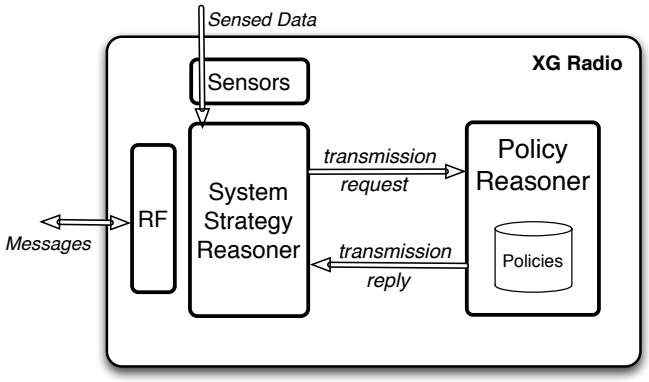


Figure 2.1: XG radio architecture.

sensory information, formulates transmission strategies, and acts as an interface for transmitting and receiving signals. The SSR interacts with the PR to determine the available spectrum opportunities that conform to the policies. Specifically, the SSR formulates a transmission strategy, based on collected sensory information and its current state, and sends this information to the PR in the form of a transmission request.

- Policy Reasoner (PR): The PR receives the transmission requests generated by the SSR and evaluates them for policy conformance. The result of the evaluation is returned to the SSR as a transmission reply along with a set of opportunity constraints, if applicable.

These components are shown in Figure 2.1. A more detailed description of the XG architecture can be found in [11], [12].

2.1.2 Spectrum Policies and Policy Languages

The main requirements for a policy language, as stated in [13], are accreditability, extensibility and expressiveness. The accreditability requirement dictates that it should be possible to accredit the policies independently of the policy reasoner and the radio. The extensibility

requirement prescribes that it should be straightforward to add new domains to the policies, while the expressiveness requirement dictates that the policy language should be capable of satisfying the requirements of any new domain introduced. The XG policy language was devised to satisfy these three requirements. The XG policy language is a declarative, semantic language, and it uses Semantic Web Rule Language First Order Logic (SWRL FOL), which is an extension of the Semantic Web Rule Language (SWRL) [14].

A policy language uses the concepts of permissions and prohibitions to express the actions that the device can undertake. Two types of policies are used—viz, permissive policies and prohibitive policies. The permissive policy expresses the states in which the policy is applicable to the device and the constraints the device must satisfy in order to allow the transmission. On the other hand, the prohibitive policy defines the states and the conditions under which the device is prohibited from transmitting. Spectrum access policies usually adopt a de-confliction rule, such as a rule that prescribes that a prohibitive policy always overrides a permissive policy. This means that if a prohibitive policy and a permissive policy conflict with each other (i.e., both cannot be satisfied), then the transmission is prohibited.

2.1.3 Policy Representation and MTBDDs

A Binary Decision Diagram (BDD) is a data structure that is used to represent a Boolean function [15]. A Boolean function can be represented as a rooted, directed, and acyclic graph, which consists of decision nodes and two terminal nodes called the 0-terminal and the 1-terminal that each has zero out-degree. Each decision node, u, is labeled by a Boolean variable, $var(u)$, and has two child nodes called a low child, $low(u)$ and a high child, $high(u)$. An edge from a node to a low (high) child represents an assignment of the variable to 0 (1). In a figurative representation of a BDD, an edge to a low child is represented using a dotted line whereas an edge to a high child is represented using a solid line. Such a BDD is called an *ordered* BDD if different variables appear in the same order on all paths from the root. A BDD is said to be *reduced* if the following two rules have been applied to its graph:

- uniqueness: no two distinct nodes u and v have the same variable name and the same low- and high-successors, i.e., $low(u) = low(v)$, $high(u) = high(v)$, and $var(u) = var(v)$ implies that $u = v$.
- no variable node u has identical low- and high- successors, i.e., $high(u) \neq low(u)$.

In popular usage, the term BDD almost always refers to Reduced Ordered Binary Decision Diagram (ROBDD). The advantage of an ROBDD is that it is unique for a particular functionality. The size of the BDD is determined both by the function being represented and the chosen ordering of the variables.

Multi terminal BDDs [16] (a.k.a. algebraic decision diagrams) extend BDDs such that they can represent functions of an arbitrary range, while their domain is still a multidimensional Boolean space, i.e., an MTBDD provides a compact representation of functions of the form $P : B^n \rightarrow E$, which maps bit vectors over a set of variables, B^n, to a finite set of results E. So the structure of an MTBDD is the same as a BDD with the difference that MTBDDs have more than two terminal nodes.

BRESAP processes MTBDDs with three terminal nodes. We use the following explicit representation of MTBDDs. Nodes will be represented as numbers 0, 1, 2, ..., with 0, 1 and 2 reserved for the terminal nodes N, Y and NA, respectively. The variables in the ordering $x_0 < x_1 < \ldots < x_{n-1}$ are represented by their indices 0, 1, 2, ..., $n - 1$. The MTBDD is stored in a table $T : u \rightarrow (i, l, h)$ which maps a node u to its three attributes $var(u) = i$, $low(u) = l$ and $high(u) = h$.

MTBDDs have been proven to be a simple and an efficient representation method for policies written in XML (eXtensible Markup Language). All the operators in the fine-grained algebra that our policy reasoner employs can be mapped to efficient operations on the underlying MTBDD. BDDs is an appropriate data structure for representing and processing dynamic spectrum access policies, because they are scalable and there are powerful tools for processing and merging policies represented by BDDs. Another important advantage is the simplicity

of conflict resolution when BDDs are used to represent and process spectrum policies.

2.1.4 XML Policy Algebra

To precisely describe the workings of BRESAP, we need to formally define the notion of transmission requests and spectrum access policies. Throughout this chapter, we use the terms *transmission request* and *request* interchangeably. Suppose there exists a finite set of names, A. Each attribute, characterizing an object in a policy, has a name, a, in A. In spectrum access policies, these attributes include transmission power level, frequency band, peak sensed power, location and time. The formal definitions of transmission requests and spectrum access policies are given in Definition 1 and Definition 2, respectively. A spectrum access policy is defined by the set of transmission requests that are permitted by the policy and the set of transmission requests that are prohibited by the policy. This simple notion will provide us with a precise characterization of the meaning of policy integration in terms of the sets of permitted and prohibited requests.

Definition 1. *Let $a_1, a_2, ..., a_k$ represent a sequence of attribute names and $v_1, v_2, ..., v_k$ represent a sequence of attribute values, where v_i is an attribute value of the attribute a_i. We define a transmission request as $r = \{(a_1, v_1), (a_2, v_2), \cdots, (a_k, v_k)\}$.*

Definition 2. *Let P be a 3-valued spectrum access policy. We define the semantics of P as a 3-tuple $\langle R_Y^P, R_N^P, R_{NA}^P \rangle$, where R_Y^P is the set of requests that is permitted, R_N^P is the set of requests that is prohibited, and R_{NA}^P is the set of requests that do not belong to either of the other two sets.*

BRESAP utilizes the Fine-grained Integration Algebra (FIA) proposed in [17]. We briefly explain this policy algebra here. The FIA is given by $\langle \Sigma, P_Y, P_N, +, \&, \neg, \Pi_{dc} \rangle$, where Σ is a vocabulary of attribute names and their domains, P_Y and P_N are two policy constants, $+$ and $\&$ are two binary operators, and $\neg$ and Π_{dc} are two unary operators.

We now describe the policy constants and operators in FIA. In what follows, P_1 and P_2 denote two policies to be combined and P_I denotes the policy obtained from the combination. Operators on policies are described as set operations.

- Permit Policy (P_Y): It is a policy constant that permits everything.
- Deny Policy (P_N): It is a policy constant that denies everything.
- Addition (+): Addition of policies P_1 and P_2 results in a combined policy P_I in which requests that are permitted by either P_1 or P_2 are permitted, requests that are denied by one policy and is not permitted by the other are denied.
- Intersection (&): Given two policies P_1 and P_2 , the intersection operator returns a policy P_I which is applicable to all requests requests having the same decision from P_1 and P_2.
- Negation ($\neg$): Given a policy P, $P_I = \neg P$ returns a policy, which permits (denies) all requests denied (permitted) by P. The negation operator does not affect those requests that are not applicable to the policy.
- Domain projection (Π_{dc}): The domain projection operator takes as a parameter the domain constraint, *dc*, and restricts a policy to the set of requests identified by *dc*.

2.2 BRESAP Architecture

BRESAP consists of six main components: *Policy Parser*, *Policy Converter*, *Transmission Request Interpreter*, *Policy Merger*, *Policy Preprocessor*, and *Reasoner*. Figure 2.2 illustrates the architecture of BRESAP. In what follows, we will describe these components and their functionalities.

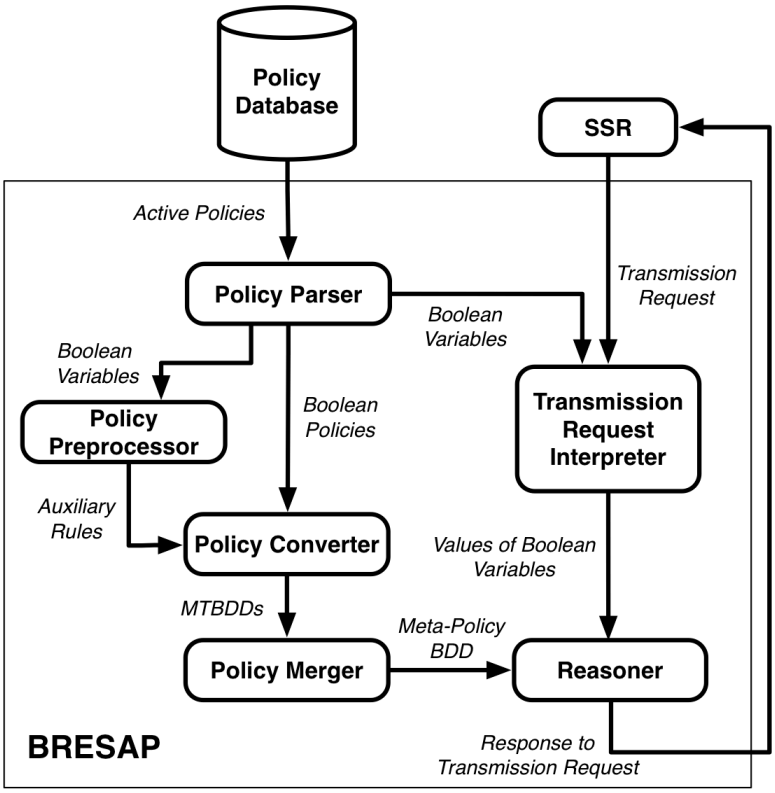


Figure 2.2: Architecture of BRESAP.

2.2.1 Policy Parser

The Policy Parser module converts the spectrum access policies that are written in SWRL into Boolean expressions using *Boolean encoding*. The *attributes* are the building blocks of an SWRL FOL policy which represent the various parameters of Dynamic Spectrum Access domain such as center frequency, transmit power, sensed power, time and location. Each of these attributes has a certain value assigned to it, called an *attribute value*. A fragment of an SWRL FOL policy designating the center frequency to 231 MHz is shown below:

$< swrlx : datavaluedPropertyAtom\ swrlx : property =$“xg;centerFrequency”$>$

$< ruleml : var > tr < /ruleml : var >$

$< owlx : DataValue\ owlx : datatype =$“xsd;integer”$>$ 231000000 $< /owlx : DataValue >$

$< /swrlx : datavaluedPropertyAtom >$

This fragment represents an *attribute-value* pair.

As mentioned in Section 2.1.4, we can define policy P as a function $P : R \rightarrow E$ that is a mapping from the domain of requests R onto the domain of effects E, where $E = \{Y, N, NA\}$. Before a policy can be transformed into an MTBDD, it has to be converted into a Boolean expression (parsing). A Boolean expression is composed of atomic Boolean expressions (AEs) combined using the logical operations $\vee$ (logic or) and $\wedge$ (logic and). We provide an example below that explains how a policy is converted into a Boolean expression.

Example. Consider the following spectrum access policy, $P1$. P1: Allow transmission in [255 MHz, 328 MHz] or [2200 MHz, 2290 MHz] frequency bands, if the mode is "Day to Day" or "Special Event" and the transmission power level is less than 25 dBm. Deny transmission if the mode is "Day to Day" and the time is between 11 a.m. and 2 p.m.

We now map each policy constraint to an atomic Boolean expression, AE_i, represented by variable x_i such that: $x_i = 0$ if AE_i is false; $x_i = 1$ if AE_i is true:

x_0 : mode = Special Event

x_1 : mode = Day to Day

x_2 : $TXpower < 25dBm$

x_3 : $11 < time < 14$

x_4 : $255MHz < frequency < 328MHz$

x_5 : $2200MHz < frequency < 2290MHz$

Using the above Boolean encoding, policy $P1$ can be transformed into the function $P : B^n \rightarrow E$, which is a mapping from a vector of Boolean variables, $\overrightarrow{x} = (x_0, x_1, ..., x_{n-1})$, onto the finite set of effects, $E = \{Y, N, NA\}$, where n is the number of unique atomic Boolean expressions in policy $P1$.

After Boolean encoding, the policy $P1$ is transformed into the function:

$$P_1(r) = \begin{cases} Y, & \text{if } ((x_0 \vee x_1) \wedge (x_4 \vee x_5) \wedge x_2) \\ N, & \text{if } (x_1 \wedge x_3) \end{cases}$$

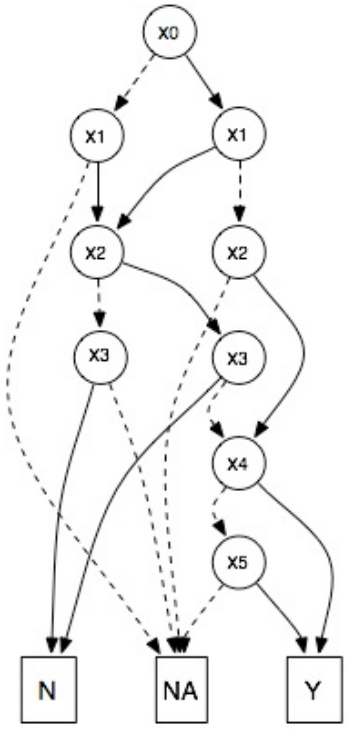


Figure 2.3: An MTBDD representation of policy $P1$

These Boolean expressions are then passed on to the policy converter and the policy preprocessor module.

2.2.2 Policy Converter

The Policy Converter converts each active policy's Boolean expression that is generated by the policy parser into an MTBDD.

The internal (or non-terminal) nodes represent variables of the atomic Boolean expressions and the terminal nodes represent values in the policy's set of effects. Each complete path in the MTBDD, from the root node to one of the terminal nodes, represents one instance of assigning values to the Boolean variables. In other words, the value of the Boolean function, representing a given policy, can be determined for a given variable assignment by following a path down the MTBDD graph from the root node to one of the terminal nodes. Any

given transmission request, r, corresponds to one of the paths. A complete transmission request would correspond to a complete path and an under-specified transmission request would correspond to an incomplete path.

Note that different orderings of the variables may result in different MTBDD representations of different sizes. In this chapter, we use the fixed variable ordering $x_0 < x_1 < ... < x_{n-1}$, where n is the number of variables in the BDD.

An MTBDD representation of policy $P1$ is shown in Figure 2.3. In the figure, the dashed lines denote 0-edges (Boolean variables assigned the value 0) and solid lines denote 1-edges (Boolean variables assigned the value 1). Policy $P1$ is composed of a permissive policy and a prohibitive policy, so it has all three terminal nodes (i.e., Y, N and NA nodes). A single permissive (prohibitive) policy can be represented by a BDD with two terminal nodes: Y (N) and NA nodes.

2.2.3 Transmission Request Interpreter

The PR assumes that the transmission request submitted by the SSR is in the format prescribed by Definition 1. For example, a request that conforms to policy $P1$ of the previous example may take the form $\{(\text{frequency}, v_1), (\text{mode}, v_2), (\text{power}, v_3), (\text{time}, v_4)\}$. In order for the Reasoner component of the PR to compute a response to a transmission request, assignment of values to the Boolean variables of the meta-policy's MTBDD representation needs to be executed. A *meta-policy* is the integration of all currently active policies. The Transmission Request Interpreter evaluates the transmission request and assigns a value to each of the Boolean variables of the meta-policy BDD based on the transmission request. In other words, the Transmission Request Interpreter translates each complete request to a complete path from the root node (the node corresponding to the first variable in the variable ordering) to one of the terminal nodes. An under-specified request is translated as an incomplete path. An example is provided below.

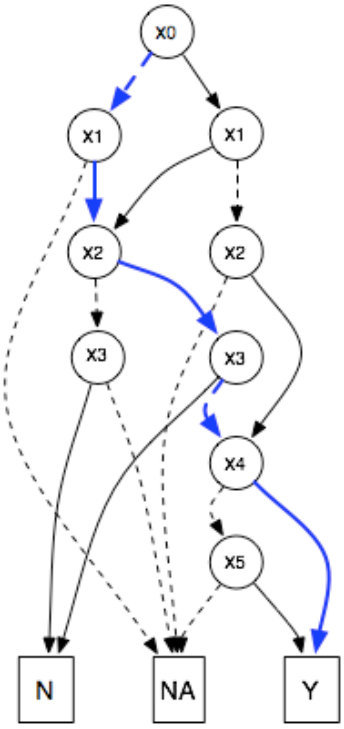


Figure 2.4: The path corresponding to a transmission request.

Example. Consider policy $P1$ of Section 2.2.1 and the following transmission request: $R1 = \{(mode, DaytoDay), (power, 20dBm), (time, 6.00), (frequency, 300MHz)\}$. The Transmission Request Interpreter assigns values to the Boolean variables as follows: $x_0 = 0, x_1 = 1, x_2 = 1, x_3 = 0, x_4 = 1, x_5 = 0$. The path corresponding to this request is shown in Figure 2.4 with bold blue lines. It can be seen that the response to this request is Yes (i.e., transmission request permitted).

2.2.4 Policy Merger

The Policy Merger combines all of the MTBDDs that the policy converter has generated (these represent the set of *active* policies) to build a single *meta-policy* MTBDD. All of the binary and unary FIA operations on policies can be performed on the corresponding MTBDD representations using equivalent graph-theoretic operations. Many efficient operations for

manipulating data in MTBDDs have been proposed in the literature. In this chapter, we use the *Apply* operation [16] to perform the FIA binary operations $\{+, -, \&\}$ and the *Not* operation [17] to perform the FIA unary negation operation $\neg$.

The Apply operation combines two MTBDDs by using a specific binary arithmetic operation. The Apply operation traverses each of the MTBDDs simultaneously starting from the root node down towards the terminal nodes. When the terminal nodes of the MTBDDs are reached, the binary arithmetic operation is applied to the terminal nodes of the two MTBDDs to obtain a terminal node for the resulting combined MTBDD. A variable ordering needs to be specified in order to execute the Apply operation. The following two equations show how Apply combines decision nodes (assuming that $x_i < x_j$ in the variable ordering):

$$\begin{aligned} &\text{Apply}(\text{Node}(x_i, low_1, high_1),\ \text{Node}(x_i, low_2, high_2)) \\ &= \text{Node}(x_i,\ \text{Apply}(low_1, low_2),\ \text{Apply}(high_1, high_2)) \end{aligned} \tag{2.1}$$

$$\begin{aligned} &\text{Apply}(\text{Node}(x_i, low_1, high_1),\ \text{Node}(x_j, low_2, high_2)) \\ &\quad = \text{Node}(x_i,\ \text{Apply}(low_1,\ \text{Node}(x_j, low_2, high_2)), \\ &\qquad\qquad \text{Apply}(high_1,\ \text{Node}(x_j, low_2, high_2))) \end{aligned} \tag{2.2}$$

Equation 2.1 is used for combining two decision nodes with the same variable, while equation 2.2 is used for combining two decision nodes with different variables.

The Reasoner module of our policy reasoner uses the meta-policy MTBDD that the Policy Merger has created to carry out policy reasoning. Once the meta-policy MTBDD has been created, determining a transmission reply in response to a transmission request is straightforward. The Reasoner simply determines the transmission reply by traversing the path in the meta-policy MTBDD dictated by the transmission request. If this transmission request is a "complete" request, then it will dictate a complete path from the root node to one of the terminal nodes—viz, Y, N, or NA node. If the path ends at the Y node, the transmission

request is granted permission; if the path ends at either the N or NA node, the transmission request is denied.

The Policy Merger in our policy reasoner uses a pre-defined rule for resolving policy conflicts. Our current implementation uses the *deny override* rule to resolve conflicts. A conflict occurs when the respective responses of two different policies to a single request are different. Deny override rule denies a request if the request is denied by any of the currently active policies. For instance, suppose that the path specified by a request ends at the N terminal node of a given policy MTBDD. Also, suppose that this MTBDD is merged with other MTBDDs to form the meta-policy MTBDD. In this case, the path specified by the same request in the meta-policy MTBDD will also end at the N terminal node of the meta-policy MTBDD.

When merging multiple policies into one policy, the NA terminal node plays an important role in certain instances. For example, suppose a policy permits a transmission request and a second policy neither prohibits nor permits the same request. After merging the two policies, the correct response to the request is a "Yes" according to the merged policy. However, if the second policy prohibits the transmission request, then the correct response specified by the merged policy is a "No". We need to separate these two cases. In other words, we need to include the NA terminal node in a MTBDD to correctly represent a policy that neither prohibits nor permits a given request. However, it should be noted that there is no difference between the N and NA terminal nodes in the meta-policy MTBDD because all transmission requests whose corresponding paths terminate at either the N or NA node are denied by the Reasoner module. Therefore, we can simply treat the meta-policy MTBDD as a BDD with two terminal nodes (viz Y and N nodes) by combining the N and NA nodes into a single N node. This means that we can use a BDD instead of a MTBDD to represent a meta-policy. The use of a BDD as opposed to a MTBDD is advantageous because it enables us to decrease the size of the meta-policy representation (i.e., smaller number of nodes) and makes available a number of graph-theoretic algorithms that work only on BDDs.

2.2.5 Reasoner

To compute a transmission reply in response to a transmission request, the Reasoner uses the meta-policy BDD and the Boolean interpretation of the transmission request. The Reasoner sends one of the following responses to the SSR:

- **Yes:** This response is given to a *valid* transmission request. The transmission is allowed. The SSR cannot transmit unless it receives such a message.
- **No, invalid TX request:** This response is given to an *invalid* transmission request. The transmission is not allowed, but if the SSR applies a set of appropriate changes to its transmission request, the transmission will be allowed. These changes are called *opportunity constraints*.
- **No, under-specified TX request:** This response is given to an *under-specified* transmission request. The transmission request is incomplete, but will be allowed if the SSR applies the appropriate opportunity constraints to its transmission request.

If a transmission request is under-specified, the Reasoner simplifies the meta-policy BDD by using the RESTRICT algorithm [15] and the values assigned to the AEs specified by the transmission request. RESTRICT(u, j, b) searches for all nodes in the BDD u whose variable value is j and replaces them by their low- or high-child depending on the Boolean value of b. In other words, a node specified by variable j is replaced by a high-child (low-child) if $b = 1$ ($b = 0$). After simplifying the meta-policy BDD using the RESTRICT algorithm, the Reasoner uses the FindPath algorithm to find the opportunity constraints such that if the SSR satisfies the constraints in its transmission request, its transmission request will be permitted by the policy reasoner. The FindPath algorithm is explained in the next section.

When a transmission request is invalid, the path specified by the request ends at the N terminal node of the meta-policy BDD. In such a case, the Reasoner needs to compute the correct opportunity constraints by employing the following procedure: (1) Gradually

backtrack from the N terminal node upwards (towards the root node) until an intermediate node is reached from where a path to the Y node exists; (2) A new path that terminates at the Y node is created by finding a path from the intermediate node to the Y node while keeping the path from the root node to the intermediate node unchanged. The Boolean variables and their values represented by this new path constitute the opportunity constraints. There are several performance issues that need to be considered when determining the new path. We provide more details on how to determine the new path in the next section.

2.2.6 Policy Preprocessor

When merging two BDDs into one BDD, we need to analyze the Boolean expressions represented by the nodes of each BDD prior to merging. Without such analysis, the resulting BDD after merging may contain nodes that represent illogical policy constraints. Such nodes in the meta-policy BDD can cause the Reasoner module to compute illogical opportunity constraints. For example, consider the following two policies:

- $P1$: Allow transmission only in frequency band [233 MHz, 235 MHz].
- $P2$: Allow transmission only in frequency band [250 MHz, 255 MHz].

Suppose we assign the Boolean variables, $X1$ and $X2$, to the frequency bands [233 MHz, 235 MHz] and [250 MHz, 255 MHz], respectively. The BDD representations of $X1$, $X2$, and $X1\&X2$ are shown in Figure 2.5.

Suppose that the BDD representation of $X1\&X2$ represents the meta-policy BDD. In this case, the Reasoner would compute the opportunity constraint represented by the path to the Y terminal node—viz, $X1 = 1$ and $X2 = 1$. It is obvious that this opportunity constraint is illogical, because the transmission frequency cannot be in both bands simultaneously. Existence of such illogical paths can cause the reasoner to return illogical opportunity constraints and hence such illogical paths must be removed from the meta-policy BDD.

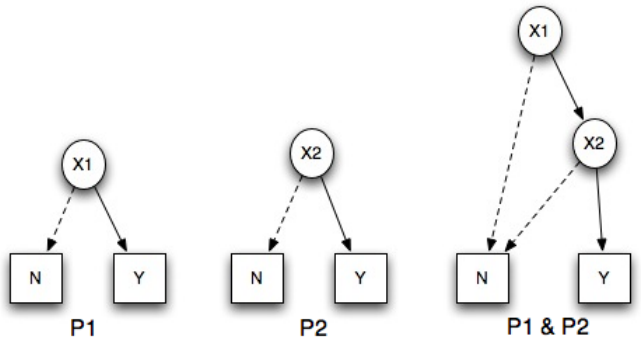


Figure 2.5: BDD representations for $P1$, $P2$ and their combination.

The policy preprocessor *semantically* analyzes the Boolean expressions generated by the Policy Parser module and generates additional constraints (Boolean expressions), which we refer to as *auxiliary rules*, such that, when these auxiliary rules are combined with the meta-policy BDD, they eliminate all the illogical paths from the meta-policy BDD. These auxiliary rules represent the *constraints* for dynamic spectrum access, and the Preprocessor module is thus responsible for enforcing the constraints by eliminating the illegal paths which violate them. For instance, in the previous example, the Policy Preprocessor would generate the rule: "Either $X1$ or $X2$ can be true, but not both.", i.e. $\neg(X1\&X2)$ where $\neg$ and & represent binary negation and the logical AND operation respectively. Then the Policy Converter converts this rule to an MTBDD. The Policy Merger merges this MTBDD with policy MTBDDs so that all of the illogical paths end at the N terminal node in the meta-policy BDD.

2.3 The Reasoning Algorithms

We have devised three different algorithms that are used by the Reasoner module for evaluating the transmission requests and computing opportunity constraints. These algorithms are: the *FindPath* algorithm for evaluating transmission requests and checking if they are under-specified or not, the *GreedyPath* algorithm that computes suboptimal opportunity con-

Table 2.1: Reasoning algorithms parameters.

Variable	Description
Y	Y-terminal node of BDD
N	N-terminal node of BDD
$high(t)$	high child of node t
$low(t)$	low child of node t
$var(t)$	variable of node t
p	the path that is determined by the transmission request
$path$	the array that stores opportunity constraint
$\langle x \rangle$	the value assigned to Boolean variable x
$Size(B)$	number of nodes in BDD B
$c[i]$	the flip cost of the Boolean variable x_i
$MC[i]$	the minimum cost sum of any path between node i and the Y-terminal node

straints for an invalid transmission request, and the *MinCostPath* algorithm that computes optimal opportunity constraints for an invalid transmission request. Table 2.1 summarizes the variables that are used in these algorithms.

2.3.1 FindPath Algorithm

The *FindPath* algorithm utilizes the observation that in BDDs, if a given node is not the N terminal node, it has at least one path leading to the Y terminal node. This algorithm is used for evaluating transmission request via traversing the meta-policy BDD but we can use it for computing a transmission opportunity for an under-specified transmission request. A pseudo-code of the FindPath algorithm is given below. In this pseudo-code, terminal nodes are represented by N and Y. Also, high(t) and low(t) represent high-child and low-child of node t, respectively. The FindPath algorithm finds a path that terminates at the Y terminal node and stores this path in the array $Path$.

To find a path from the root node to the Y terminal node, we apply FindPath algorithm to the root node. The FindPath algorithm traverses a BDD from top to bottom, one level at a time, recursively. Hence, the number of recursive calls to the algorithm is at most n and the algorithm's complexity is $O(n)$, where n is the height of the BDD. Note that n is also

equal to the number of variables in the BDD. This algorithm is guaranteed to find a path from the root node to the Y terminal node, if such a path exists.

Algorithm 1 FindPath Algorithm

Input: BDD, start node.
Output: *Path*.

```
1: FindPath(t)
2: Add node t to array Path
3: if (high(t) = Y or low(t) = Y) then
4:     return True
5: else if (low(t) = N) then
6:     return FindPath(high(t))
7: else
8:     return FindPath(low(t))
9: end if
```

2.3.2 GreedyPath Algorithm

The *GreedyPath* algorithm finds a suboptimal path that leads to the Y terminal node. In the context of opportunity constraints, *optimality* refers to minimum changes to the original transmission request that is provided by the SSR. The GreedyPath algorithm does not guarantee an optimal solution, but many of its output paths are optimal.

Suppose that the path that ends in the N terminal node has length m and is stored in the array $p[0], \ldots, p[m-1]$, i.e., $p[0]$ is the root node, $p[m-1]$ is the N terminal node and the other elements of p represent the other nodes along the path. Also, assume that $\langle x_i \rangle$ is a value assigned to the Boolean variable x_i by the Transmission Request Interpreter during the process of evaluating a transmission request. The notations high(t), low(t), and var(t) represent the high-child, the low-child and the variable of node t, respectively. The GreedyPath algorithm finds a suboptimal path that leads to the Y terminal node and stores the new path in the array *Path*. The policy reasoner returns this array to the SSR as a set of opportunity constraints. The pseudo-code for the GreedyPath algorithm is given below.

Like the FindPath algorithm, the GreedyPath algorithm traverses a BDD from the top to the

Algorithm 2 GreedyPath Algorithm

Input: BDD, p, $\langle x_i \rangle$.
Output: *Path*.

```
1: GreedyPath (t)
2: Add node t to Path
3: if (high(t) = Y or low(t) = Y) then
4:     return True
5: else if ⟨x_var(t)⟩ = 0 then
6:     if (low(t) = N) then
7:         return GreedyPath (high(t))
8:     else
9:         return GreedyPath (low(t))
10:    end if
11: else if (high(t) = N) then
12:    return GreedyPath (low(t))
13: else
14:    return GreedyPath (high(t))
15: end if
```

bottom, one level at a time, recursively. However, unlike FindPath, we apply GreedyPath to a node that is directly connected to the N terminal node by an edge (i.e., node stored in $p[m-2]$). Starting from this node, GreedyPath finds a path towards the Y terminal node such that the value of each Boolean variable in the path is the same as its value assigned by the Transmission Request Interpreter unless this assigned value leads to the N terminal node.

The number of recursive calls to the GreedyPath algorithm is at most n and the algorithm's complexity is $O(n)$, where n is the height of the BDD. The number of variables is also n.

2.3.3 MinCostPath Algorithm

The *MinCostPath* algorithm is used by the Reasoner when it needs to compute *optimal* opportunity constraints in response to an invalid transmission request sent by the SSR. The definition of optimality is described in the previous subsection. Unlike the GreedyPath algorithm, the MinCostPath algorithm is guaranteed to find an optimal path from the root

node to the Y terminal node. The MinCostPath algorithm finds an optimal opportunity constraint by solving the following problem:

Problem 1. *Suppose that $X = (x_1, \ldots, x_n)$ is a sequence of BDD variables and that each variable, x_i, is associated with a constant non-negative cost c_i. In the BDD, find a path from the root node to the Y-terminal node such that the cost sum, $\alpha_1 c_1 + \ldots + \alpha_n c_n$, is minimized, where α_i denotes a Boolean value assigned to x_i.*

By solving the above problem, MinCostPath can be used to compute the opportunity constraints that are optimal in the context of the following cases:

- **Case 1.** Suppose that the parameters of a transmission request are not weighted. In this case, the Reasoner assigns a unit*flip cost* to each of the variables in the meta-policy BDD, i.e. the cost of flipping each Boolean variable x_i is equal to one. Using the MinCostPath algorithm, the Reasoner computes a set of opportunity constraints by finding a path to the Y-node with a minimum cost sum. This path minimizes the *number* of changes to the variable assignment made by the Transmission Request Interpreter. Hence, as a result, the number of transmission parameter changes prescribed by the computed opportunity constraints is minimized. If we assume that the parameters included in the transmission request were optimized by the SSR to meet some performance criteria, it is obvious that it is advantageous to make as few changes to those parameters as possible.

- **Case 2.** Suppose that the SSR assigns a non-negative *changing cost* to each of the transmission parameters based on a pre-defined criterion such as the difficulty of changing a parameter or its importance. For example, the SSR may assign a greater changing cost to the transmission power level compared to the modulation method because of its greater effect on network performance. The reasoner assigns a flip cost value to each variable in the meta-policy BDD that is equal to the changing cost assigned to the corresponding transmission parameter. Using the MinCostPath algorithm, the Reasoner computes a set of opportunity constraints by finding a path to the Y-node with

a minimum cost sum. The resulting path is found by making preferential changes to the variable assignment made by the Transmission Request Interpreter—i.e., making preferential changes to variable assignments in which changing the variables have less cost whenever possible. Using this strategy, the Reasoner can compute a set of opportunity constraints that prescribes changes to the transmission parameters which take into account their weighted importance.

In the MinCostPath algorithm, we assume that the variable ordering is $x_1 < \ldots < x_n$ and the nodes are numbered from the bottom of the BDD, i.e., if $var(x)$ represents the Boolean variable of node x then $var(i) > var(j) \Rightarrow i < j$. We define $Size(B)$ as the number of nodes in BDD B. We define the following arrays used in the algorithm:

- $c[i]$= The flip cost of the Boolean variable x_i.
- $MC[i]$= The minimum cost sum of any path between node i and the Y terminal node.
- $Path[i]$= The Boolean value for variable of node i such that the cost sum of the path that the node is on is minimized.
- $\langle x_i \rangle$ = The value assigned to the Boolean variable x_i by the Transmission Request Interpreter during the process of evaluating a transmission request.

The pseudo-code of the algorithm is given below. In the algorithm, we set MC[0] = $+\infty$— this prevents the algorithm from outputting a path that terminates at the N-node (node 0).

The correctness of this algorithm can be proved by using strong induction.

Proof. Suppose that AP_i denotes the set of all paths between node i and the Y terminal node. We define three functions $R(i)$, $t(u)$ and $C(u, v)$ as follows:

$$R(i) = \min_p \{cost(p) | p \in AP_i\}$$

$$t(u) = \begin{cases} high(u) & \text{if} \quad \langle x_{var(u)} \rangle = 0 \\ low(u) & \text{if} \quad \langle x_{var(u)} \rangle = 1 \end{cases}$$

$$C(u,v) = \begin{cases} c[var(u)] & \text{for} \quad v = t(u) \\ 0 & \text{otherwise} \end{cases}$$

Because all paths from node i to terminal nodes passes through $low(i)$ or $high(i)$, we can define a recursive definition for $R(i)$:

$$R(i) = \min_{child(i)} \{R(child(i)) + C(i, child(i))\} \tag{2.3}$$

where $child(i) \in \{high(i), low(i)\}$. We want to prove $MC[i] = R(i)$ by strong induction on i. We define $R(0) = MC[0] = +\infty$ and $R(1) = MC[1] = 0$.

Assume that for all $i \leq m-1$, we have $MC[i] = R(i)$. We need to prove that $MC[m] = R(m)$. Because the ordering is $x_1 < \ldots < x_n$, we know that $var(m) < var(low(m))$ and $var(m) < var(high(m))$, hence $low(m) < m$ and $high(m) < m$ that results in $MC[low(m)] = R(low(m))$ and $MC[high(m)] = R(high(m))$. Suppose that $\langle x_{var(m)} \rangle = 0$, so we have $t(m) = high(m)$ and consequently $C(m, child(m)) = \begin{cases} c[var(m)] & \text{for } child(m) = high(m) \\ 0 & \text{otherwise} \end{cases}$. From the algorithm, we know that:

$$\begin{aligned} MC[m] &= \min(MC[low(m)], MC[high(m)] + c[var(m)]) \\ &= \min(R(low(m)), R(high(m)) + c[var(m)]) \\ &= \min_{child(m)} \{R(child(m)) + C(m, child(m))\} \\ &= R(m) \end{aligned}$$

The last equality is a result of equation 2.3.

Similarly for $\langle x_{var(m)} \rangle = 1$, it can be proved that $MC[m] = R(m)$. $\square$

Algorithm 3 MinCostPath Algorithm

Input: BDD, $c[i]$.
Output: $Path$.

```
1: MC[0] = +∞
2: MC[1] = 0
3: for i = 2 to Size(B) − 1 do
4:     l = low(i)
5:     h = high(i)
6:     v = var(i)
7:     if ⟨x_v⟩ = 0 then
8:         if (MC[l] ≤ MC[h] + c[v]) then
9:             MC[i] = MC[l]
10:            Path[i] = 0
11:        else
12:            MC[i] = MC[h] + c[v]
13:            Path[i] = 1
14:        end if
15:    else
16:        if (MC[h] ≤ MC[l] + c[v]) then
17:            MC[i] = MC[h]
18:            Path[i] = 1
19:        else
20:            MC[i] = MC[l] + c[v]
21:            Path[i] = 0
22:        end if
23:    end if
24: end for
```

Because the MinCostPath algorithm includes a loop that iterates $Size(B)$ times, the algorithm's complexity is $O(Size(B))$.

2.3.4 Variable Ordering and Reasoning Algorithm Performance

The runtime efficiency of the MinCostPath algorithm is determined by the size of the meta-policy BDD (i.e., number of nodes), whereas the performance of the other two reasoning algorithms (viz FindPath and GreedyPath) is determined by the number of variables. Hence, keeping the BDD manageably small is important for the runtime efficiency of MinCostPath.

Moreover, keeping the BDD small is critical in terms of the required memory capacity when implementing the proposed policy reasoner in practice. It is well known that the variable ordering has a significant impact on the size of a BDD [15]. Finding an optimal variable ordering is an NP-hard problem [18]. An optimal variable ordering results in the smallest BDD size. A good variable ordering can lead to a smaller BDD and faster runtime, whereas a bad ordering can lead to an exponential growth in the size of the BDD. For example, if we use the variable ordering $x_1 < x_3 < \ldots < x_{2n-1} < x_2 < x_4 < \ldots < x_{2n}$ for the Boolean function $f(x_1, \ldots, x_{2n}) = x_1x_2 + x_3x_4 + \ldots + x_{2n-1}x_{2n}$, the BDD needs 2^{n+1} nodes to represent the function. However, using the ordering $x_1 < x_2 < x_3 < x_4 < \ldots < x_{2n-1} < x_{2n}$, the BDD consists of only $2n$ nodes.

Fortunately, spectrum access policies have a certain property that enables us to find an optimal variable ordering in an efficient manner when building a BDD representation of a spectrum access policy[1].

Property 1. *The Boolean function representation of a single cognitive radio policy is a read-once function.*

Read-once functions are Boolean functions in which each variable occurs exactly once.

Formally, a read-once function can be defined as follows. The function $f(x_1, \ldots, x_n)$ is a read-once Boolean function if either (i) $n = 1$ and $f(x_1) = x_1$; or (ii) $f(x_1, \ldots, x_n) = g(x_1, \ldots, x_k) \circ h(x_{k+1}, \ldots, x_n)$, where $\circ$ is a binary operation and f and g are read-once Boolean functions. In [19], the authors proposed an algorithm with complexity $O(n)$ that computes the optimal variable ordering for read-once functions. Using a variable ordering prescribed by this algorithm results in a BDD with $O(n^\beta)$ nodes, where $\beta = log_4^{3+\sqrt{5}} \simeq 1.19424$. Hence, the size of an MTBDD representing a single individual policy is always polynomial with respect to the number of variables. However, when multiple policies are merged into one, the resulting merged policy is likely to *not* satisfy Property 1. Fortunately,

[1]This observation is based on the spectrum access policies presented in the various literature on dynamic spectrum access. Even if a given policy does not have the property specified in Property 1, it can be readily modified to have such a property.

our experience with spectrum access policies indicates that the size of the BDD representation of merged policies is polynomial in *most* cases. In [20], the authors observed that while in the worst case, the number of nodes in an MTBDD is exponential in the number of variables, in practice, the number of nodes is often polynomial or even linear.

As mentioned previously, our policy reasoner can use one of two reasoning algorithms for computing the opportunity constraints for an invalid transmission request—these algorithms are GreedyPath and MinCostPath. The complexity of the GreedyPath Algorithm is $O(n)$, where n is the number of variables, so the variable ordering does not affect the performance of this algorithm. The complexity of the MinCostPath algorithm is $O(Size(B))$, where $Size(B)$ is the number of nodes in the BDD, so the variable ordering impacts the performance of this algorithm. It is obvious that if $Size(B) >> n$, MinCostPath is much slower than GreedyPath . Recall that MinCostPath computes a set of *optimal* opportunity constraints, whereas GreedyPath computes a set of *suboptimal* opportunity constraints. Therefore, the Reasoner needs to make an appropriate tradeoff between execution speed and optimality of the opportunity constraints when considering which reasoning algorithm to employ.

To make the aforementioned tradeoff, the Reasoner may adopt an algorithm that uses a predetermined threshold to decide which reasoning algorithm to employ. One such algorithm is specified in Algorithm 4. This algorithm instructs the Reasoner to use the GreedyPath algorithm when the size of the meta-policy BDD is greater or equal to the threshold, η, and instructs the Reasoner to use the MinCostPath algorithm otherwise. One would need to determine the threshold value based on a cognitive radio's processing capability and memory capacity.

2.4 Experimental Evaluation

We implemented BRESAP in Java. The policy reasoner has the following four modules: Policy Converter module, Policy Merger module, Transmission Request Interpreter module

Algorithm 4 Hybrid Algorithm

Input: BDD, η.
Output: *Path*.

```
1: if (Size(B) ≥ η) then
2:     Use GreedyPath
3: else
4:     Use MinCostPath
5: end if
```

and Policy Reasoner module.

The processing tasks performed by the policy reasoner can be divided into two types: *offline* tasks and *online* tasks. These two types of processing tasks are described below.

- **Offline tasks:** Offline tasks can be carried out while initializing the radio or whenever the radio downloads a new policy and hence we consider the processing involved in this module to be less time critical. Converting the policies and creating the meta-policy BDD are from this type.
- **Online tasks:** Online tasks are time critical and affect the performance of policy reasoner. Interpreting transmission requests and reasoning are from this type.

For the purpose of evaluation we generated frequency, time and radio location dependent SWRL policies. The number of policies was varied from five to fifty while varying the number of variables (or AEs) in each of the policies from five to twenty. The experiments were conducted on Intel Core 2 CPU 1.86GHz machine with 3GB RAM.

2.4.1 Offline Processing

Three tasks in the policy reasoner carried out offline: converting the SWRL policies into Boolean functions using the JDOM parser (parsing), converting the Boolean functions into MTBDDs and generating the meta-policy BDD by merging the individual MTBDDs.

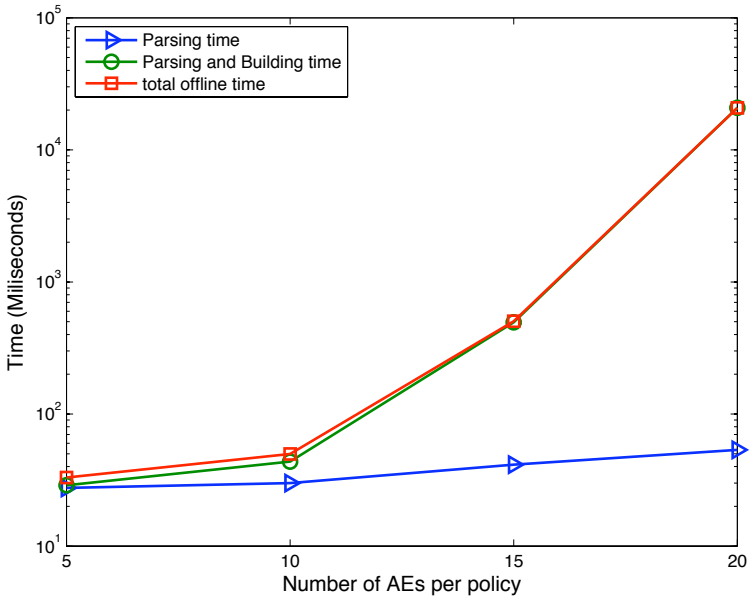


Figure 2.6: Offline processing time vs. number of AEs

Figure 2.6 shows the performance of offline tasks in a semi-logarithmic format. In this figure we plot the time taken to parse a policy into a Boolean function, build an MTBDD from the Boolean function and then combine this MTBDD with the existing meta-policy BDD. The curve with the triangle markers shows the time taken by the policy converter module to parse a policy when the number of AEs is varied from 5 to 20; the curve with the circle markers shows the cumulative time taken till the generation of the MTBDD; the curve with the square markers demonstrates the cumulative time taken till the combination of the MTBDD with the existing meta-policy BDD. This figure also shows the relative performance of the components of the offline section. We can see from the figure that the parsing a policy and merging it with the meta-policy BDD take polynomial time while generating the MTBDD is exponential in time.

We further analyzed the performance of merging MTBDDs and generating the meta-policy BDD. Figure 2.7 shows the time taken to generate the meta-policy BDD when the number of policies is varied from 5 to 50 and the number of AEs in each of the policies is varied from

5 to 20.

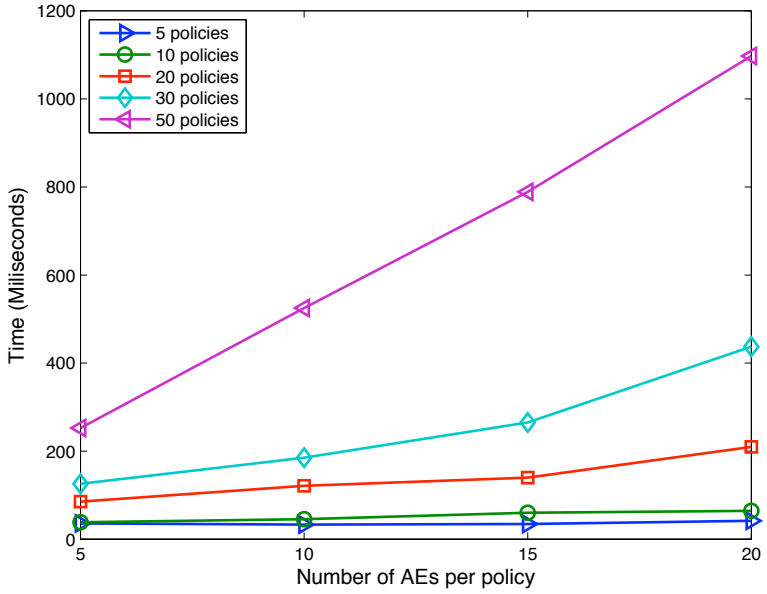


Figure 2.7: MTBDD merging time vs. number of AEs

2.4.2 Online Processing

The time taken for reasoning and returning the constraints depends on the number of AEs in the policies and the type and complexity of the policies that have been used to create the meta-policy BDD. In order to evaluate the performance of the Reasoner, we divided the policies into two sets—viz homogeneous permissive policies and heterogeneous policies. The first set contains only permissive policies while the second set contains permissive as well as prohibitive policies.

We observed that in general, the type of policies that need to be merged affects the size of the meta-policy BDD. Specifically, the merging of homogeneous policies results in a smaller meta-policy BDD as compared to the size of the meta-policy BDD resulting from merging heterogeneous policies.

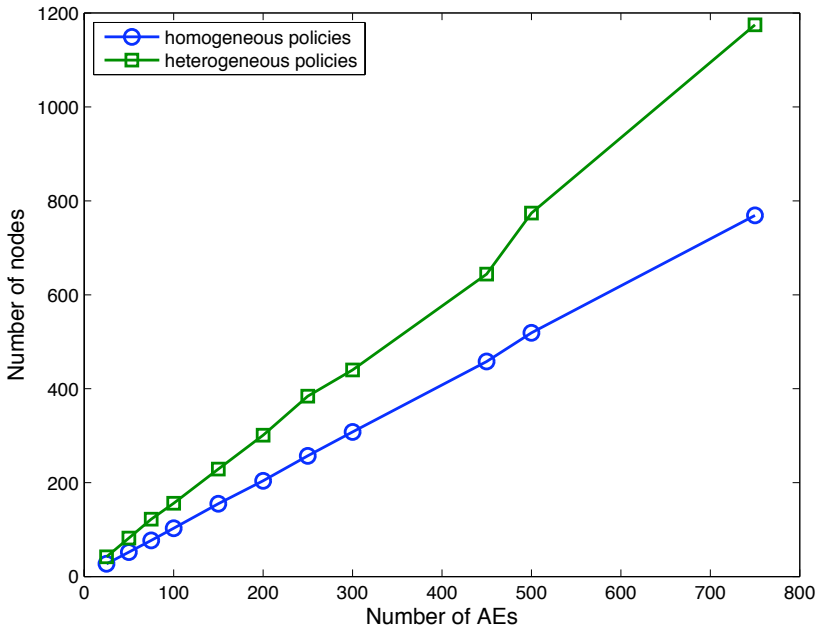
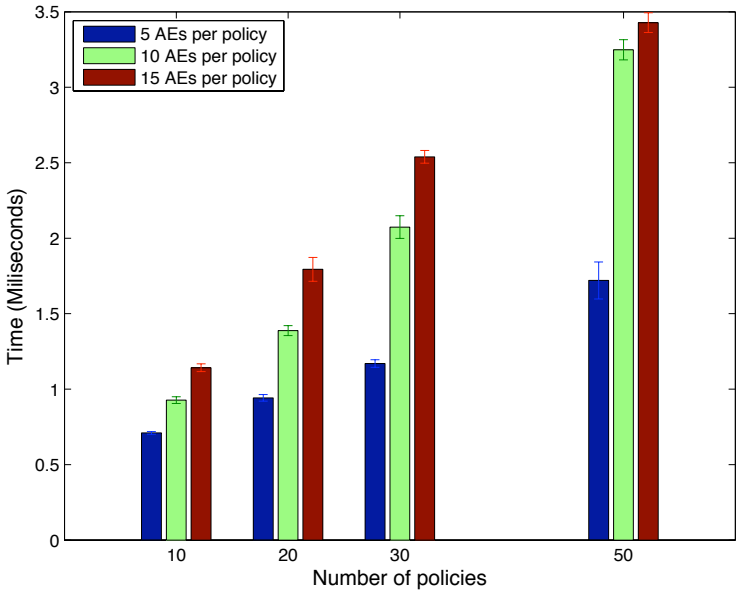


Figure 2.8: Size of meta-policy BDD vs. number of AEs

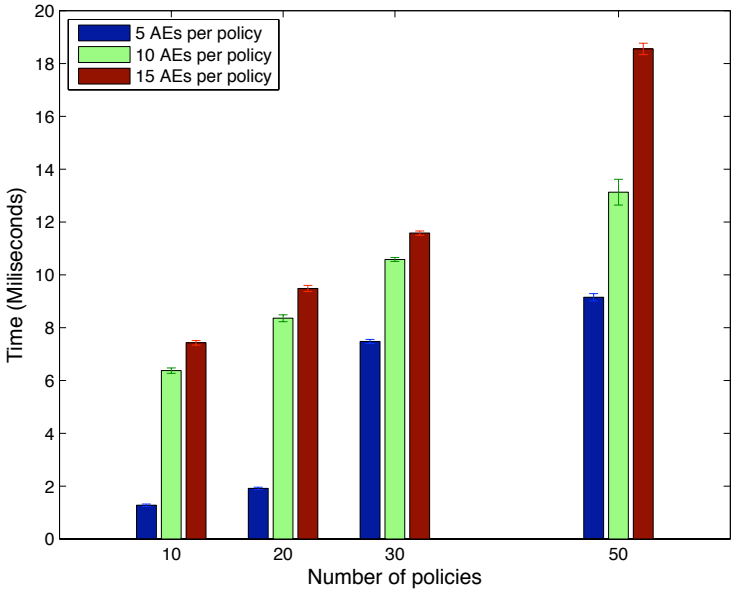
Figure 2.8 shows the relation between the size of the meta-policy BDD (i.e., number of nodes) and the total number of AEs in the policies used to create the meta-policy BDD. From the figure, we can observe that the size of the meta-policy BDD is polynomial in the number of AEs. This result is consistent with the authors' observation given in [20] in which the authors stated that while in the worst case, the size of an MTBDD is exponential in the number of variables, in practice, the size is often polynomial or even linear.

Figures 2.9(a) and 2.9(b) show the time required for the online processing tasks when the PR is dealing with homogeneous and heterogeneous policies, respectively. From the figures, we can observe that executing the online processing tasks for the set of heterogeneous policies takes significantly more time compared to executing the same tasks for the homogeneous policies.

Figure 2.10 compares the running time of the reasoning algorithms, GreedyPath and MinCostPath. The results were obtained from running these two algorithms on 30 heterogeneous



(a) Homogeneous permissive policies



(b) Heterogeneous policies

Figure 2.9: Online processing time for (a) homogeneous and (b) heterogeneous policies

policies.

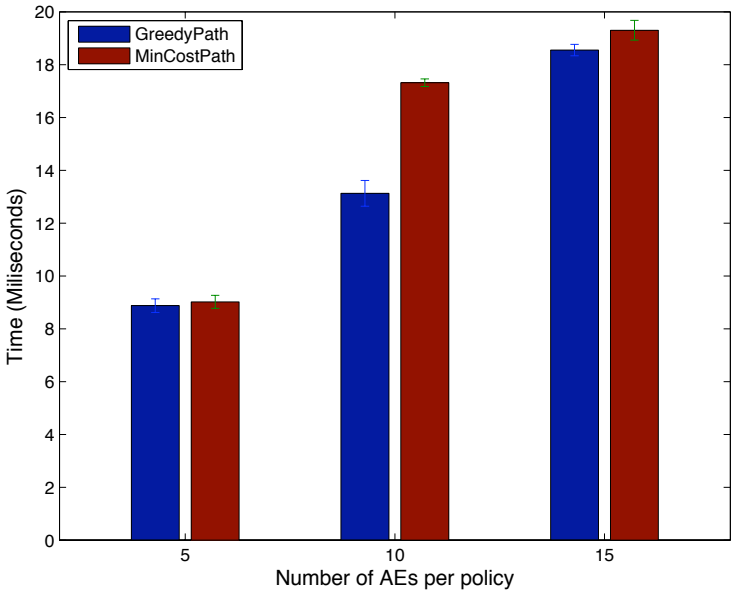


Figure 2.10: Running time of reasoning algorithms vs. number of AEs per policy

2.5 Summary

In this chapter, we described the design and implementation of a novel policy reasoner. The proposed policy reasoner uses multi-terminal binary decision diagrams (MTBDDs) to represent, interpret, and process policies in an efficient manner. It uses a set of efficient graph-theoretic algorithms to translate policies into MTBDDs, merge policies into a single meta-policy, and compute opportunity constraints. The proposed policy reasoner has the capability to process either underspecified or invalid transmission requests sent by the system strategy reasoner. We have also proposed three different graph-based algorithms for computing the opportunity constraints. We have also demonstrated that spectrum access policies can be processed efficiently by converting the policy reasoning problem into a graph-based Boolean function manipulation problem. This work takes a small step forward in addressing the problem of designing policy reasoners that can correctly process complex policies while satisfying the stringent spectrum access control requirements of the spectrum stakeholders.

Chapter 3

Ontology-based Spectrum Access Policies for Policy-based Cognitive Radios

In chapter 2, we introduced BRESAP, a BDD-based policy reasoner for processing spectrum access policies. BRESAP translates each constraint in every spectrum policy into a boolean expression and uses the MTBDD to organize these boolean expressions. MTBDDs are combined to form a single meta-policy BDD, and the policy reasoner uses this BDD during the transmission request evaluation process. The major drawback of BRESAP is that it predefines the attributes of the spectrum policies and hard-wires them into the spectrum policy reasoner. BRESAP cannot process policies whose attributes have not been predetermined in advance. BRESAP can process such policies only after modifying the reasoning software itself.

To address these drawbacks, in this chapter, we describe the design of a policy reasoner that processes *ontology-based spectrum policies*. The main advantage of using ontology-based policies is that the policy reasoner can understand and process any spectrum policies authored by any organization by relying on the spectrum ontologies. In our implementation,

the spectrum ontology defines the various dynamic spectrum access concepts, models the domain of DSA networks in a machine-understandable manner, and uses SWRL rules to represent spectrum policies. Unfortunately, ontological reasoning needed to process ontology-based spectrum policies incurs greater computation overhead compared to non-ontological reasoning. This drawback can be a critical one as it can impede a CR from meeting its real-time performance requirements. We have carried out a number of experiments, using our implementation, to evaluate whether a radio controlled by ontology-based policies can meet its real-time performance requirements. Based on our experimental results, we propose a set of guidelines for the design of ontology-based spectrum access policies.

Using ontology-based spectrum policies has several other advantages, such as facilitating policy management, flexible knowledge representation, interoperability, flexible querying, and self-awareness, which we discuss in Section 3.2. Unlike simple rule-based policy reasoners such as BRESAP that predefine the attributes of the spectrum policies and hard-wires them into the spectrum policy reasoner, our ontology-based policy reasoner can process policies whose attributes have not been predetermined in advance, without modifying the reasoning software itself.

Our policy reasoner implementation is able to handle all ontology operations, including ontology-consistency checking and ontology information editing (i.e., adding, deleting, and updating). The policy reasoner uses a *tableau-based algorithm* [21] to carry out ontology reasoning, and the *Rete algorithm* [22] to process spectrum policies and evaluate transmission requests. We modified the Rete algorithm so that it can compute *opportunity constraints*.

3.1 Overview of Ontologies

Ontology refers to a system of categories that describe a particular vision of the world. Ontology relies on a certain philosophical view and is independent of natural languages. In computer and information sciences, ontology is a formal representation of knowledge and

shared vocabulary about a particular domain of interest. It contains a set of concepts in that domain and relationships between these concepts, which can be used to describe and model this particular domain in a machine understandable manner [23]. Ontology is widely used in the realm of artificial intelligence, semantic web, system engineering, biomedical informatics, library science, etc.

A typical ontology consists of the following components:

- **Individuals:** Individuals are the basic components of an ontology. An individual in an ontology can be any concrete object or abstract object.
- **Classes:** A class is an abstraction of objects that have similar features. For example, *Waveform* is a class with individuals that are particular waveforms. Classes are organized into a hierarchy of classes that indicates relationships between the classes.
- **Properties:** Properties are attributes, features or parameters that an object or a class in an ontology can have, such as carrier frequency of a waveform, number of symbols in an alphabet, etc.
- **Relations:** Relations describe how objects can be related to other objects in an ontology. For example, a waveform represents a sequence of symbols from an alphabet; in this case, *represent* is a relation that links waveforms to symbol sequences.
- **Rules:** Rules are statements that describe the logical inferences that can be drawn from an assertion in the pattern of antecedent implicating consequent. For example, the following rule is a rule in the form of an *if-then* statement: "If a frame belongs to the SDLC (Synchronous Data Link Control) protocol, then the address field has 8 bits".

When an actual ontology is built using the above components, it is important to check the consistency of the ontology and make sure there are no conflicts in the ontology. Tableau based reasoning algorithms are widely used to check the consistency of the ontologies. In

such algorithms, a set of tableau expansion rules are applied to ontology data iteratively until no rule is applicable. Such operations can be time consuming, especially when the ontology is large. The details on the ontology reasoning process will be discussed later in Section 3.3.

3.2 Advantages and Challenges of Ontology-based Spectrum Policies

Ontology-based policies rely largely on the expressive features of Description Logic languages, such as OWL (Web Ontology Language), to classify contexts and policies, thus enabling deductive inferences and static policy conflict resolution. In contrast, rule-based policies take the perspective of logic programming to encode the axioms and rules in a clear way [4]. Moreover, a rule-based approach facilitates the straightforward mapping of policies to lower-level enforcement mechanisms thanks to its concise and understandable syntax. But a simple rule-based policy reasoner (e.g., BRESAP) has a number of shortcomings compared to a more complex ontology-based reasoner. For example, consider the following two policies:

Policy 1. *Permit transmission in [255 MHz, 328 MHz], if the modulation type is GMSK.*

Policy 2. *Prohibit transmission in [255 MHz, 328 MHz], if the modulation type is FSK.*

A rule-based policy reasoner would not detect a conflict between these two policies. But an ontology-based reasoner can recognize that GMSK (Gaussian Minimum Shift Keying) is a subclass of FSK (Frequency Shift Keying), so there is a conflict between these two policies. In other words, since an ontology-based reasoner is able to recognize subclass relations, it can identify and resolve policy conflicts that a rule-based policy reasoner cannot.

Using ontology-based spectrum access policies has several other advantages as discussed below.

Policy management: Ontologies can provide important supplemental information to sim-

plify policy management and composition. In other words, ontologies facilitate the specification of complex regulatory policies for spectrum access.

Flexible knowledge representation: Ontologies can be used as the knowledge representation language for decision-making systems such as policy reasoners. This enables reuse of domain knowledge in the context of policies, and facilitates the analysis of domain knowledge. It is important for cognitive radios to represent knowledge so it can be used in the most flexible way, because they are required to react to the circumstances they have not seen before [24].

Interoperability: Ontologies facilitate the sharing of the policy structure among diverse software agents [5]. If different software agents use the same set of well defined terms for describing the domain and data of the spectrum access policies, it will be easier for them to "talk" to one another.

Flexible querying: Similar to humans, intelligent agents need to be able to answer queries from their users and other agents. For agents to formulate such queries, they must understand a common language, formally defined in syntax and semantics. Using ontologies, information can be queried, and such queries can be answered without having any explicit preprogrammed monitoring capability [24].

Self-awareness: A cognitive radio should be aware of its own capabilities and reflect on its own behavior. Using ontologies, radios can understand their own structure and modify their operation characteristics at runtime based on this understanding.

As discussed above, ontology-based policy reasoning has a number of advantages when used for policy-based cognitive radios. However, it also introduces a number of challenges not shared by most of the other application areas of ontologies. The main challenge in using ontology-based spectrum access policies is meeting the real-time processing requirements of a cognitive radio. Real-time processing demands higher performance for inference and reasoning than an interactive application. In addition, the knowledge base of a cognitive radio includes state information that is continually varying. This is in contrast with the

static knowledge bases employed by most ontology-based reasoning systems.

3.3 System Design

In this section, we first briefly describe the design requirements of a policy-based cognitive radio that employs an ontology-based policy reasoner. Then, we introduce the architecture of our radio system and its core component, the ontology-based policy reasoner.

3.3.1 System Design Requirement

Cognitive radios have stringent real-time requirements. For example, XG radios need to satisfy the 100 ms channel evacuation time [3]. One way of satisfying real-time requirements is to tightly control the timing of software execution. Introducing ontology into the policy reasoning process will certainly cause additional runtime overhead and hinder the timing control of the software execution. This means that the ontology-based policy reasoner needs to be highly optimized to minimize the runtime overhead.

The design methodology of existing software defined radio architectures suggest that software portability will likely be one of the primary requirements that will drive the hardware and software architectures of most software defined radios and cognitive radios. The importance of portability suggests that ontology-based policy reasoning systems that require support from specific hardware, operating system, or any other part of the computing system, other than the reasoning engine, may have limited utility.

Another design requirement of an ontology-based policy reasoner is the ability to handle a dynamic knowledge base that is continually varying. Unlike the other ontology-based reasoning systems that use static knowledge bases, an ontology-based policy reasoner needs to handle a continuously changing knowledge base.

3.3.2 Overall System Architecture

Figure 3.1 illustrates the architecture of the cognitive radio system that our ontology-based policy reasoner is a part of. Our policy reasoner was designed to work as a component of such a system. The system consists of six main components: radio hardware and software, sensors, system strategy reasoner, reasoning interface, ontology-based policy reasoner, and a remote ontology and policy server. The role of the system strategy reasoner in our system is the same as that in the XG radio, which was described in section 2.1.1. The functionalities of the rest of the components are described below.

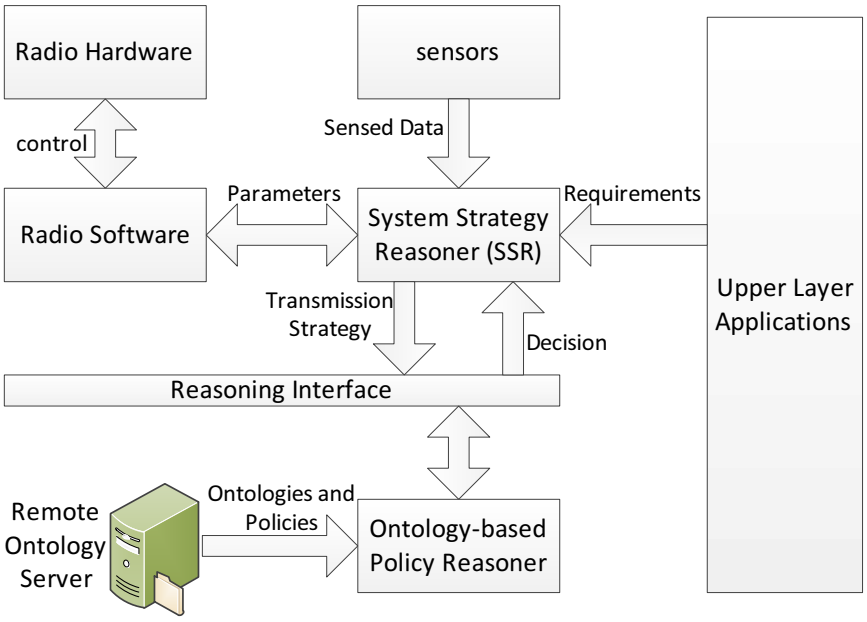


Figure 3.1: Architecture of the cognitive radio system.

- **Radio Hardware and Software:** The radio software receives parameter configurations from the SSR, establishes the datapath and corresponding transmission parameters, and sends control message to the radio hardware in order to transmit the waveforms.

- **Reasoning Interface:** The reasoning interface receives transmission strategies from the SSR, translates them into a transmission request and sends it to the policy reasoner.

The reasoning interface also receives the transmission replies from the policy reasoner and forwards them to the SSR.

- **Ontology-based Policy Reasoner:** The ontology-based policy reasoner carries out all of the spectrum ontology and policy related tasks, including loading and processing spectrum ontologies and policies, checking the consistency of spectrum ontologies, etc. In addition, the policy reasoner evaluates the transmission request against the currently active spectrum policies to check its conformance with the policies. The results of the evaluation, in the form of a transmission reply (or transmission decision), is sent to the reasoning interface;
- **Remote Ontology and Policy Server:** The remote server stores and manages the spectrum ontologies and policies created by various spectrum regulatory entities.

When the cognitive radio is turned on, its components perform various initialization tasks. The ontology-based policy reasoner first loads the spectrum ontologies and the spectrum policies from the ontology and policy server. After loading is finished, the reasoner translates the spectrum ontologies and policies into forms amenable to processing, and then checks the consistency of the spectrum ontologies. After all of these initialization tasks have been executed, the radio is ready for transmitting waveforms.

When a waveform needs to be transmitted, the upper layer application sends the transmission requirements to the SSR. In response, the SSR collects spectrum availability information from the sensors and formulates a transmission strategy using the spectrum availability information and the transmission requirements. The SSR sends the transmission strategy to the ontology-based policy reasoner in the form of a transmission request via the reasoning interface. The policy reasoner checks whether the transmission request conforms to the loaded spectrum policies and then formulates a transmission reply (decision). If the transmission request conforms to the policies, then the policy reasoner generates a transmission reply indicating transmission approval; otherwise, it generates a reply indicating transmission denial.

In the latter case, the policy reasoner also generates *opportunity constraints*—these constraints are specifications of missing transmission parameters (in case of an under-specified transmission request) or revised transmission parameters (in case of an invalid transmission request). If the transmission is approved, the SSR sends the transmission parameter configuration to the radio software, and the radio software sends control messages to the radio hardware to begin waveform transmission.

3.3.3 Ontology-based Policy Reasoner

We employ *Pellet* [25] as the ontology/policy reasoning engine with some modifications to enable the computation of opportunity constraints. Figure 3.2 illustrates the reasoner's architecture. The policy reasoner has three main components: ontology and policy loader, ontology reasoner, and policy reasoner. In the rest of this section, we will describe each of these components and the algorithm for computing the opportunity constraints in details.

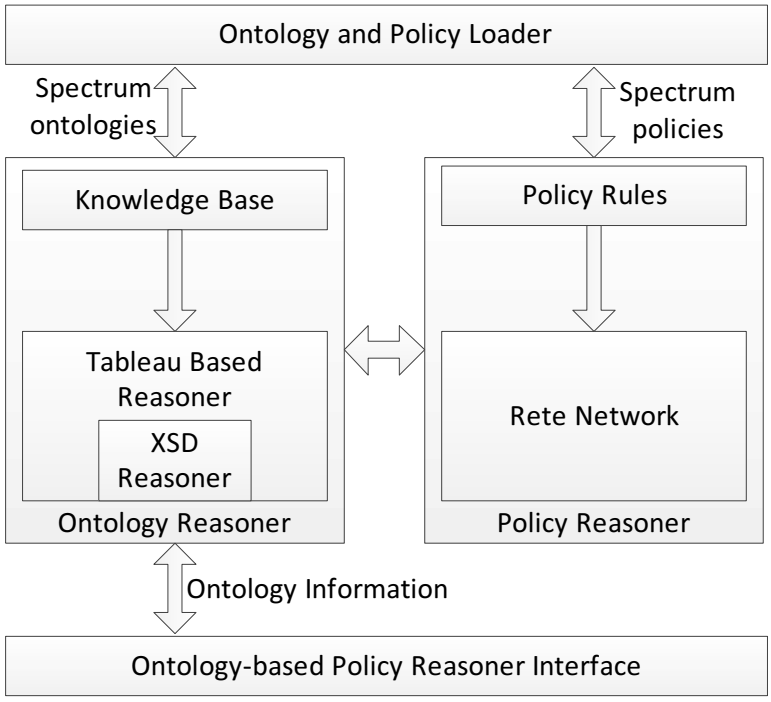


Figure 3.2: Architecture of the ontology-based policy reasoner.

3.3.3.1 Ontology and Policy Loader

The ontology and policy loader is responsible for loading spectrum ontologies and active spectrum policies from the remote spectrum ontology and policy server. After the ontologies and policies are loaded onto the radio, the ontology and policy loader parses them into internal data structures and separates the ontologies and the policies. The spectrum ontologies are fed into the ontology reasoner and the reasoner performs the necessary processing. Then the spectrum policies are sent to the policy reasoner and the policy-related processing is performed.

3.3.3.2 Ontology Reasoner

Once the ontology reasoner receives spectrum ontology information from the ontology and policy loader, it translates the ontology information into ontology facts and axioms. These ontology facts and axioms construct the knowledge base of the domain modeled by the spectrum ontologies. This knowledge base is fed to the tableau-based reasoner to check the consistency of the ontologies stored in the knowledge base. An ontology is consistent if there is an interpretation that satisfies every fact and axiom in this ontology. Such an interpretation is called a *model* of the ontology [25]. The tableau based reasoner runs the tableau-based reasoning algorithm that checks the consistency of ontologies. The tableau-based reasoning algorithm first constructs an initial completion graph from the knowledge base. The nodes in the completion graph represent individuals and literals in the ontology, and the directed edges between nodes represent the property-value assertions related to the individuals and literals in the ontology. Each node in the completion graph is associated with a certain type of an ontology class or a numeric data type. A set of tableaux expansion rules is applied to the completion graph repeatedly until either a clash occurs or there exists no more tableaux expansion rules to apply. The former case implies that the ontology is inconsistent, whereas the latter case implies that the ontology is consistent and a model of the ontology exists [25]. The tableau based reasoner also contains a XSD (XML Schema Definition)

datatype reasoner, which can reason with XSD datatypes [26]. During the tableau-based reasoning process, the XSD datatype reasoner is invoked to determine the consistency of the datatype when a literal node is found by the tableau-based reasoner in the completion graph. The details of the tableau-based reasoning algorithm for ontology consistency checking are beyond the scope of this dissertation, and we will not present them here.

Checking the consistency of the ontology is important, because if the spectrum ontologies are inconsistent, then conflicts will exist in the ontology definitions. When the policy reasoner evaluates the transmission requests based on such inconsistent spectrum ontologies, it may produce incorrect transmission replies. The ontology consistency checking needs to be carried out whenever there is a change to the spectrum ontologies. Checking the consistency of the ontology is important, because if the spectrum ontologies are inconsistent, then conflicts will exist in the ontology definitions. When the policy reasoner evaluates the transmission requests based on such inconsistent spectrum ontologies, it may produce incorrect transmission replies. The ontology consistency checking needs to be carried out every time the spectrum ontologies are changed.

3.3.3.3 Policy Reasoner

The policy reasoner evaluates transmission requests against the active spectrum policies using the Rete algorithm and related Rete networks. The Rete algorithm is an efficient pattern matching algorithm for implementing production rule systems. It sacrifices memory for increased execution speed. This algorithm constructs a data flow network (i.e., Rete network) to represent the rules [22].

A Rete network has three elements: working memory, alpha network and beta network:

- **Working Memory (WM):** The working memory stores all the asserted facts that will be evaluated against the rules represented by the Rete network. The working memory contains several working memory elements (WMEs), and each working memory element

stores a single fact. In most of the cases, the asserted facts are represented by a three tuple, in the form of $<$ *subject predicate object* $>$.

- **Alpha Network:** The alpha network of a Rete network is a discrimination sub-network that selects facts stored in WMEs. It carries out simple conditional tests on the fact attribute values against constant values or pattern matching tests on the patterns of facts stored in WMEs. Each node in an alpha network represents a certain condition or pattern in the rules. For example, if the pattern represented by an alpha node is $<$ *? color red* $>$, then all the objects that are red pass the pattern test carried out by this alpha node, and they will be passed to subsequent alpha nodes if there is any. Different sets of alpha nodes are connected together to form different paths that represent different conditions described in the rules. Facts that pass all the alpha nodes in a path are stored in the corresponding alpha memory, which connects the alpha network and the beta network.

- **Beta Network:** The beta network consists of beta nodes and corresponding beta memories, and mainly performs join operations between different WMEs. Every beta node has two inputs. One input is connected to the corresponding alpha memory of a certain alpha node path (a condition described in the rules), and the other input is connected to other beta memories. Hence, beta nodes also perform the role of connecting different rule conditions (alpha node paths) together to form different rules. Each beta node performs the join operation to merge the WMEs coming from the two inputs into a single list of WMEs and stores it directly into the corresponding beta memory. This list of WMEs stores the facts that partially match the corresponding rule. Such join operations are carried out repeatedly by the beta nodes, and the resulting WME list travels the beta network in a downward direction. When the WME list reaches the terminal node of a rule, this means that all of the necessary conditions have been combined together to form the required rule and that the list contains all of the facts that satisfy that rule.

Spectrum policies have the same role as the rules in a production rule system. When the policy reasoner receives the spectrum policies from the ontology and policy loader, it first parses the spectrum policy rules into several policy conditions, and then, uses these policy conditions to construct the Rete network that is used later for evaluating transmission requests.

The following example shows the Rete network construction process for a simple spectrum access policy.

Policy 3. *Allow transmission if the transmission center frequency is 240 MHz and the bandwidth is 10 MHz.*

This spectrum policy is translated into an internal policy rule structure that has the following policy conditions:

- TransmissionRequest(?TranReqVar [1])
- CenterFrequency(?TranReqVar, ?ValueVar)
- Bandwidth(?TranReqVar, ?ValueVar)
- AllowTransmissionRequest(?TranReqVar)

The first three conditions constitute the body part of the spectrum policy rule, and the last condition corresponds to the conclusion part. Using these four policy conditions, a Rete network, which represents a policy, can be constructed. Each condition in the body part of the spectrum policy rule is represented by an alpha node in the Rete network. Beta nodes are used to connect all the alpha nodes to form the Rete network. The policy condition that corresponds to the conclusion part of the spectrum policy is placed in the last beta node, which acts as the terminal node of the Rete network. Figure 3.3 shows the resulting Rete network that represents the spectrum policy used in this example.

[1] A string of characters that starts with a question mark is a variable

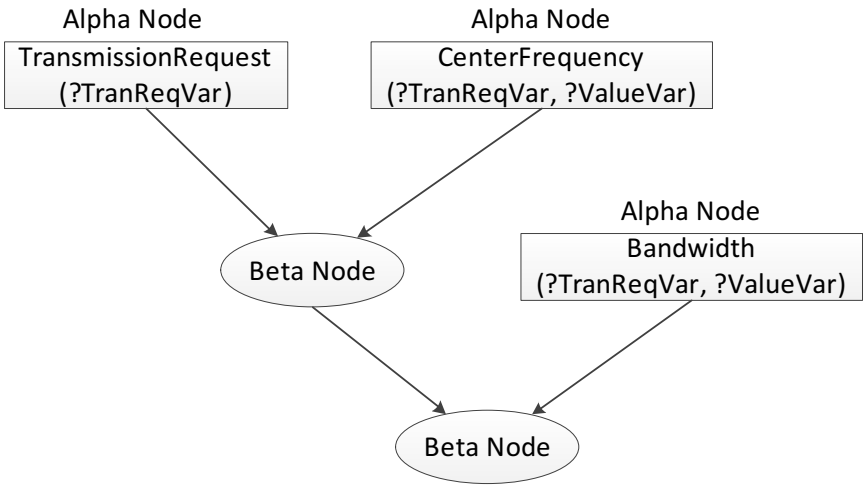


Figure 3.3: Rete network for the example spectrum policy.

3.3.3.4 Evaluating Transmission Requests

The transmission request evaluation process is carried out by the ontology-based policy reasoner in two phases. In the first phase, the patterns of a transmission request are matched against the patterns of the spectrum policies using the Rete algorithm. The ontology-based policy reasoner treats a transmission request as an ontology individual associated with the ontology class, *TransmissionRequest*, in the spectrum ontology. A transmission request has a set of properties, such as center frequency, bandwidth, transmission power, etc. These properties associated with the transmission request are each stored as a three-tuple in the ontology-based policy reasoner. When the pattern matching process begins, these three-tuples are fed to every alpha node in the alpha network to perform a pattern matching test. The tuples that pass the pattern matching test are stored in the corresponding alpha memory. A beta node connected to an alpha memory and previous beta memory performs join operations (i.e. combining) on the tuples stored in the two kinds of memories, and forwards the results of the join operations to the next beta node. The first phase of the transmission request evaluation process ends when the results of the join operations reach the last beta node (i.e., terminal node) and are non-empty. The individuals contained in the results match all the patterns described in the spectrum policies.

Algorithm 5 Transmission Request Evaluation

Input: An individual of a transmission request, T.
Output: A transmission reply, D, and opportunity constraints, $List_{OC}$, if needed.

```
1: rete = construct_rete_network()
2: Set_tuples: a set of tuples that describe T
3: Set_tuples = compile_facts(T)
4: for tuple in Set_tuples do
5:    result = rete.match_pattern(tuple)
6:    if result ≠ "matched" then
7:       rete.get_opportunity_constraint(List_OC)
8:       D = "not allowed"
9:       return
10:   else
11:      Set_matched.add(tuple)
12:   end if
13: end for
14: for tuple in Set_matched do
15:   if tuple.has_numerical_property_value() then
16:      result = compare_with_policies(tuple)
17:      if result ≠ "conform" then
18:         get_opportunity_constraint(List_OC)
19:         D = "not allowed"
20:         return
21:      end if
22:   end if
23: end for
24: D = "allowed"
25: return
```

The completion of the first phase indicates that the patterns of the transmission request conform to the policy's patterns. The second phase is needed to test whether the values of the transmission request properties (e.g., center frequency, transmission power, etc.) conform to the conditions prescribed in the spectrum policies. The ontology-based policy reasoner invokes particular data type converters to convert the transmission request property values into related data types, and compares these values to the values given in the spectrum policies. Once all the comparisons for the value conditions described in the spectrum policies have been completed, the legality of the transmission request can be determined.

When a transmission request is declined, the ontology-based policy reasoner computes the spectrum opportunity constraints in addition to the transmission reply. Whenever a pattern matching test in the first phase or a value comparison test in the second phase fails, the unsatisfied rule pattern and the corresponding spectrum policy are recorded. After the evaluation process is finished, the recorded spectrum opportunity constraints are sent to the SSR. A pseudo code of the algorithm for evaluating a transmission request is given by Algorithm 5.

3.4 System Prototype Implementation

We have implemented a prototype of a software-defined radio whose transmission behavior is controlled by ontology-based spectrum policies. In this section, we discuss the implementation of the critical components of the radio system.

3.4.1 Spectrum Ontology and Policy Documents

The spectrum ontologies are implemented using OWL DL (Web Ontology Language Description Logic) variant of standard OWL web ontology language [27] in the OWL/RDF syntax. The OWL web ontology language is a semantic markup language based on XML that can represent and share ontologies on the Web. OWL is an extension of RDF (Resource Description Framework) in vocabulary and it is advocated by W3C as a standard web ontology language.

The spectrum policies are written in the SWRL [14]. SWRL combines OWL DL and OWL Lite variants of the OWL web ontology language with the Unary/Binary Datalog RuleML sublanguage of the Rule Markup language (RuleML). With the features of RuleML language, SWRL can express horn-like rules. Since SWRL also combines the OWL DL variant, the

horn-like rules can thus be integrated into OWL ontologies using SWRL [2].

The spectrum ontology and policy documents stored in the spectrum ontology and policy server are organized as follows. Spectrum ontologies defined by each spectrum regulatory agency are stored in a separate document, and the spectrum ontologies can refer to each other to construct complicated spectrum ontologies using the *import* feature provided by the OWL web ontology language. Spectrum policies created by different spectrum organizations are stored in separate policy documents, and the policy documents can also refer to spectrum ontologies to acquire ontology information using the *import* feature. The component-based structure described above facilitates the management of the spectrum ontologies and policies. Figure 3.4 illustrates how the spectrum ontology and policy documents interact with one another.

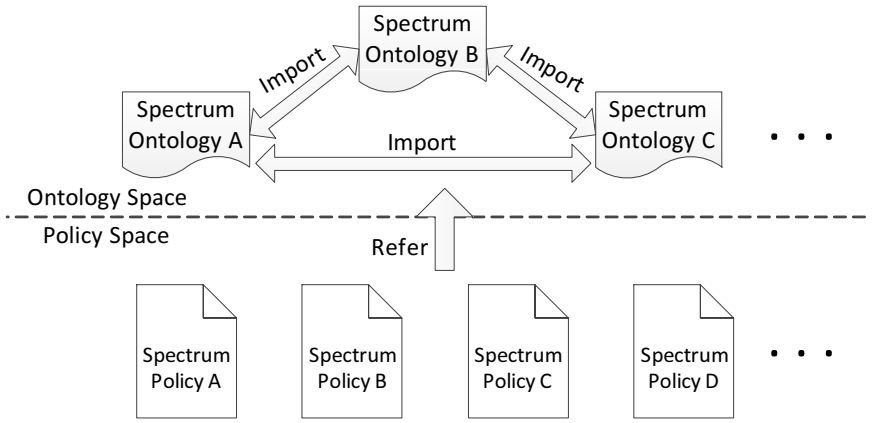


Figure 3.4: Interaction between spectrum ontology and policy documents.

3.4.2 Ontology-based Policy Reasoner

We employed *Pellet* (version 2.1.1) as the core ontology reasoning engine. Pellet is an open source OWL DL reasoner, and is written in Java. Pellet provides sound-and-complete

[2]A horn rule is an implication from an antecedent (a set of atomic formulae) to a consequent (a single atomic formula). All the variables in the consequent must occur in at least one atom of the antecedent, and are all considered to be universally quantified. [14]

OWL DL reasoning and supports reasoning with ontology individuals, ontology queries, and OWL/Rule hybrid reasoning [25]. One of the most important features of Pellet is that it adopts a number of optimization techniques for ontology reasoning to minimize the runtime overhead of ontology reasoning. Such a feature makes Pellet an attractive choice for use in a cognitive radio since a cognitive radio has strict real-time processing requirements. We modified parts of the Pellet's source code to enable the policy reasoner to compute spectrum opportunity constraints.

3.4.3 Waveform Generation and System Integration

We employed GNU Radio and USRP 1 boards to generate the waveforms and implement the radio frequency transceiver. We used two USRP 1 boards, both connected to host computers via USB 2.0 ports, one as the transmitter and the other one as the receiver. Separate Python programs are used to control the transmission and reception of waveforms. Because our experiments focused on the evaluation of the ontology-based policy reasoner, we did not implement a full-blown SSR, and instead functionalities of the SSR were emulated by a Python program that interacts with the policy reasoner. This SSR implementation generates randomly chosen transmission strategies and sends them to the policy reasoner. In our implementation, the ontology-based policy reasoner is written in Java whereas the rest of the software defined portions of the radio system is written in Python and C++. Hence, we had to bridge the Java run-time and Python run-time environments. We used JPype to implement the Java-Python interface. JPype is a tool that enables full access to Java libraries from Python programs. Access to Java libraries in Python programs was implemented by interfacing at the native level in both Java and Python virtual machines [28]. JPype can a start Java virtual machine in the Python run-time environment so that Java libraries can be invoked. In our radio system, the ontology-based policy reasoner is packaged into a Jar file and invoked from the SSR program. The communication between the SSR and the policy reasoner flows through JPype. Data type conversion from Python data type

to Java data type and vice versa is automatically handled by JPype.

3.5 Experimental Evaluation

Our radio system prototype executes in two phases: *preparation* phase and *transmission* phase. The preparation phase is executed before waveform transmission. These two phases are described below:

- **Preparation phase:** In the preparation phase, the radio carries out all of the initialization tasks of the radio system, which include loading the spectrum ontology information, executing initial ontology consistency checks, loading currently active spectrum policies, and preparing a Rete network using the spectrum policies. The tasks carried out in the preparation phase are not time critical because those tasks are completed before a transmission request is evaluated.

- **Transmission phase:** In the transmission phase, the radio evaluates a transmission request generated by the SSR and transmits the corresponding waveform if the transmission request is allowed by the policy reasoner. The runtime performance of this phase affects the performance of the upper layer applications. The tasks carried out in the transmission phase are time critical and need to satisfy the radio's real-time latency requirements (e.g., channel evacuation time).

In our experiments, we used the spectrum ontology and related spectrum policies created by SRI International [29]. The syntax of the SRI's ontology and policies were modified so that they can be understood and processed by our policy reasoner. SRI's spectrum ontology contains 40 classes, 75 properties and 24 individuals.

All the experiments were carried out on a PC with an Intel Core i5 2.67 GHz CPU, 4 GB memory, and running Ubuntu 10.04 operating system. It is obvious that the processing time

of our policy reasoner implementation is dependent on the computing platform's computing power. In all of the time measurements, we calculated the average of 20 independent runs.

3.5.1 Preparation Phase Evaluation

In this section, we describe the results from two experiments that were performed to evaluate the performance of the preparation phase. In the first experiment, we evaluated the impact of the spectrum ontology's size on the ontology loading and ontology consistency checking time. In the second experiment, we investigated the effect of the active spectrum policies' size on the Rete network construction time. The detailed descriptions of these two experiments and their results are described below.

3.5.1.1 Impact of Ontology Size on Loading and Consistency Checking Time

In this experiment, we only consider the impact of the number of individuals and the complexity of the individuals on the ontology loading and consistency checking time. We randomly generate individuals for a number of ontology classes and vary the total number of individuals in the spectrum ontology from 20 to 120. We also change the complexity of the individuals, which is quantified by the number of properties of an individual, in the spectrum ontology. The result of this experiment is shown in Figure 3.5. In this figure, ontology setup time denotes the time required to perform ontology loading and initial ontology consistency checking tasks.

From Figure 3.5, we observe that increasing the complexity of individuals (i.e., increasing number of properties per individual) has a noticeable impact on the growth rate of the ontology setup time. When the individuals are simple (i.e., have a small number of properties), increasing the number of individuals causes the setup time to increase at a moderate rate; when the individuals are more complex (i.e., have a larger number of properties), the setup time increases at a faster rate.

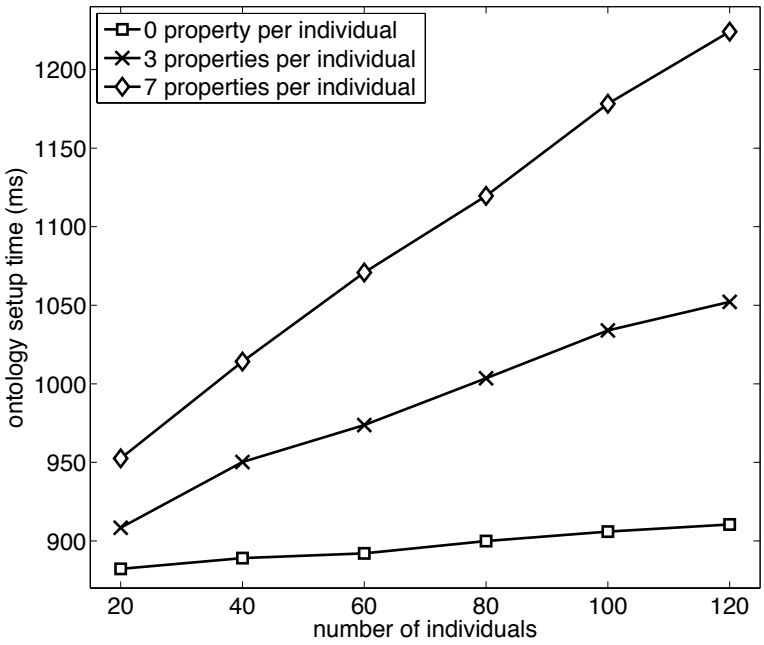


Figure 3.5: Ontology setup time vs. number of individuals.

3.5.1.2 Impact of Policy Size on Rete Network Construction Time

The Rete network is constructed from the active spectrum policies loaded into the radio system. *Active* policies are the specific policies that are considered when evaluating a transmission request. When measuring the Rete network construction time, we vary the number of loaded policies and their complexity. The metric for a policy's complexity is the number of conditions in it. The results of the experiment are given in Figure 3.6.

From Figure 3.6, we observe that the Rete network construction time is proportional to the number of loaded policies and their complexity.

3.5.2 Transmission Phase Evaluation

The most computationally expensive task performed in the transmission phase is the evaluation of the transmission request. Therefore, this task has a significant effect on whether a

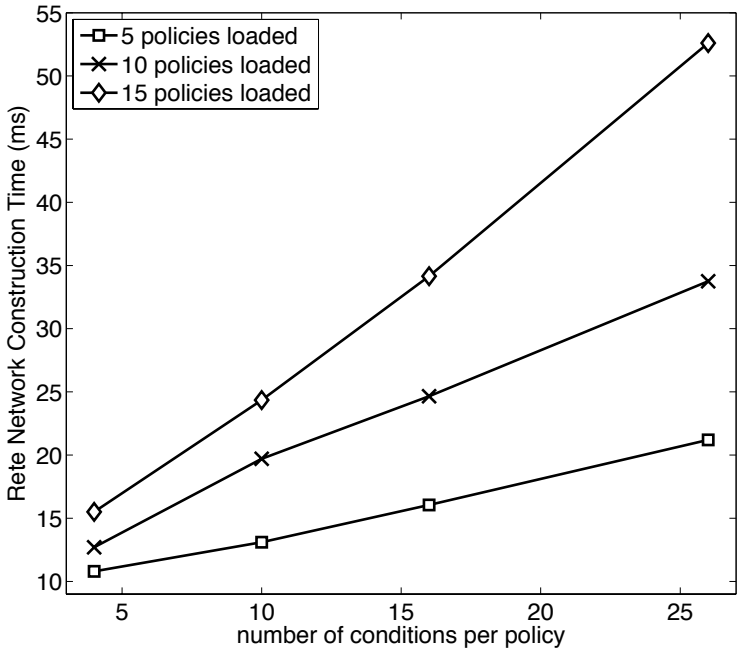


Figure 3.6: Rete network construction time.

radio can satisfy its critical timing requirements, such as the channel evacuation time. The channel evacuation time is the maximum amount of sojourn time in the current channel after the detection of a primary user signal and before hopping to a fallow channel. We performed a number of experiments to measure the runtime overhead of the transmission evaluation process.

3.5.2.1 Impact of Active Spectrum Policies

There are two policy attributes that affect the runtime overhead of a transmission request evaluation: number of active policies and their complexity. Note that the complexity of the policies affects the size of the Rete network. We can classify the conditions of spectrum policies into two categories:

- **Pattern Conditions:** Conditions that match the pattern of ontology individuals with

the spectrum policies;

- **Numeric Conditions:** Conditions that compare the values of the ontology individual properties with the values given in the spectrum policies.

In the first set of experiments, we vary the number of active spectrum policies from zero to 15. The zero policy case represents a scenario in which a transmission request is allowed without any policy-conformance checking, and this case serves as the baseline. We also vary the complexity of the spectrum policies. For each instance, the time required to evaluate the same single transmission request is measured. Figure 3.7 shows the results of these experiments.

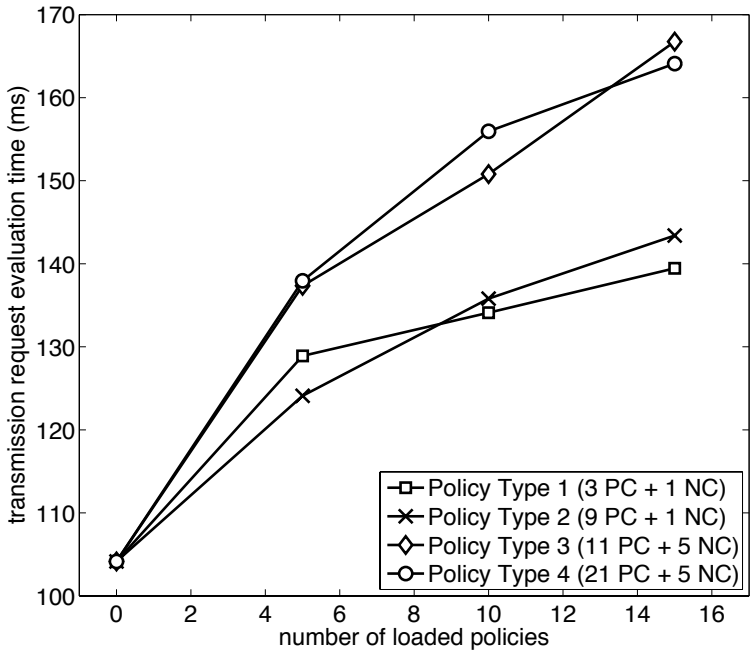


Figure 3.7: Impact of spectrum policies on transmission request evaluation performance.

As expected, the time required for transmission request evaluation increases proportionally to the increase in the number of loaded spectrum policies. From the figure, we can observe some interesting results related to policy types. By comparing the four curves in Figure 3.7,

we observe that the number of numerical conditions (in a policy) has a noticeable impact on the transmission request evaluation time, whereas the number of pattern conditions has a minor impact. This phenomenon can be attributed to the fact that the policy reasoner needs to perform data type conversion operations and data value comparison operations when processing numeric conditions, and these are computationally costly operations.

3.5.2.2 Impact of Spectrum Ontology Size

Our ontology and policy reasoner uses a Rete pattern matching algorithm to evaluate a transmission request against spectrum policy rules. In the reasoning process, the reasoner examines all of the individuals in the spectrum ontologies against the spectrum policy rules while ignoring the classes and properties of the ontologies. Therefore, we expect that the number of individuals in the ontologies to have a direct impact on the transmission request evaluation time, whereas the number of classes and properties to have no or minor impact. We ran two sets of experiments to study the effect of ontology size on the transmission request evaluation time.

In the first set of experiments, individuals in the spectrum ontologies do not match any of the rule patterns in the policies. The results are shown in Figure 3.8. From the figure, we can observe that the complexity of the ontology (i.e., number of properties per individual) has a minor impact on the transmission request evaluation time. This phenomenon can be explained by the fact that the Rete algorithm does not evaluate the individuals of the ontology because they do not match any of the policies' rule patterns. Hence, an increase/decrease in the number of properties per individual does not affect the runtime overhead of the transmission request evaluation.

In the second set of experiments, individuals in the spectrum ontologies do match some of the rule patterns in the policies. The results are shown in Figure 3.9. From the figure, we observe that the complexity of the ontology (i.e., number of properties per individual) does have a direct impact on the transmission request evaluation time. This phenomenon

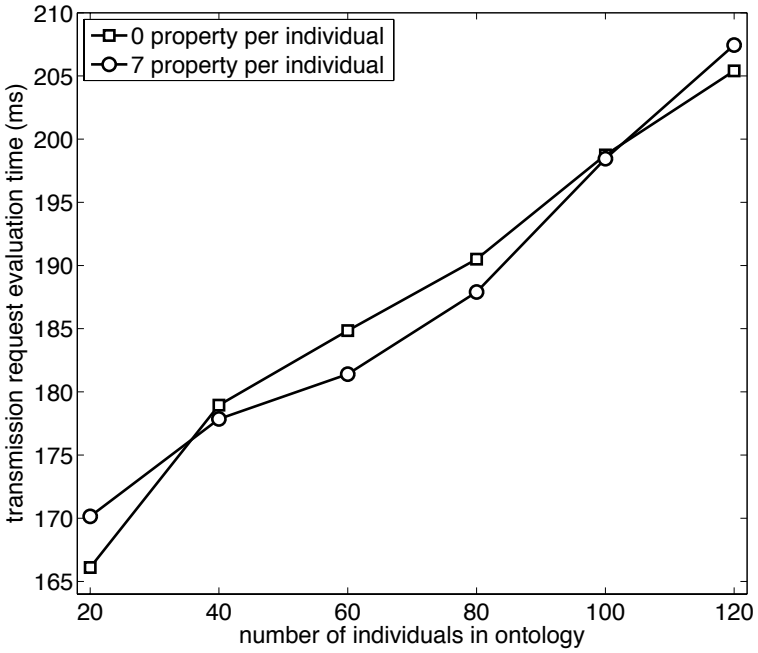


Figure 3.8: Impact of ontology size on transmission request evaluation time with no policy involvement.

can be explained by the fact that every individual in the spectrum ontology that matches the rule patterns of the spectrum policies is inserted into the Rete network, and the Rete algorithm checks whether this individual conforms to the spectrum policies. These operations performed by the policy reasoner cause additional runtime overhead, and this overhead increases proportionally with the increase in ontology complexity.

3.5.3 Guidelines for the Design of Ontology-based Policies

Through a number of experiments, we have investigated the impact of several factors—including the size and complexity of the spectrum ontology and the spectrum policies—on the runtime overhead of the preparation phase and the transmission phase. Here, we share a number of insights that we were able to gain through our experimental results. We provide these insights in the hopes that they may serve as useful guidelines on how to design spectrum

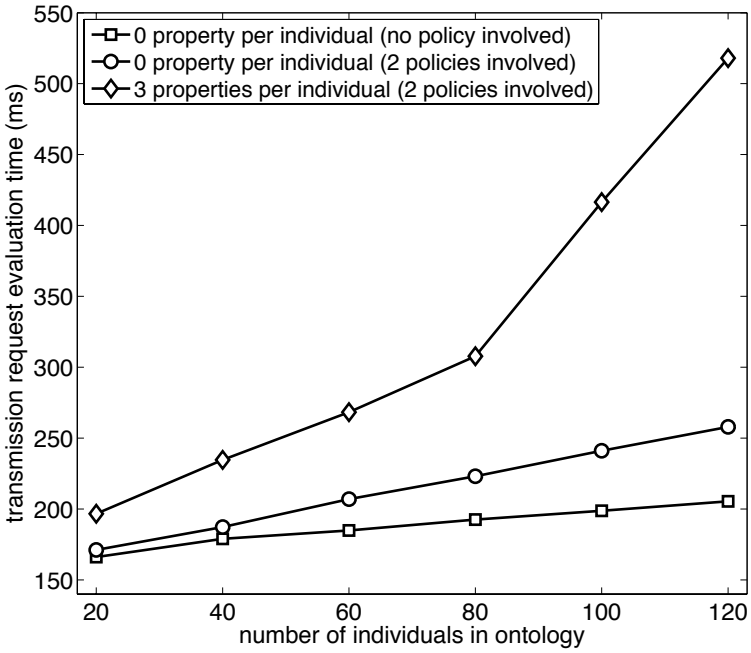


Figure 3.9: Impact of ontology size on transmission request evaluation time with policy involvement.

ontologies and policies for cognitive radio networks.

1. According to Figure 3.9, when spectrum access policies are involved, there is an inherent tradeoff between the complexity of spectrum ontologies and the runtime overhead of the ontology/policy reasoner. More complex ontologies enable spectrum regulators to author more descriptive and detailed policies but increase the processing burden of the ontology/policy reasoner.

2. The experiment results that are illustrated in Figure 3.7 imply that in order to minimize the transmission request evaluation time, spectrum policies need to be authored in such a way that their number of numeric conditions is minimized. These results show that the number of pattern conditions does not have a noticeable effect on the transmission request evaluation time.

3. The spectrum ontology should be designed in such a way that the number of pattern

matches between the ontology's individuals and the spectrum policies is minimized. According to Figure 3.8, this reduces the transmission request evaluation time significantly.

3.6 Summary

In this chapter, we discussed the design and implementation of a policy reasoner that processes ontology-based spectrum policies for cognitive radio networks. In ontology-based spectrum access policies, attributes of the spectrum policies are decoupled from the design of the policy controlled cognitive radio and the policy reasoner, which enables the policy reasoner to understand and process any spectrum policies authored by any spectrum regulators. Unfortunately, processing the spectrum ontologies and the ontology-based policies incurs additional computation overhead, which makes it more difficult to design a cognitive radio that meets the required real-time latency requirements. We have implemented a software-defined radio system, and carried out a set of experiments to evaluate the performance of the radio when it is controlled by ontology-based spectrum policies. Based on the insights gained from the experiments, we have also provided a set of guidelines for designing ontology-based spectrum policies.

Chapter 4

Security of Spectrum Learning in Cognitive Radio Systems

Considering the computing limitations and energy constraints of a battery-powered CR, a CR may not be able to perform full-spectrum sensing (i.e., sense all available spectrum bands) because of its prohibitive cost. Therefore, a spectrum sensing policy at the medium access control (MAC) layer is needed to decide which set of channels to sense. The *channel selection problem* is the problem of designing such a sensing policy. The optimal channel selection strategy for an unlicensed user (i.e., secondary user) is based on the availability statistics of the channels. The availability of the channels is determined by the presence/absence of primary user signals in those channels. The channels' availability statistics are initially unknown to a secondary user and need to be estimated using sensing samples. The critical tradeoff that the cognitive engine faces in each timeslot is between transmission ("exploitation") on the channel that has the highest expected reward (e.g., throughput) and channel sensing ("exploration") to get more information about the expected rewards of the other channels. The exploitation vs. exploration tradeoff problem, such as the one just described, is central to an area of machine learning known as *reinforcement learning*.

Spectrum learning is the process of learning the spectrum statistics (i.e., primary user occu-

pancy information), which is crucial to enable CRs to sense/interpret their spectrum environment and make intelligent decisions to achieve efficient communication. Although spectrum learning is beneficial for CRs, it can pose a serious security vulnerability. A radio that can learn has the potential to be taught by malicious entities in an adversarial environment. This kind of threat may have a long-lasting impact on the cognitive radio network.

As mentioned above, the design of the optimal sensing policy can be formulated as a *reinforcement learning* (RL) problem. When the channels are assumed to be independent, it can be formulated as a special class of RL problems known as a *restless multi-armed bandit* process. Recent results (i.e., [30] and [31]) show that a surprisingly simple *myopic policy* that ignores the impact of the current action on the future reward is optimal when channels are identical. In this chapter, we show that in adversarial environments, where an active attacker performs belief manipulation attacks against the machine learning algorithms executed on a cognitive engine, the myopic policy is no longer optimal and a *softmax policy* that exploits some level of randomness outperforms the myopic policy.

4.1 The Channel Selection System and the Attack Model

In this section, we introduce the channel selection system and establish the attack model.

4.1.1 Channel Selection System Model

We consider a general dynamic spectrum access system where a user has access to N independent and stochastically non-identical parallel Gillbert-Elliot channels [32], and chooses one channel to sense and access in each time slot, aiming to maximize its expected long-term reward (i.e., throughput).

As illustrated in Figure 4.1, the state of the k-th channel—either idle (1) or busy (0)—indicates that the channel is unused by primary users or it is occupied. The transitions

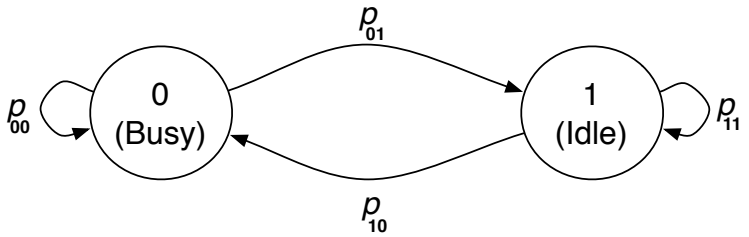


Figure 4.1: A Gillbert-Elliot channel.

between these two states follow a Markov chain with transition probabilities $\{p_{ij}^k\}_{i,j=0,1}$. We assume that these transition probabilities for all channels are learned in a non-adversarial environment before the system starts operating and thus the transition probabilities are known to the system. We also assume that $p_{11}^k > p_{01}^k$ or equivalently the channel states in two consecutive time slots are positively correlated. Note that this assumption is only used for derivation of closed-form expressions and can easily be relaxed by separately considering the case where this assumption does not hold. Due to its limited sensing and access capability, a secondary user chooses one of the N channels to sense and access in each slot. Designing an optimal sensing policy that governs the channel selection at each time slot can be formulated as a restless multi-armed bandit process for independent channels. We denote $S_k(t)$ as the state of channel k in slot t that is given by the two-state Markov chain in Figure 4.1. Let $S(t) = [S_1(t), \cdots, S_N(t)] \in \{0,1\}^N$ denote the full system state.

The channel selection system is reward-based. At each time slot the secondary user selects one of the N channels to sense. If the sensed channel is occupied by primary user signals, the user collects no reward; otherwise it accesses the channel and collects one unit of reward. The system keeps periodical sensing and transmitting on that channel until a primary user appears on the channel or a jamming attack prevents the user from transmission on the channel. The secondary user's aim is to maximize the *throughput* (reward) over a horizon of T slots by choosing an optimal sensing policy.

Due to limited sensing (i.e., sensing only one channel out of N channels), the full system state ($S(t)$) in slot t is not observable. However, it has been shown that a sufficient statistic

for optimal decision making is given by the conditional probability that each channel is in state 1, given all past observations and decisions [33]. Referred to as the belief vector, we denote this sufficient statistic by $\Omega(t) = [\omega_1, \cdots, \omega_N]$, where $\omega_k(t)$ is called the belief value of channel k which is equivalent to the conditional probability that $S_k(t) = 1$ given all past observations and decisions for that channel. Given the sensing action $a(t) = k$ (the channel that is selected to be sensed in slot t) and the observation $S_k(t)$ in slot t, the belief vector for slot $t+1$ can be updated via Bayes rule through the following equation:

$$\omega_k(t+1) = \begin{cases} p_{11}^k, & a(t) = k, S_k(t) = 1 \\ p_{01}^k, & a(t) = k, S_k(t) = 0 \\ \Gamma(\omega_k(t)), & a(t) \neq k \end{cases}, \tag{4.1}$$

where $\Gamma(x) = xp_{11}^k + (1-x)p_{01}^k$.

A sensing policy π specifies a sequence of functions $\pi = [\pi_1, \cdots, \pi_t]$, where π_t maps the belief vector $\Omega(t)$ to a sensing action $a(t)$. Multi-channel opportunistic access can thus be formulated as the following stochastic optimization problem:

$$\pi^* = \arg\max_{\pi} E_{\pi}[\sum_{t=1}^{T} R_{\pi_t(\Omega(t))}(t)|\Omega(1)], \tag{4.2}$$

where $\pi_t(\Omega(t))$ is the channel selected for sensing and $R_{\pi_t(\Omega(t))}(t)$ is the reward when the belief vector is $\Omega(t)$ and the action $\pi_t(\Omega(t))$ is taken, and $\Omega(1)$ is the initial belief vector. If no information about the initial system state is available, each entry of $\Omega(1)$ can be set to the stationary distribution ω_0^k of the underlying Markov chain:

$$\omega_0^k = \frac{p_{01}^k}{p_{01}^k + p_{10}^k}. \tag{4.3}$$

Let $V_t(\Omega(t))$ be the value function which represents the maximum expected total reward that can be obtained starting from slot t given the current belief vector $\Omega(t)$. Given that the user selects channel k and observes $S_k(t)$ in slot t, the maximum expected reward consists of the

following two parts:

1. The expected immediate reward:

$$E[R_k(t)] = E[S_k(t)] = \omega_k(t).$$

2. The maximum expected future reward:

$$V_{t+1}(\tau(\Omega(t)|k, S_k(t))),$$

where $\tau(\Omega(t)|k, S_k(t))$ denotes the updated belief vector for slot $t+1$ as given in (1). If we maximize over all channel selections, we obtain the following optimization equation:

$$V_t(\Omega(t)) = \max_{k=1,\cdots,N} \{\omega_k(t) + V_{t+1}(\tau(\Omega(t)|k, S_k(t)))\}.$$

Because the value function is limited to horizon T, we have: $V_T(\Omega(T)) = \max_{k=1,\cdots,N} \omega_k(T)$ and $V_t(\Omega(t)) = 0$ for $t > T$. Theoretically, the optimal policy π^* and its performance $V_1(\Omega(1))$ can be obtained by solving the above dynamic programming problem. However, because of the impact of the current action on the future reward and the uncountable space of the belief vector, obtaining the optimal solution via the above recursive equations is computationally prohibitive.

4.1.2 Attack Model

We assume that there exists a single attacker in the environment who tries to decrease the throughput of secondary users by preventing them from transmission in some time slots, by employing adaptive interference techniques (or cognitive jamming attacks). Although many defenses against these attacks have been proposed, but none of these defenses is perfect or can address all classes of attackers. While such defenses can restrict the instantaneous effect

of these attacks, they can not reduce the long-term effect of such attacks on cognitive radio network, when the network is using a learning system as part of its cognitive engine. These attacks gradually contaminate the knowledge base of a cognitive radio and lead the learning system to make wrong decisions which results in degrading the performance of the radio.

The attacker uses the same equipment as secondary users, i.e., it is equipped with a CR that is comparable to a typical secondary user's CR in terms of battery capacity, transmission power, computing power, memory capacity, etc. Therefore the attacker is power-limited and wish to avoid jamming continuously, which quickly drains power and causes fast detection by the attack detection module of cognitive engine. We also assume that the attacker can attack only one channel at each time slot. Suppose that the attacker perform attacks in t_a slots out of T consecutive time slots on average. We denote $\alpha = \frac{t_a}{T}$ as the *attack probability* in the environment. The attacker controls the attack probability in order to cause maximal damage to the network in terms of throughput reduction. It can easily be seen that a higher attack probability will result in a lower throughput for the cognitive radio system.

We assume that the adversary possesses full knowledge about the network and its parameters. We introduce the notion of the attacker's optimal strategy using the following definition:

Definition 3. *The attacker's* α**-optimal strategy** *is a strategy that minimizes the throughput of the target cognitive radio while keeping the attack probability fixed to value* α*.*

4.1.3 Attack Detection Model

The cognitive radio network employs a mechanism for monitoring network status and detecting potential malicious activity. This monitoring mechanism is proposed in [34] to protect the network against jamming attacks. The monitoring can be done by specific monitor nodes in a distributed network and a detection algorithm is employed by the detection module at a monitor node; it takes as input observation samples obtained by the monitor node (i.e., failed transmission/successful transmission) and decides whether there is an attack or not. On one

hand the observation window should be small enough, such that the attack is detected in a timely manner and appropriate countermeasures are initiated. On the other hand, this window should be sufficiently large such that the chance of a false alarm or a mis-detection is reduced.

The sequential nature of observations at consecutive time slots motivates the use of sequential detection techniques. A sequential decision rule is efficient if it can provide reliable decisions as fast as possible. There exists a trade-off between detection delay and detection accuracy in a detection scheme, i.e. a faster decision unavoidably leads to higher values of the probability of false alarm P_{FA} and probability of mis-detection P_M while lower values of these probabilities are attained at the expense of detection delay. For given values of P_{FA} and P_M, the detection test that minimizes the average number of required observations (and thus average delay) to reach a decision among all sequential and non-sequential tests is *Wald's Sequential Probability Ratio Test (SPRT)* [35].

In our case, the test is between hypotheses H_0 and H_1 with Bernoulli probability mass functions (p.m.fs) f_0 and f_1. Assume that Y is a random variable with the Bernoulli distribution, where $Y = 1$ denotes a failed transmission event in a slot. H_0 denotes the hypothesis that assumes the absence of an attack, and because the probability of a failed transmission in the absence of attack is p_{10} (i.e. $Pr\{Y = 1\} = p_{10}$), the corresponding p.m.f f_0 has a Bernoulli distribution with parameter $\theta_0 = p_{10}$. Similarly, because the probability of a failed transmission in the presence of an attack with attack probability α is $1 - p_{11}(1 - \alpha)$ (i.e. $Pr\{Y = 1\} = 1 - p_{11}(1 - \alpha)$), H_1 that denotes the hypothesis that assumes the existence of an attack has a Bernoulli p.m.f f_1 with parameter $\theta_1 = 1 - p_{11}(1 - \alpha)$.

The logarithm of likelihood ratio at stage k with observations $x_1, \cdots, x_k$ is:

$$S_k = \sum_{i=1}^{k} \ln \frac{f_1(x_i)}{f_0(x_i)}.$$

where $x_i = 1$, if the system observes a failed transmission at time slot i, and $x_i = 0$ otherwise.

The decision variable is defined as follows:

$$S_k \geq a \Rightarrow \text{ accept } H_1$$

$$S_k < b \Rightarrow \text{ accept } H_0$$

$$b \leq S_k < a \Rightarrow \text{ take another observation}$$

The analysis in [34] shows that using this SPRT, the average number of samples needed for detecting an attack is:

$$E[N|H_1] = \frac{C}{\theta_1 \ln(\frac{\theta_1}{\theta_0}) + (1-\theta_1)\ln(\frac{1-\theta_1}{1-\theta_0})},$$

where C is a fixed positive number. Using this equation for $E[N|H_1]$ we have:

$$\begin{aligned}\frac{\partial E[N|H_1]}{\partial \alpha} &= \frac{\partial E[N|H_1]}{\partial \theta_1} \times \frac{\partial \theta_1}{\partial \alpha} \\ &= \frac{-C(\ln(\frac{\theta_1}{\theta_0}) + \ln(\frac{1-\theta_0}{1-\theta_1}))}{(\theta_1 \ln(\frac{\theta_1}{\theta_0}) + (1-\theta_1)\ln(\frac{1-\theta_1}{1-\theta_0}))^2} \times p_{11}.\end{aligned}$$

Because $\theta_1 > \theta_0$, we have $\frac{\partial E[N|H_1]}{\partial \alpha} < 0$ and consequently $E[N|H_1]$ is a decreasing function of α, i.e., increasing the attack probability would decrease the average number of required observations for attack detection. This poses a fundamental trade-off problem to an attacker: Increasing the attack probability, α, increases the impact of the attack on the target (i.e., lower its throughput), but it also enables a detection module to detect the attack sooner. We define the attacker's cost as the inverse of $E[N|H_1]$:

$$\frac{\theta_1 \ln(\frac{\theta_1}{\theta_0}) + (1-\theta_1)\ln(\frac{1-\theta_1}{1-\theta_0})}{C},$$

Also in order to normalize the attacker's cost, we assume that the constant C is equal to the maximum value of the statement $\theta_1 \ln(\frac{\theta_1}{\theta_0}) + (1-\theta_1)\ln(\frac{1-\theta_1}{1-\theta_0})$ that happens at $\alpha = 1$, i.e.

$C = \ln(\frac{1}{p_{10}})$, therefore:

$$\text{Attacker Cost} = \frac{\theta_1 \ln(\frac{\theta_1}{\theta_0}) + (1-\theta_1)\ln(\frac{1-\theta_1}{1-\theta_0})}{\ln(\frac{1}{p_{10}})}. \tag{4.4}$$

This definition for the attacker's cost shows that by risking detection of the attack by a detection module, the attacker's cost increases. The equation 4.4 would be used as a measure for the *attacker's cost* in the rest of this chapter.

4.2 Sensing Policies Analysis in an Adversarial Environment with two channels

In this section, we analyze and compare the myopic policy [36] and the softmax policy [37] in a hostile environment for $N = 2$ identical channels, i.e. $p_{ij}^1 = p_{ij}^2 = p_{ij}$.

4.2.1 Analysis of the Myopic Policy

The myopic policy explained in this section is identical to the one presented in [36], but in this section, we analyze the performance of this policy in an adversarial environment. A myopic policy ignores the impact of the current action on the future reward and only focuses on maximizing the immediate reward. At any given time slot t, the myopic policy for selecting a channel for sensing can be expressed as follows:

$$\pi^m(t) = \arg \max_{i=1,\cdots,N} \omega_i(t).$$

In [38], the authors proved that for the channel selection system that was introduced in Section 4.1.1, the myopic policy is the optimal policy for all N. They also showed that the myopic policy has a simple structure that does not require the knowledge of the transition

probabilities p_{ij} or updates to the belief vector.

We define the steady-state throughput of the myopic policy as:

$$U^m = \lim_{T\to\infty} \frac{V^m_{1:T}(\Omega(1))}{T},$$

where $V^m_{1:T}(\Omega(1))$ is the expected total reward obtained in T slots under the myopic policy when the initial belief vector is $\Omega(1)$. Note that this definition only considers the presence or absence of a PU in a particular channel, but in practice due to competition of SUs for a channel, the throughput would be different from the above definition. The key to computing the throughput U is to first find how long a user stays in the same channel. Let us introduce the concept of a *transmission period* (TP), which represents the time that a user stays in the same channel. Let L_k denote the k-th TP. In [38], Zhao et al. showed that under the condition $p_{11} > p_{01}$, the steady-state throughput is:

$$U = 1 - \frac{1}{\overline{L}}, \tag{4.5}$$

where $\overline{L} = \lim_{K\to\infty} \frac{\sum_{k=1}^{K} L_k}{K}$ denotes the average length of a TP.

Throughput analysis is thus reduced to analyzing the average TP length $\overline{L}$. For $N = 2$, we can derive a closed-form expression of $\overline{L}$ as a function of the attack probability α, which leads to a closed-form expression of the myopic policy throughput $U^m(\alpha)$.

Theorem 1. *For $N = 2$, the average TP length of a myopic policy as a function of the attack probability, α, is given by:*

$$L^m(\alpha) = 1 + \frac{\overline{\omega}}{1 - p_{11}(1-\alpha)}, \tag{4.6}$$

where

$$\overline{\omega} = \frac{(1-\alpha)p^{(2)}_{01}}{(1-\alpha)p^{(2)}_{01} - A}, \tag{4.7}$$

and

$$A = \omega_0(1-\alpha)[1 - \frac{(p_{11}-p_{01})^3(1-p_{11}(1-\alpha))}{1-p_{11}(1-\alpha)(p_{11}-p_{01})}], \tag{4.8}$$

and

$$p_{01}^{(2)} = \frac{p_{01} - p_{01}(p_{11}-p_{01})^2}{p_{01}+p_{10}}.$$

Proof. From the structure of the myopic policy, $\{L_k\}_{k=1}^{\infty}$ forms a first-order Markov chain for N = 2. When the system is running in an adversarial environment with attack probability α, the transition probabilities of $\{L_k\}_{k=1}^{\infty}$ are given by

$$r_{ij} = \begin{cases} 1 - p_{01}^{(i+1)}(1-\alpha) & j = 1 \\ p_{01}^{(i+1)}(1-\alpha)^{j-1}p_{11}^{j-2}(1-p_{11}(1-\alpha)) & j \geq 2 \end{cases},$$

where $P_{01}^{(j)}$ is the j-step transition probability which is equal to $\omega_0 - \omega_0(p_{11}-p_{01})^j$. Let $\mathbf{R} = \{r_{ij}\}$ denote the transition matrix of $\{L_k\}_{k=1}^{\infty}$ and let $\mathbf{R}(:,k)$ denote the k-th column of $\mathbf{R}$. We have

$$\mathbf{1} - \mathbf{R}(:,1) = \frac{\mathbf{R}(:,2)}{1-p_{11}(1-\alpha)}, \tag{4.9}$$

and

$$\mathbf{R}(:,k) = \mathbf{R}(:,2)(p_{11}(1-\alpha))^{k-2} \text{ for } k \geq 2, \tag{4.10}$$

where $\mathbf{1}$ is the unit column vector $[1,1,\cdots]^T$. We denote $\Lambda = [\lambda_1, \lambda_2, \cdots]$ as the stationary distribution of $\{L_k\}_{k=1}^{\infty}$, i.e. $\Lambda\mathbf{R} = \Lambda$. Thus we have:

$$[\lambda_1, \lambda_2, \cdots]\mathbf{R}(:,k) = \lambda_k. \tag{4.11}$$

Combining (4.9), (4.10) and (4.11) results in

$$\lambda_1 = 1 - \frac{\lambda_2}{1-p_{11(1-\alpha)}},\ \lambda_k = \lambda_2(p_{11}(1-\alpha))^{k-2}. \tag{4.12}$$

Substituting (4.12) into (4.11) and solving for λ_2, we get $\lambda_2 = \overline{\omega}(1-p_{11}(1-\alpha))$, where $\overline{\omega}$ is

given in (4.7). From (4.12), we can find the stationary distribution as

$$\lambda_k = \begin{cases} 1-\overline{\omega}, & k=1 \\ \overline{\omega}(1-p_{11}(1-\alpha))(p_{11}(1-\alpha))^{k-2} & k>1 \end{cases}. \tag{4.13}$$

Using (4.13) to compute $L^m(\alpha) = \sum_{k=1}^{\infty} k\lambda_k$ results in (4.6). □

Using the results of Theorem 1, we can show that $U^m(\alpha)$ is a decreasing function of α, and thus an attacker can lower the target radio's throughput (which is employing myopic sensing) by increasing the attack probability α. However, as we discussed in Section 4.1.3, increasing α increases the probability that the attack is detected.

4.2.2 Analysis of the Softmax Policy

The *softmax* action selection policies are randomized policies where, at time t, the action a_t is chosen at random by the user according to some probability distribution giving more weight to actions which have performed well in the past. The greedy action is given the highest selection probability, but all the others are ranked and weighted according to their accumulated rewards [37]. The most common softmax action selection method uses a Gibbs or Boltzman distribution for the action selection probabilities. It chooses action a at time slot t, with probability

$$p_a(t) = \frac{e^{\omega_a(t)/\tau}}{\sum_{i=1}^{N} e^{\omega_i(t)/\tau}},$$

where τ is a positive parameter called the *temperature* and controls the greediness of the policy. High temperatures cause all the actions to be all equiprobable while low temperatures cause a high probability for greedy action, and in the limit as $\tau \to 0$, the softmax policy becomes equivalent to the myopic policy.

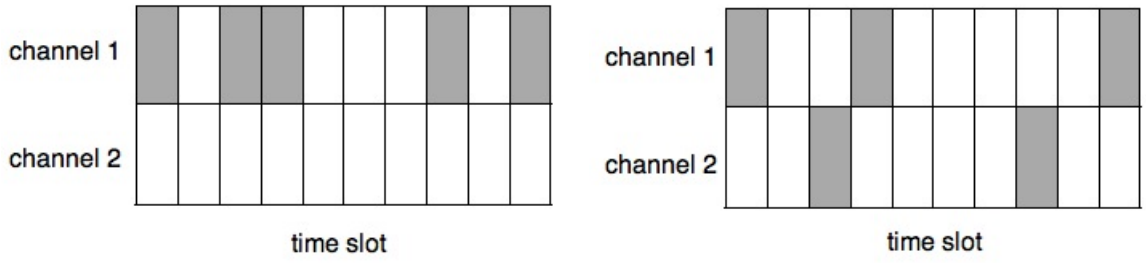
In this section, for simplicity and ease of computation, we use a Bernoulli distribution instead

of a Boltzmann distribution, i.e., we choose action a at time slot t with probability

$$p_a(t) = \begin{cases} q & \text{if} \quad a = \arg\max\limits_{i=1,2} \omega_i(t) \\ 1-q & \text{if} \quad a = \arg\min\limits_{i=1,2} \omega_i(t) \end{cases} . \tag{4.14}$$

We define the *main probability* q as the probability of taking greedy action (i.e. selecting the channel that has the highest ω_i). According to the definition of q and the softmax policy, we have $0.5 \leq q \leq 1$ when $N = 2$. Also note that for $q = 1$, the softmax policy reduces to myopic policy, i.e., myopic policy is a special case of softmax policy.

The attacker's optimal strategy for attacking a channel selection system employing the myopic policy is simple: only attack the channel that has the biggest belief value, since the user does not transmit on other channels. The attacker's optimal strategy against the softmax policy is not so straightforward.



(a) attacker's strategy with division probability d = 1

(b) attacker's strategy with division probability d = 0.6

Figure 4.2: attack strategy examples for fixed attack probability $\alpha = 0.5$

As mentioned in Section 4.1.2, the α-optimal strategy for an attacker is a strategy that minimizes the throughput of a cognitive radio while keeping the attack probability α fixed. Knowing that the softmax policy uses a fixed main probability q for channel selection (i.e., it uses (4.14) for channel selection), the attacker divides its attacks between the two channels. We define d as the conditional probability of channel 1 (the greedy option for the policy) being attacked at a given timeslot, assuming that the timeslot is attacked and call it the

division probability. Figure 4.2.2 illustrates the concept of the division probability (the slots colored in gray are the time slots in which an attacker jams a channel). The attacker chooses d such that a cognitive radio's throughput is minimized. On the other hand the channel selection system exploits its knowledge about the optimal strategy of the attacker to select the main probability q such that the throughput is maximized. Assume that $U^s(q, d)$ define the throughput of the radio when the softmax policy uses a main probability of q and the attacker exploits the division probability d, constructing optimal attack strategy and its corresponding optimal channel selection probability distribution can be formulated as the following optimization problems:

Optimization Problem 1:

$$d^* = \min_d U^s(q, d)$$
$$\text{s.t. } 0 \leq d \leq 1$$

Optimization Problem 2:

$$q^* = \max_q U^s(q, d^*)$$
$$\text{s.t. } 0.5 \leq q \leq 1$$

The steady-state throughput of the softmax policy is given by:

$$U^s = \lim_{T \to \infty} \frac{V^s_{1:T}(\Omega(1))}{T},$$

where $V^s_{1:T}(\Omega(1))$ is the expected total reward obtained in T slots under the softmax policy when the initial belief vector is $\Omega(1)$. As we mentioned earlier, we only need the average TP length to compute the throughput for the softmax policy. Suppose that the attacker uses its optimal strategy. An analysis similar to what we did in Section 4.2.1, results in the following theorem.

Theorem 2. *For $N = 2$, the average TP average length for a softmax policy with main probability q is given by:*

$$L^s(q, d) = qL^m(\alpha d) + (1 - q)L^n(\alpha(1 - d)),$$

where $L^m(\cdot)$ is the function defined in (4.6) *and $L^n(x) = 1 + \frac{p_{01}(1-x)}{1-p_{11}(1-x)}$.*

Proof. Using a procedure similar to the one used in the proof of Theorem 1, we can readily show that if the channel selection algorithm always selects the channel with the smaller belief value (which is the opposite of a greedy action), then the stationary distribution of $\{L_k\}_{k=1}^{\infty}$ is

$$\lambda_k = \begin{cases} 1 - p_{01}(1 - \alpha), & k = 1 \\ p_{01}(1 - \alpha)(1 - p_{11}(1 - \alpha))(p_{11}(1 - \alpha))^{k-2} & k > 1 \end{cases} \tag{4.15}$$

Using (4.15) to compute $L^n(\alpha) = \sum_{k=1}^{\infty} k\lambda_k$ results in

$$L^n(\alpha) = 1 + \frac{p_{01}(1 - \alpha)}{1 - p_{11}(1 - \alpha)}.$$

By using the Bayes' rule and the statement of Theorem 1, we can obtain the statement of Theorem 2. □

To quantify how much randomness is added to the channel selection system by the softmax policy, we use the entropy of the channel selection probability distribution, $\mathcal{H}$. For the $N = 2$ case, because we use Bernoulli distribution, $\mathcal{H} = -(q \ln(q) + (1 - q) \ln(1 - q))$. By changing q from 1 to 0.5, the entropy increases from 0 to its maximum value $\ln(2)$.

We denote $U^{\pi}(\alpha)$ as the throughput of the cognitive radio, when it uses policy π for channel selection and the attacker uses its α-optimal strategy. We use the following definitions to quantify the robustness and the performance of policy π:

Definition 4. *The* **robustness** *of a policy π for a channel selection system under an α-optimal attack is: $R^{\pi}(\alpha) = 1 - \frac{U^{\pi}(0)-U^{\pi}(\alpha)}{U^{\pi}(0)}$.*

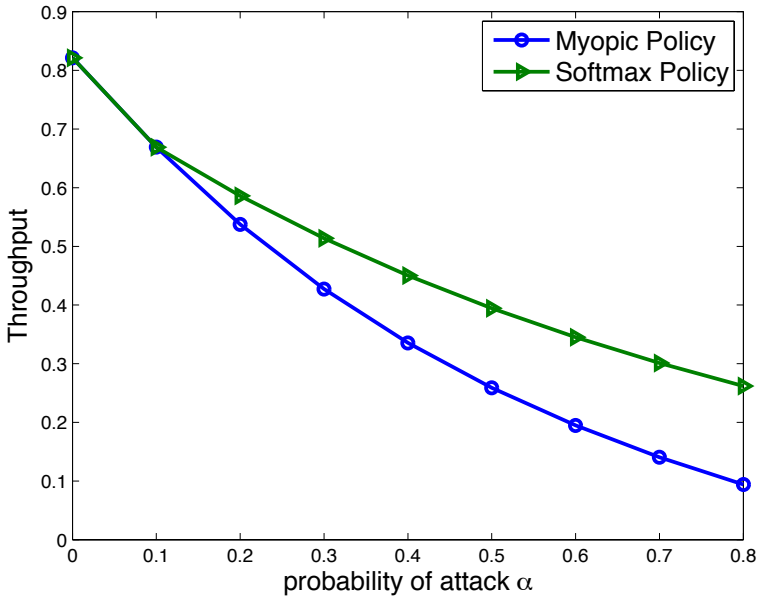


Figure 4.3: Throughput vs. attack probability for $N = 2$.

Definition 5. *The* **performance** *of a policy π for a channel selection system under an α-optimal attack is:* $P^{\pi} = U^{\pi}(0)$.

4.2.3 Numerical Results for Two Channels

Using the results of Theorem 2, it can easily be shown that for $\alpha = 0$, $q^* = 1$, i.e., a non-adversarial environment the optimal softmax policy is equivalent to the myopic policy. Solving the optimization problems 1 and 2 for other values of α is straightforward due to the problem's small solution space. Figure 4.3 shows the solutions to this problem for a range of attack probabilities. Because myopic sensing is a special case of softmax sensing, it is obvious that $U^s \geq U^m$. Numerical results illustrated in Figure 4.3 shows that except for small values of α ($\alpha < 0.1$), U^s is strictly greater than U^m, i.e., softmax sensing outperforms myopic sensing.

Figure 4.4 shows the trade-off between the robustness and the performance of the system for fixed attack probability $\alpha = 0.5$. As it can be seen, increasing the randomness that is used

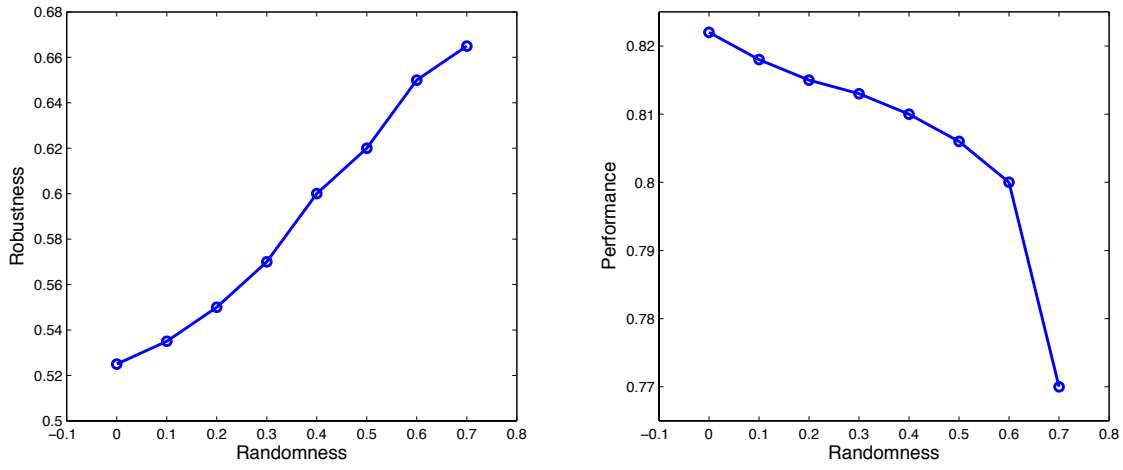


Figure 4.4: Performance and robustness vs. randomness for $N = 2$.

in the system by the softmax policy, increases the policy's robustness but it would decrease the policy's performance when no attacker exists.

4.3 Sensing Policies with More than Two Channels in an Adversarial Environment

As mentioned in [36], because L_k is a random process with higher-order memory, obtaining a closed-form expression of throughput for $N > 2$ channels with different statistics is very difficult. Nevertheless we can show that a softmax policy that uses a Boltzmann distribution with an intelligent choice for the temperature, τ, can outperform myopic policy for all possible strategies of the attacker, including its optimal strategy.

Assume that the channel selection system selects one of the N channels at each time slot. When a channel is selected, the system keeps using that channel until a primary user begins to transmit on the channel. Suppose $\Omega(t) = (\omega_1(t), \omega_2(t), \cdots, \omega_N(t))$ is the belief vector of the system at time t and $Q(t) = (q_1(t), q_2(t), \cdots, q_N(t))$ is the corresponding probability vector, i.e. $q_i(t) = f_i(\Omega(t))$ is the probability of selecting channel i at time t where $f(\cdot)$ is

a function that maps the belief vector into a probability value. For the sake of notation simplicity, we omit t in the belief and and probability vectors henceforth. Without loss of generality, assume that $\omega_N \geq \omega_{N-1} \geq \cdots \geq \omega_1$ and consequently $q_N \geq q_{N-1} \geq \cdots \geq q_1$. As we mentioned in Section 2, the α-optimal strategy for the attacker is a strategy that minimizes the throughput while keeping the attack probability α fixed. In order to do so, the attacker needs to attack channel i with probability αd_i, where d_i is the division probability for channel i (i.e., the conditional probability of channel i being attacked at a given timeslot, assuming that the timeslot is attacked) which is a function of Q and Ω. Also to keep α fixed, we need to have $\sum_{i=1}^{N} d_i = 1$. The following theorem states the optimization problem that the attacker needs to solve to find its optimal strategy.

Theorem 3. *In order to find its α-optimal strategy, the attacker needs to solve the following equality-constrained convex optimization problem:*

Optimization Problem 3.

$$\min_{d_1,\cdots,d_N} \sum_{i=1}^{N} q_i L(\omega_i, \alpha d_i)$$

$$s.t. \sum_{i=1}^{N} d_i = 1, \forall i : d_i \geq 0$$

where

$$L(\omega_i, \alpha) = 1 + \frac{\omega_i(1-\alpha)}{1 - p_{11}^i(1-\alpha)}. \tag{4.16}$$

Proof. In order to minimize the throughput, attacker needs to minimize the expected value for $L_k(\omega)$. We know that the attacker attacks this channel with probability αd_i, so we have:

$$Pr\{L_k(\omega_i) = l\} = \begin{cases} 1 - \omega_i(1-\alpha d_i), & l = 1 \\ \omega_i(1-\alpha d_i)(p_{11}^i(1-\alpha d_i))^{l-2} p_{10}^i, & l > 1 \end{cases}$$

It can easily be shown that

$$E[L_k(\omega)|\text{channel } i] = 1 + \frac{\omega_i(1-\alpha d_i)}{1 - p_{11}^i(1-\alpha d_i)}.$$

So we have:

$$E[L_k(\omega)] = E[E[L_k(\omega)|\text{channel } i]]$$
$$= \sum_{i=1}^{N} q_i(1 + \frac{\omega_i(1-\alpha d_i)}{1 - p_{11}^i(1-\alpha d_i)}).$$

To prove that the $E[L_k(\omega)]$ is a convex function of d_i's, we compute the Hessian matrix of this function: $H_{N\times N} = [h_{ij}]$. We can see that the Hessian of this function is a diagonal matrix where:

$$h_{ii} = \frac{2q_i\omega_i\alpha^2 p_{11}^i}{(1 - p_{11}^i(1-\alpha d_i))^3}.$$

It is obvious that we have $\forall i : h_{ii} \geq 0$, thus the matrix H is positive semidefinite and as a result $E[L_k(\omega)]$ is a convex function. □

The above equality-constrained convex optimization problem can be solved by using elimination and Newton's method in $\frac{(N+1)^3}{3}$ steps [39]. However, solving this optimization problem requires attacker's exact knowledge about the channel selection strategy of the system and the statistics of all channels during a long period of time. Obviously, acquiring such knowledge is not feasible under most circumstances. Therefore instead of using the α-optimal strategy, the attacker can use alternative, simpler strategies. Each of these strategies differ in terms of the amount of knowledge on the channel selection system that is required. These strategies are:

- Greedy Strategy: The attacker only knows the best channel for transmission at each time slot and attacks this channel, i.e., $d_N = 1$ and $d_i = 0$ for $1 \leq i \leq N-1$. This strategy is the optimal attack strategy when the channel selection system uses a myopic policy.

- Uniform Strategy: The adversary attacks all channels equiprobably, i.e., $d_i = \frac{1}{N}$: $1 \leq i \leq N$. This strategy can be used when the attacker does not have any knowledge about the channel selection system.

- Ω Strategy: The attacker only knows the channel statistics $\Omega = (\omega_1, \cdots, \omega_N)$ and has no knowledge about the channel selection policy of the system. It has a Boltzmann distribution with an arbitrary temperature τ_a, i.e., $d_i = \frac{e^{\omega_i/\tau_a}}{\sum_{j=1}^{N} e^{\omega_j/\tau_a}} : 1 \leq i \leq N$. Simulation results show that when the attack probability, α, is large, this strategy inflicts approximately the same effect as the α-optimal strategy.

In order to find the best channel selection strategy, the system assumes the worst-case attack scenario when the attacker uses its α-optimal strategy, which is the solution to the Optimization Problem 3: $d_i^* = f_i^*(Q, \Omega)$. The channel selection system needs to solve the following optimization problem in order to maximize the cognitive radio's throughput.

Optimization Problem 4.

$$\max_{q_1, \cdots, q_N} \sum_{i=1}^{N} q_i L(\omega_i, \alpha f_i^*(Q, \Omega))$$

$$\text{s.t.} \sum_{i=1}^{N} q_i = 1, \forall i : q_i \geq 0$$

Finding the global optimal solution of the non-linear optimization problem 4, gives us the amount of randomness that we need to add to the system in order to minimize the attack's effect and maximize the throughput.

We can also show that *for all possible attack strategies, including the α-optimal strategy, a softmax policy that uses a Boltzmann distribution with a well-chosen temperature outperforms myopic policy.*

Theorem 4. *When the channel selection system is consisted of more than two identical channels, for all attack strategies that the adversary may employ, with a fixed attack probability $\alpha > \frac{(\omega_0 - p_{01})N}{(\omega_0 N - p_{01})}$, a softmax policy that uses a Boltzmann distribution with temperature*

$$\tau > \frac{\omega_0 - p_{01}}{\ln \frac{p_{01}(N-\alpha)}{\omega_0 N(1-\alpha)}}, \tag{4.17}$$

achieves a greater throughput than a myopic policy.

Proof. Let $\omega_N \geq \omega_{N-1} \geq \cdots \geq \omega_1$ denote the belief values of all channels in the first slot of the k-th TP. For the myopic policy, the length L_k of this TP has the following distribution.

$$Pr\{L_k(\omega_N) = l\} = \begin{cases} 1 - \omega_N(1-\alpha), & l = 1 \\ \omega_N(1-\alpha)(p_{11}(1-\alpha))^{l-2}p_{10}, & l > 1 \end{cases}$$

And for the softmax policy, the length L_k of this TP has the following distribution.

$$Pr\{L_k(\overline{\omega}) = l\} = \begin{cases} 1 - \overline{\omega}, & l = 1 \\ \overline{\omega}(p_{11}(1-\alpha\overline{d}))^{l-2}p_{10}, & l > 1 \end{cases},$$

where $\overline{\omega} = \sum_{i=1}^{N} q_i\omega_i(1-\alpha d_i)$ and $\overline{d} = \sum_{i=1}^{N} q_i d_i$ in which $q_i = \frac{e^{\omega_i/\tau}}{\sum_{j=1}^{N} e^{\omega_j/\tau}} = \frac{1}{\sum_{j=1}^{N} e^{(\omega_j-\omega_i)/\tau}}$ are the action selection probabilities that follow a Boltzmann distribution. It is readily observable that if $\overline{\omega} \geq \omega_N(1-\alpha)$, then $L_k(\overline{\omega})$ stochastically dominates $L_k(\omega_N)$ and consequently the throughput of the softmax policy would be greater than the throughput of the myopic policy. We use the fact that $\forall i$, $p_{01} \leq \omega_i \leq \omega_0$ which results in

$$\frac{1}{Ne^{(\omega_0-p_{01})/\tau}} \leq q_i \leq \frac{1}{Ne^{(p_{01}-\omega_0)/\tau}}, \tag{4.18}$$

and also the fact that $q_N > \frac{1}{N}$ and $d_N > \frac{1}{N}$.

Now we show that $\overline{\omega} \geq \omega_N(1-\alpha)$. Using (4.18) and (4.17), we have:

$$\begin{aligned} \overline{\omega} &= \omega_N q_N(1-\alpha d_N) + \sum_{i=1}^{N-1} q_i\omega_i(1-\alpha d_i) \\ &\geq \omega_N q_N(1-\alpha d_N) + \sum_{i=1}^{N-1} p_{01}q_i(1-\alpha d_i) \\ &\geq \omega_N q_N(1-\alpha d_N) + \sum_{i=1}^{N-1} p_{01}\frac{1}{Ne^{(\omega_0-p_{01})/\tau}}(1-\alpha d_i) \end{aligned}$$

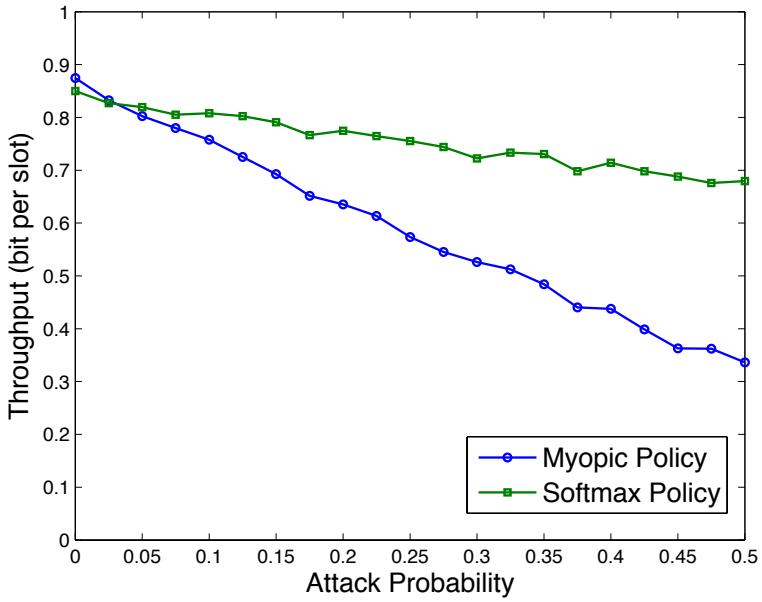


Figure 4.5: Throughput vs. attack probability for $N = 4$.

$$
\begin{aligned}
&\geq \omega_N q_N(1-\alpha d_N) + \frac{p_{01}}{N}\frac{\omega_0 N(1-\alpha)}{p_{01}(N-\alpha)}(N-1-\alpha(1-d_N)) \\
&\geq \omega_N(q_N(1-\alpha d_N) + \frac{(1-\alpha)}{(N-\alpha)}(N-1-\alpha(1-d_N))) \\
&\geq \omega_N(\frac{1}{N}(1-\alpha) + \frac{(1-\alpha)}{(N-\alpha)}(N-1-\alpha(1-\frac{1}{N}))) \\
&= \omega_N(1-\alpha)
\end{aligned}
$$

□

4.3.1 Numerical Results for More than Two Channels

We simulated the channel selection system to evaluate the performance of myopic sensing and softmax sensing for more than two channels. Figures 4.5 and 4.6 show the throughput

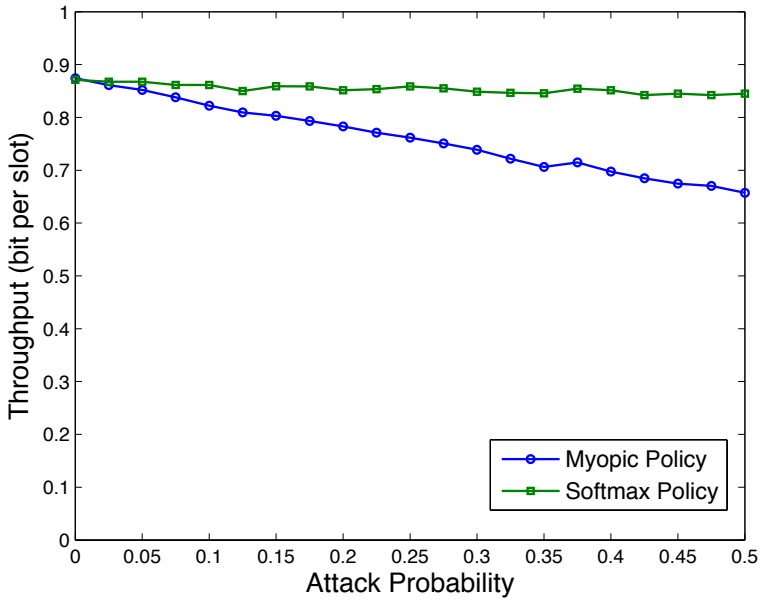


Figure 4.6: Throughput vs. attack probability for $N = 10$.

of the channel selection system versus the attack probability for myopic and softmax policies when 4 channels and 10 channels are available, respectively. To observe the effect of the number of channels on our scheme, we used identical channels with transition probabilities $p_{11} = 0.9$, $p_{10} = 0.1$, $p_{00} = 0.8$ and $p_{01} = 0.2$ to obtain the results in Figures 4.5 and 4.6. Softmax policies in these figures use a Boltzmann distribution with fixed temperature $\tau = 2$ for channel selection.

As it can be seen, the throughput drop caused by the increase in the attack probability is more severe for the myopic policy. Comparing the slope of lines in Figures 4.5 and 4.6, we can see that increasing the number of channels from 4 to 10 makes the softmax policy more robust against the belief manipulation attacks. In Figure 4.6, the softmax policy's drop in throughput is barely noticeable even as the attack probability is increased to 0.5.

Figures 4.5 and 4.6 also show that in a non-adversarial environment (i.e. when $\alpha = 0$), the performance of myopic policy is better than the performance of a softmax policy. These figures confirm the findings of [36] in which the authors showed that the optimal policy in

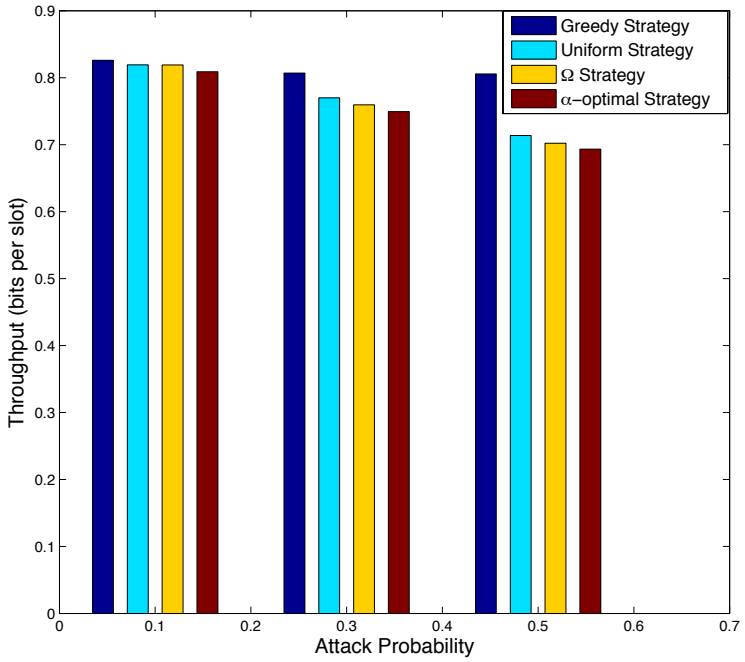


Figure 4.7: Comparing different attack strategies.

Table 4.1: Transition Probabilities

	p_{11}	p_{10}	p_{00}	p_{01}
Channel 1	0.9	0.1	0.8	0.2
Channel 2	0.95	0.05	0.8	0.2
Channel 3	0.9	0.1	0.85	0.15
Channel 4	0.95	0.05	0.85	0.15

non-adversarial environments is the myopic policy.

To compare different attack strategies and to investigate the amount of required randomness in the channel selection system, we used the more realistic model of non-identical channels with different transition probabilities. Figure 4.7 shows the effectiveness of the different attack strategies for different attack probabilities in a system of 4 non-identical channels with transition probabilities shown in Table 4.1.

As can be seen in Figure 4.7, the Ω strategy's performance is close to that of the α-optimal strategy for large values of α. We can also see from the figure that the greedy strategy is

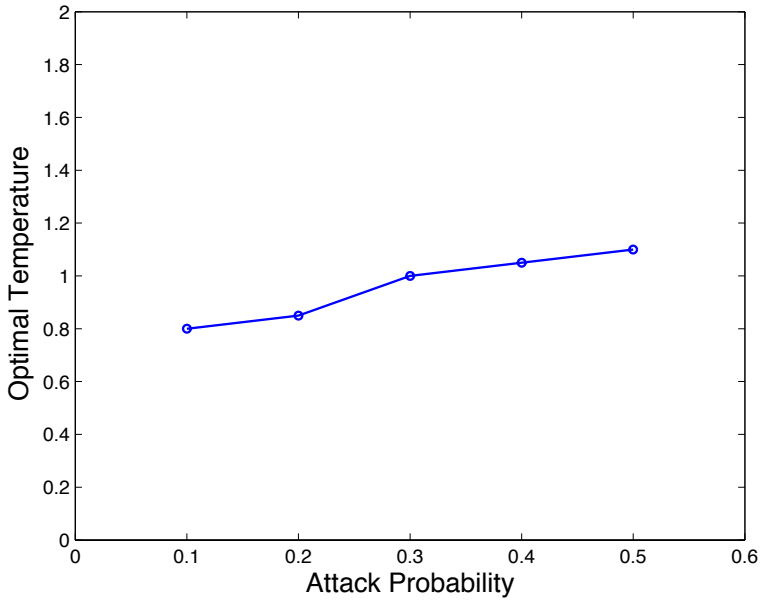


Figure 4.8: Optimal amount of randomness vs. attack probability

the worst strategy among these four different strategies.

It is well known that the temperature of a Boltzmann distribution, τ, is a measure of the amount of randomness in the distribution, which can be inferred from the fact that the entropy of the Boltzmann distribution is an increasing function of τ [40].

Figure 4.8 plots the optimal temperature value for the channel selection system when the attacker uses the α-optimal strategy. These temperature values were obtained by solving Optimization Problem 4. From the figure, we can observe that the channel selection system needs to increase the randomness (i.e., τ) in its learning process as the attack probability (i.e., α) is increased to minimize the effect of the attack.

We observed the effect of increasing attack probability on performance of the channel selection system. Figure 4.9 plots the attacker's cost using the equation 4.4 versus the attack probability. This figure shows that though increasing the attack probability reduces the performance of the channel selection system significantly, it also increases the attacker's cost with a high rate.

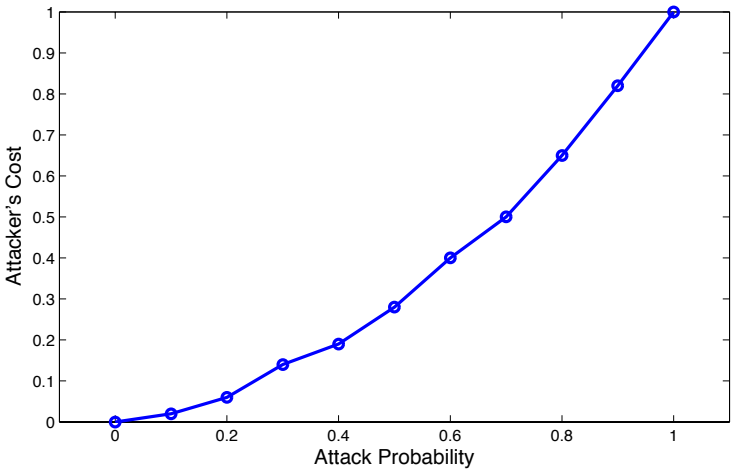


Figure 4.9: Attacker's cost vs. attack probability

4.4 Summary

In this chapter, we analyzed the security of a reinforcement learning algorithm that is used for solving the channel selection problem. We proposed a sensing policy that uses some level of randomness in the decision-making function to hide information about the learning algorithm. The obtained theoretical and simulation results show that the proposed mitigation technique can cause a dramatic improvement in the robustness of the channel selection process to adaptive jamming attacks. This countermeasure is applicable to other cognitive radio applications that use the same type of machine learning algorithm to reduce the effect of belief manipulation attacks.

Chapter 5

Location Privacy of Primary Users in Database-Driven Spectrum Sharing

When PUs and SUs share spectrum, the SUs must adopt technologies that enable them to avoid causing any interference to PUs. The FCC ruling on TV white spaces proposes relying on a database of the incumbents' spectrum usage information as the primary means of determining white space availability at any white space device [7]. The database is required to house an up-to-date repository of incumbents including television stations, and in certain cases, wireless microphones, and use this information to determine white spaces availability at a WSD's location. It has been shown that sensing-only devices do not generally utilize spectrum as efficiently as geolocation enabled devices, due to the large margins in incumbent detection thresholds that must be built into sensing-only devices [41]. Geolocation enabled devices have knowledge of the specific interference protection requirements of each licensed incumbent, which allows varying levels of protection to be applied, and thus maximizing utilization of the spectrum.

Although using geolocation databases for spectrum sharing has many advantages, it poses a potentially serious privacy problem. For instance, SUs, through seemingly innocuous queries to the database, can determine the *types*, *locations*, and *times of operation* of incumbent

systems operating in a given region of interest—we refer to this as the *operational privacy* of the incumbents. When the incumbent systems are commercial systems, such as the case in TV spectrum, this is not an issue. However, when the incumbents are federal government systems, including military systems, then the information revealed by the databases can result in serious breach of privacy.

The prospects of spectrum sharing between federal government systems and non-government systems has been heightened by a recent *notice of proposed rulemaking (NPRM)* published by the FCC. In December 2012, the FCC published an NPRM to create a *Citizens Broadband Service* in the 3.5 GHz band that will promote two major technical advances that enable more efficient use of spectrum: small cells and spectrum sharing [42]. As part of the NPRM, the FCC is looking at whether it will be feasible to open up approximately 100 megahertz of spectrum in the 3550-3650 MHz bands for small cell technologies on an unlicensed basis. The 3.5 GHz band is currently used by the U.S. Department of Defense (DoD) for certain radar installations as well as by non-federal users. It is highly likely that spectrum sharing in the 3.5 GHz band will be enabled by geolocation databases aided by local spectrum sensing. In the 3.5 GHz band, the criticality of the incumbents' operational privacy is obvious. The operational privacy of the incumbents is one of the major hurdles holding back the federal government from opening up some of its spectrum to spectrum sharing with commercial systems. Research programs recently launched by federal funding agencies, including the NSF and DARPA, have stressed the importance of security and privacy in spectrum sharing [43], [44].

The problem of operational privacy of PUs cannot be addressed by tightly controlling access to the database, since all SUs need access to it to enable spectrum sharing. A more viable approach is to "obfuscate" the information revealed by the database in an intelligent manner such that a certain level of privacy is assured while supporting efficient use of the spectrum. In this chapter, we focus on the problem of ensuring the *location privacy* of the incumbents. Privacy-preserving techniques as well as metrics for evaluating them often need to be tailored to a given specific problem. We propose four new privacy-preserving techniques that address

the specific problem of location privacy of incumbent systems.

5.1 Technical Background

As mentioned previously, PUs' location privacy is a critical concern when federal government users (i.e., PUs) share spectrum with commercial users (SUs). The policies, technologies, and infrastructure needed to support such a spectrum sharing scenario are at a very early conceptual stage, and there is almost no existing work on the topic. Therefore, in this chapter, we assume that some of the technologies and infrastructure for enabling spectrum sharing (between federal government users and non-government users) are similar to those that have been adopted for TV white space spectrum when such an assumption is appropriate.

5.1.1 An Overview of the Database Access Protocol

Here, we briefly review a database access protocol that is typical of protocols used in database-driven spectrum sharing. A database-driven cognitive radio network consists of three fundamental components: primary users, secondary users, and geolocation database (GDB). GDB is an entity which contains spectrum availability information for a given location. PUs have priority access to the spectrum, and SUs have secondary access under the condition that they incur little or no interference to the PUs. SUs opportunistically use fallow parts of the spectrum (i.e., whitespace). A SU can be an access point, a base station, or a portable device. SUs are required to query the database to obtain spectrum availability information. Before starting transmission, a SU must send a query to the database to request a list of available whitespace channels based upon its geolocation. The query should include parameters such as the SU's location, accuracy of that location, antenna characteristic information (e.g., directivity, height, etc.), and the device identifier. If the SU meets all regulatory domain requirements, then the database responds with the following information: a set of available whitespace channels, duration of the allowed use of those channels, and

associated maximum power level or a notification of any additional requirement for sensing. The regulatory domain requirements may include entity authentication of the SU querying the GDB and submission of valid location information by the SU.

5.1.2 Protected Contour of a Primary User

The exclusion area for a PU, where no SU is allowed to transmit with maximum power, is called the PU's *protected contour* and is stored in the database. Protected contours are defined by spectrum regulatory agencies such as the FCC. For example, for TV whitespace, FCC has specified the use of $F(50, 90)$ propagation curves when computing contour levels for digital TV stations. $F(X, Y)$ curves represent the actual field strength that would exceed a certain threshold at X% of the locations for Y% of the time. To prevent interference to TV reception within these protected contours, further separation distances are added to the contours according to the SU's antenna height—e.g., when the antenna height is less than 3 m, the distance requirement is 4 km for co-channel and 400 m for adjacent channels [7]. For instance, using a $F(50, 90)$ curve, it can be shown that a WRAN station that is operating at its maximum power would typically have to protect 132 km around a transmitting DTV station [45].

In its third memorandum opinion and order (MO & O) for TV bands [7], the FCC eliminated the requirement that TV band devices that incorporate GDB access must also listen (sense) to detect the signals of TV stations and low-power auxiliary service stations (i.e., wireless microphones). According to these regulations, SUs may not operate co-channel to a registered low power auxiliary station within a distance of 1 km of the registered wireless microphones.

In this chapter, we assume that the maximum transmission power (MTP) that is assigned to a SU by the database is calculated as $P = h(d)$, where P is the maximum transmission power, d is the distance between the SU and a PU receiver, and $h(\cdot)$ is a deterministic function called the MTP function which is dictated by a regulatory agency. The function can be specified

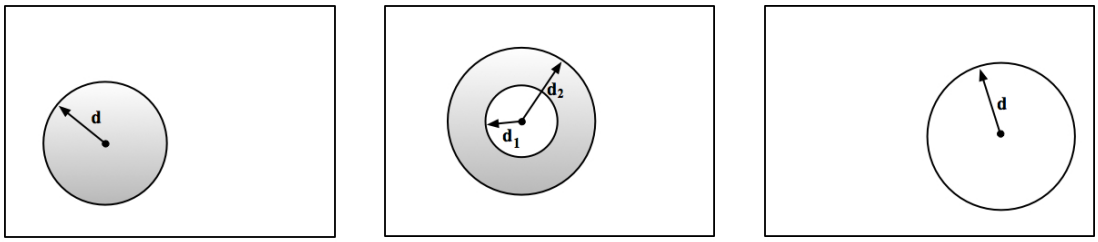
using $F(X, Y)$ propagation curves or a computationally simple but relatively accurate two-ray propagation model, like the one proposed in [41]. The details of this function are out of the scope of this dissertation.

5.2 Inferring the Location of Stationary Incumbents

In this section, we show that a group of SUs can infer the location of *stationary* PUs in a database-driven cognitive radio network through a set of seemingly innocuous queries.

5.2.1 Problem Formulation: Interaction between the Database and SUs

We assume that the region that is serviced by a GDB is divided into $m \times n$ square cells, $c(i, j)$, where $1 \leq i \leq m$ is the row index and $1 \leq j \leq n$ is the column index. We assume that there are A primary users whose movements are confined to their cells (i.e., each PU can move in his/her cell, but cannot go from one cell to another one) and B secondary users that can be either stationary or mobile within the region serviced by the database. Primary and secondary users are represented as PU_i and SU_j, respectively, where $1 \leq i \leq A$ and $1 \leq j \leq B$. Let P^{max} define the maximum possible transmission power for a SU. The PUs operate on C channels and these channels can be used by SUs under the condition that they do not cause interference for PUs. Before starting transmission, a secondary user, SU_i, sends a query, $Q = (ID_i, loc_i, A_i)$, to the database, where ID_i is the unique identifier of SU_i, $loc_i = (x_i, y_i)$ are its location coordinates, and A_i denotes antenna attribute information. The unique identifier ID_i of SU_i is used by the database to authenticate the user and keep the track of its queries. If the SU meets all regulatory domain requirements, the database checks all the C channels to find out the channels that can be used by the SU and sends a response, $R = ((ch_1, P_1, t_1), (ch_2, P_2, t_2), \cdots, (ch_k, P_k, t_k))$, where ch_i is the i-th available



(a) Case 1: Channel is not available.

(b) Case 2: Channel is available but transmission power is limited.

(c) Case 3: Channel is available and maximum possible transmission power is allowed.

Figure 5.1: Three possible database responses to secondary users' queries.

channel, P_i is the maximum allowed transmission power ($0 < P_i \leq P^{max}$), and t_i is the time interval that represents the duration of the channel's availability.

5.2.2 Threat Model

We assume that the attacker is a single mobile SU that can move throughout the region serviced by the database and send queries to the database. Alternatively, we can assume that the attacker represents a group of colluding stationary (non-mobile) SUs that are located throughout the region. In terms of our threat model, these two cases are equivalent. The attacker's goal is to infer the location of the PUs by analyzing the database's responses to its queries. We also assume that there is a non-negligible time gap between two queries of a SU such that the database has enough time to update its knowledge about the user before responding to another query from the same user.

5.2.3 An Algorithm for Inferring the Location of Stationary PUs

Here, we present an algorithm, Algorithm 6, that an attacker can employ to infer the location of a stationary PU. Unlike other localization techniques that use information such as signal strength, angle of arrival, time difference of arrival, and so on, the algorithm merely uses a series of database query replies—which by themselves do *not* directly reveal the PU's

location—to infer the location within certain accuracy. This algorithm demonstrates the feasibility of location inference attacks against geolocation databases. The following paragraphs describe the details of the algorithm.

Assume that $X_{ij}^{(k)}$ is a Bernoulli random variable that is equal to 1 if a PU exists in cell $c(i,j)$ that operates on channel k, and is 0 otherwise. Let $p[X_{ij}^{(k)} = 1] = p_{ij}^{(k)}$ and $p[X_{ij}^{(k)} = 0] = 1 - p_{ij}^{(k)}$. The attacker uses the value, $p_{ij}^{(k)}$, to infer the location of the PUs. We assume that the attacker has no prior information about the presence of PUs in all the cells; in other words, $p_{ij}^{(k)} = \frac{1}{2}$ for all values of i, j and k. After each response of the database to the attacker's query, the attacker updates the value of $p_{ij}^{(k)}$ for a group of cells in the grid. If the value of $p_{ij}^{(k)}$ exceeds a threshold, δ, the attacker infers that there is a PU in cell $c(i,j)$ that operates on channel k. To update the values of $p_{ij}^{(k)}$, the attacker uses the procedure explained in the following paragraphs.

After querying the GDB and obtaining the corresponding responses, the attacker analyzes all of the C channels to extract information about the location of PUs; here, C denotes the total number of available channels. For each channel ch_k, where $1 \leq k \leq C$, there are three different cases to be considered by the attacker:

- Case 1: channel unavailable. Channel ch_k is not in the list of available channels that are returned by the database, i.e. $P_k = 0$. Figure 5.1(a) illustrates this case. The unavailability of channel ch_k means that there is a PU within distance d from the SU's location that operates on this channel. The value of d is determined by the MTP function, i.e., the SU finds the value of d that results in $h(d) = 0$. The grey circle in figure 5.1(a) indicates the region in which a PU may exist. A cell inside this circle, where the probability of at least one PU's presence is nonzero, is called a *p-cell*. Note that the actual shape of the protected contour of a PU may not necessarily be a perfect circle. However, to simplify the computations, we approximate this region using a circle which encloses all of the protected contour. Also if only a part of a cell fall into this circle, we still consider it as a p-cell. In this case, the attacker updates

$p_{ij}^{(k)}$ values for each p-cell, $c(i,j)$, using the Bayes rule. Suppose that G denotes the number of p-cells, event H represents the hypothesis of the existence of a PU in a p-cell, $c(i,j)$, and event O denotes the attacker's observation (extracted from the database's response) that there is a PU in cell $c(i,j)$ or one of the other $G-1$ p-cells. If event H^c denotes the absence of a PU in $c(i,j)$, then the probability of observing the same p-cells, $(p(O|H^c))$, is $\frac{G-1}{G}$ (existence of a PU in one of the other $G-1$ p-cells). The Bayes rule indicates that:

$$\begin{aligned} p(H|O) &= \frac{p(O|H)p(H)}{p(O|H)p(H) + p(O|H^c)p(H^c)} \\ &= \frac{1 \times p_{ij}^{(k)}}{1 \times p_{ij}^{(k)} + \frac{G-1}{G} \times (1 - p_{ij}^{(k)})}. \end{aligned}$$

Therefore for all the p-cells, the attacker should update the $p_{ij}^{(k)}$ values using the following equation:

$$p_{ij}^{(k)} = \frac{p_{ij}^{(k)}}{1 - \frac{1}{G}(1 - p_{ij}^{(k)})}, \tag{5.1}$$

where G is the number of p-cells. Note that when G is a large value, $p_{ij}^{(k)}$ does not change significantly, but for small values of G, $p_{ij}^{(k)}$ approaches 1 rapidly. In the extreme case of when $G = 1$, $p_{ij}^{(k)} = 1$.

- Case 2: channel available but transmission power is limited. Using channel ch_k is allowed under a limited transmission power constraint, i.e. $P_k < P^{max}$. This reply indicates that there is no PU within distance d_1 from the SU, and there exists at least one primary at a distance between d_1 and d_2 from the SU. A cell that is empty of PUs is called an *e-cell*. Distances d_1 and d_2 are computed using the MTP function. Figure 5.1(b) illustrates this case. The attacker sets $p_{ij}^{(k)} = 0$ for e-cells, and uses Equation 5.1 to update the value of $p_{ij}^{(k)}$ for the p-cells.

- Case 3: channel available. Channel ch_k is available and maximum possible transmission power is allowed, i.e. $P_k = P^{max}$. This reply indicates that there is no PU within

distance d from the SU. However, this reply reveals no information about the possible presence of PUs beyond distance d. Figure 5.1(c) illustrates this case. The attacker sets $p_{ij}^{(k)} = 0$ for the e-cells and does not change the $p_{ij}^{(k)}$ values for other cells.

Based on the above three cases, an attacker can formulate Algorithm 6.

5.2.4 Evaluation of the Algorithm for Inferring the Location of Stationary PUs

In Section 5.2.3, we proposed an algorithm that enables an adversary to infer the location of a stationary PU using a series of queries to the geolocation database. To evaluate the algorithm, we applied the algorithm to the Spectrum Bridge TV whitespace database [46]. Spectrum Bridge's TV White Space database platform was one of the first to be certified by the FCC. Although the location privacy of PUs in TV white space is not important, we use the Spectrum Bridge's database in our evaluations because it is one of few geolocation databases that are available to the public. When a SU sends a query to this database for a given location, the database returns a list of all available channels for that location in addition to the allowed antenna HAAT (Height Above Average Terrain) for stationary devices and maximum allowed transmission power for portable devices.

In our evaluation of Algorithm 6, the aim is to find the location of a TV station transmitter using responses obtained from the Spectrum Bridge database; queries are sent from different locations. We used a 333 km by 360 km rectangular region with latitude between $N36.00°$ and $N39.00°$, and longitude between $W78.00°$ and $W82.00°$ as the starting search space for the PUs. We divided this region into a 300 by 400 grid of cells, where the length and width of each cell is 0.01 degrees in latitude and longitude, respectively. This region covers parts of three states: Virginia, West Virginia, and North Carolina. We selected channel 13 (210-216 MHz), and queried the database from 30 different locations throughout this region. The database's responses to the availability of channel 13 were used as inputs to Algorithm 6.

Algorithm 6 Stationary PU Location Inference Algorithm

Input: Sequence of queries $Q = \{q_1, q_2, \cdots\}$ and their corresponding responses $R = \{r_1, r_2, \cdots\}$.

Output: Location of PUs in the grid.

Initialize $p_{ij}^{(k)}$ values:
for all cells $c(i,j)$ and channels k **do**
    $p_{ij}^{(k)} = \frac{1}{2}$.
end for
while an inference has not occurred **do**
    Send query $q_i = (ID_i, loc_i, A_i)$ to the database.
    Receive the list of available channels $r_i = \{(ch_k, P_k, t_k)\}$.
    for each channel k **do**
        if k is not in R **then**
            Compute distance d using the MTP function.
            Use d and location of the SU to find p-cells.
            Update $p_{ij}^{(k)}$ values for the p-cells using equation 5.1.
            if $p_{ij}^{(k)} > \delta$ **then**
                an inference has occurred.
                return $c(i,j)$ and k
            end if
        else if k is in R with a limited power **then**
            Compute distances d_1 and d_2 using the MTP function.
            Use d_1, d_2 and location of the SU, to find p-cells and e-cells.
            Update $p_{ij}^{(k)}$ values for the p-cells using equation 5.1.
            if $p_{ij}^{(k)} > \delta$ **then**
                an inference has occurred.
                return $c(i,j)$ and k
            end if
            Put $p_{ij}^{(k)} = 0$ for the e-cells.
        else if k is in R with maximum possible power **then**
            Compute distance d using the MTP function.
            Use d and location of the SU to find e-cells.
            Put $p_{ij}^{(k)} = 0$ for the e-cells.
        end if
    end for
end while

Algorithm 6's outputs consist of candidate cells that have the greatest possibility of containing the PU of interest. These cells cover the region with latitude between $N37.30°$ and $N37.33°$, and longitude between $W79.61°$ and $W79.65°$. By looking at the map of tv transmitter locations, we observe that the WSET-TV transmitter, which transmits on VHF channel 13, is located in one of this region. In this example, the maximum distance from the TV transmitter (i.e., PU of interest) to any point in the region is 2.75 km. In other words, the algorithm was able to infer the location of a PU, which is located inside a 333 km by 360 km region, to within an accuracy of 2.75 km.

5.3 Inferring the Location of Mobile Incumbents

We propose an algorithm that an attacker can employ to infer the path of movement of a mobile PU even when the query replies from the database do not directly reveal any location-related information. The algorithm does not make any specific assumptions about the PU's mobility model. This algorithm uses recursive Bayesian estimation (a.k.a. Bayes filter) to track the mobile PUs using the information extracted from the GDB's query replies.

5.3.1 Recursive Bayesian Estimation

Recursive Bayesian estimation is a probabilistic technique for estimating an unknown probability density function recursively over time using sensory observations and a mathematical process model [47].

Let x be an unobserved Markov process that defines the true state (that we need to estimate), and z represent the observed states of a Hidden Markov Model (HMM). Figure 5.2 illustrates a Bayesian Network of this model.

Assume that Z_k represents the joint event of observations $z_1, z_2, \cdots, z_k$. Because of the Markov assumption, the probability distribution over all states of the HMM can be written

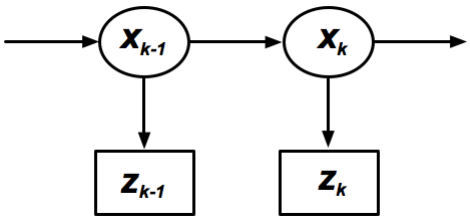


Figure 5.2: Bayesian network of a hidden Markov model.

as:

$$p(x_0, \cdots, x_k, Z_k) = p(x_0) \prod_{i=1}^{k} p(z_i|x_i) p(x_i|x_{i-1}) \tag{5.2}$$

Using 5.2 we can write the *predict* and *update* steps of the Bayes filter probabilistically. The probability distribution associated with the predicted state is

$$p(x_k|Z_{k-1}) = \int p(x_k|x_{k-1}) p(x_{k-1}|Z_{k-1}) dx_{k-1},$$

and the probability distribution of update is

$$p(x_k|Z_k) = \frac{p(z_k|x_k) p(x_k|Z_{k-1})}{p(z_k|Z_{k-1})}.$$

In order to use the recursive Bayesian estimation technique for our application, we need to devise algorithms for computing probabilities $p(z_k|x_k)$ and $p(x_k|Z_{k-1})$
based on the location of the SU. Note that for our application, x_k is a random variable that denotes a PU's path of movement at the time of the k-th query, and z_k denotes the maximum allowed transmission power for transmitting in the channels indicated in the database's response to the k-th query. In order to find the probabilities $p(z_k|x_k)$ and $p(x_k|Z_{k-1})$, we need to establish the relation between x_k and z_k as well as between x_k and x_{k-1}. The general model that is usually used for this relation is:

$$\begin{aligned} x_k &= f(x_{k-1}) + v_k \\ z_k &= h(x_k) + w_k, \end{aligned} \tag{5.3}$$

where $f(\cdot)$ and $h(\cdot)$ are deterministic functions, and v_k , and w_k are noise components.

The problem of inferring a PU's path of movement can be interpreted as a *target-tracking* problem. The motion information, x_k, consists of two components: loc_k and vel_k—the former denotes a point in a two dimensional Euclidean space that indicates the location of the PU, and the latter denotes a two dimensional velocity vector that indicates direction and speed of the motion. We assume that the PU is moving with a constant average speed. The value, z_k, is a function of the distance between loc_k and the current location of the SU (i.e., querier). To find z_k from loc_k, we use the MTP function introduced in Section 5.1.2.

To model the relation between x_k and x_{k-1}, the well-known discrete-time kinematic equation can be used [48]:

$$x_k = Fx_{k-1} + v_k, \tag{5.4}$$

where:

$$F = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix},$$

and $v_k = (v_k^1, v_k^2)^t$ is an additive white noise vector with zero mean ($E[v_k] = 0$) and the covariance matrix is:

$$\Sigma = E[v_k v_k^t] = \begin{bmatrix} \frac{1}{3}T^3 & \frac{1}{2}T^2 \\ \frac{1}{2}T^2 & T \end{bmatrix} \sigma,$$

where σ is the noise intensity and T is the time difference between two queries. Since the functions $f(\cdot)$ and $h(\cdot)$ are not linear, the appropriate Bayes filter to use for this problem is the *particle filter* [49].

Algorithm 7 Mobile PU Location Inference Algorithm

Input: Sequence of queries $Q = \{q_1, q_2, \cdots\}$ and their corresponding responses $R = \{r_1, r_2, \cdots\}$.

Output: Kinematic information of mobile PUs in the grid.

Initialization:
for all N_p particles i and C channels ch_s **do**
    $w_0^i(s) = \frac{1}{N_p}$
end for
$k = 1$. (number of queries)
while an inference has not occurred **do**
    Send the k-th query $q_k = (ID_k, loc_k, A_k)$ to the database.
    Receive the list of available channels $r_k = \{(ch_k, P_k, t_k)\}$.
    for all channels s **do**
        Observe $z_k = P_s$ of channel ch_s in r_k.
        T = time difference between queries k and $k-1$.
        Resampling:
        Select a set of N particles randomly: $\{x_{k-1}^i\}$.
        Prediction:
        for all N chosen particles i **do**
            Compute x_k^i using equation 5.4:
            $loc_k^i = loc_{k-1}^i + T \times vel_{k-1}^i + v_k^1$.
            $vel_k^i = vel_{k-1}^i + v_k^2$
            $x_k^i = (loc_k^i, vel_k^i)$
        end for
        Update:
        for all N chosen particles i **do**
            Use MTP function $h(\cdot)$ to compute $\tilde{w}_k^i$:
            $\tilde{w}_k^i = p(z_k|x_k^i) = \begin{cases} 0 & z_k \neq h(x_k^i) \\ 1 & z_k = h(x_k^i) \end{cases}$
        end for
        for all N chosen particles i **do**
            $w_k^i(s) = \frac{\tilde{w}_k^i}{\sum_{j=1}^N \tilde{w}_k^j}$
        end for
        $N_{eff} = \frac{1}{\sum_{i=1}^N (w_k^i(s))^2}$
        if $\lceil N_{eff} \rceil > N_{thr}$ **then**
            an inference has occurred.
        end if
    end for
    $k = k + 1$
end while

5.3.2 Particle Filters and the Location Inference Algorithm

Here, we present an algorithm, Algorithm 7, that can infer the location of a mobile PU using information extracted from a series of GDB query replies. The algorithm uses a particular type of Bayes filter—namely, a particle filter—to track the PU's path of movement. Specifically, the algorithm uses one particle filter for each channel, ch_i ($1 \leq i \leq C$). Each particle filter has N_p particles, and the weights of those particles are updated after each query response received from the GDB. The following paragraphs describe the particle filter used by Algorithm 7.

Suppose that a set of N random samples are taken from the posterior pdf, $p(x_{k-1}|Z_{k-1})$. We denote these samples or particles as $\hat{x}^i_{k-1}$, where $1 \leq i \leq N$. In the prediction phase of a particle filter, the particle filter generates a set of prior particles, x^i_k, by applying Equation 5.3 to posterior particles $\hat{x}^i_{k-1}$ using the following equation:

$$x^i_k = f_{k-1}(\hat{x}^i_{k-1}) + v^i_{k-1}, \tag{5.5}$$

where v^i_{k-1} denotes an independent sample drawn from the pdf of the system noise. This procedure produces a set of particles from the prior pdf $p(x_k|Z_{k-1})$.

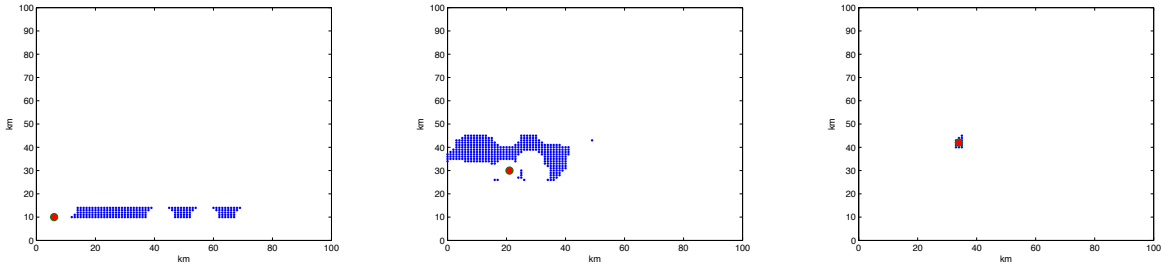
To update the prior samples using the new observed state z_k, a weight $\tilde{w}^i_k$ is calculated for each particle. This weight is the likelihood of the observed state, evaluated at the value of the prior sample:

$$\tilde{w}^i_k = p(z_k|x^i_k).$$

The weights are then normalized so they sum to unity:

$$w^i_k = \frac{\tilde{w}^i_k}{\sum_{j=1}^{N} \tilde{w}^j_k},$$

and the prior particles are resampled (with replacements) according to these normalized weights to produce a new set of particles $\hat{x}^i_k$ such that $p(\hat{x}^i_k = x^j_k) = w^j_k$ for all i and j.



(a) Particles after 15 queries. (b) Particles after 50 queries. (c) Particles after 75 queries.

Figure 5.3: Particles tracking a mobile primary user using Algorithm 7.

In other words, a member of the set of prior samples is chosen with a probability equal to its normalized weight, and this procedure is repeated N times to build up the new set of particles $\{\hat{x}_k^i\}$. The new particles are samples of the required pdf $p(x_k|Z_k)$ and so an iteration of the algorithm is complete.

To find the convergence point of the algorithm (convergence results in a successful inference of the PU's location), we use a measure called the effective sample size [50] defined as

$$N_{eff} = \frac{1}{\sum_{j=1}^{N}(w_k^j)^2}, \tag{5.6}$$

where $(1 \leq N_{eff} \leq N)$ for each channel. A value close to 1 indicates that almost all the probability mass is assigned to one particle and there is only one useful sample in the set. Conversely, if the weights are uniformly spread amongst the particles the effective sample size approaches N. It is often suggested that the resampling process should only be performed if N_{eff} falls below some threshold N_{thr} that is chosen empirically. At the end of every iteration, each particle filter compares the effective sample size given by Equation 5.6 with the threshold, N_{thr}, to determine the algorithm's convergence point.

5.3.3 Evaluation of the Algorithm for Inferring the Location of Mobile PUs

In this section, we evaluate the efficacy of Algorithm 7 in inferring the location of mobile PUs. The region of interest (starting search space) is a 100×100 grid of square cells, where the side length of each cell is 1 km. We use a random walk mobility model for modeling the mobility of the PU. At each iteration, the PU decides with equal probability to move to the east or to the north with a constant speed of 1 km/min. At each iteration of Algorithm 7, a SU that is randomly located inside the grid (an arbitrary cell) queries the database. The database replies to a query with the maximum allowed transmission power which is calculated as

$$P = h(d) = \begin{cases} 0 & \text{for} \quad d < 2 \\ 40 & \text{for} \quad 2 \leq d < 5 \\ 100 & \text{for} \quad d \geq 5 \end{cases} \quad \text{(mW)}, \tag{5.7}$$

where d is the distance between the SU and the closest mobile PU in the grid.

The algorithm uses 90000 particles, and each particle is denoted as (x, y, v_x, v_y), where $1 \leq x, y \leq 100$ and $-1 \leq v_x, v_y \leq 1$ and all values are integers. The algorithm converges quickly, and is able to infer the PU's location and velocity with a high accuracy. At each iteration, the algorithm is provided with an input that represents the database's response to a SU's query. It is assumed that the SU is located randomly in the region of interest. The input is used to update the particle weights.

Figure 5.3 illustrates the process of tracking a mobile PU using a particle filter described in 5.3.2. The blue dots denotes the particles with the highest weights, and the red circle denotes the location of the PU of interest. After 75 queries, there are only a few particles remaining that accurately estimate the location of the PU.

5.4 Techniques for Preserving Location Privacy of Primary Users

In this section we propose four privacy-preserving techniques for location privacy of primary users in a geolocation database. Note that since the source of information for both inference attacks against the location privacy of mobile and stationary PUs is the same, we can use the countermeasures described below for preserving location privacy of both types of PUs.

5.4.1 Perturbation with Additive Noise

The perturbative masking method (a.k.a randomization method) is a technique for privacy-preserving databases that uses data distortion in order to mask the attribute values of records. In this method, we add sufficiently large noise to individual record values to prevent recovery of these values by an adversary [51]. One key advantage of the randomization method is that it is relatively simple, and does not require knowledge of the distribution of other records in the data. Additive noise is the most basic perturbative method that can be used in privacy-preserving databases. In this technique, the vector of observations x_j for the j-th attribute of the original dataset X_j is replaced by a vector $z_j = x_j + \epsilon_j$, where ϵ_j is the random noise vector. We can use this technique to obfuscate the information that the attacker can infer from the database's reply to its queries.

Recall that the database's response to each SU query is a set of values indicating the maximum allowed transmission power for all the channels available in a given area, i.e., $\{(ch_k, P_k, t_k)\}$, where P_k denotes the maximum transmission power for channel ch_k and t_k represents the time interval that the channel is available. To preserve the privacy of PUs' location using perturbation with additive noise, the database can provide a *false positive* response to a query. This false positive response corresponds to adding a negative random noise value ϵ_k to P_k and replacing P_k with $P'_k = P_k + \epsilon_k$. Note that we must use a negative

noise to have $P'_k \leq P_k$ for each k, because replacing P_k with P'_k such that $P'_k > P_k$ (i.e. allowing the SU to transmit with a power higher than real maximum allowed transmission power) will likely result in interference to PUs. Also note that using a negative additive noise to transmission power values P_k is equivalent to increasing the radius of the protected contour of PUs.

Though the technique described above, that employs perturbation with additive noise, increases location privacy of PUs, it reduces the spectrum utilization efficiency of the secondary network. We address this fundamental tradeoff problem between the PUs location privacy and the SUs' spectrum utilization efficiency in Section 5.6.

5.4.2 Perturbation with Transfiguration

Another form of perturbation that might be used to preserve the location privacy of PUs is changing the shape of the protected contour of PUs. Replacing the circular protected contours with some random convex shapes that envelops the actual circular protected contour will increase the location privacy of PUs. This method obfuscates the location of PUs by preventing the attacker from using the symmetry of the circular shape of a PU's protected contour. Figure 5.4 illustrates an example where a circular protected contour is inscribed in a pentagon-shaped protected contour. The circular contour is the actual contour, and the pentagon-shaped contour is the transfigured contour. Note that since the new protected contour's area is larger than the original one, this method also reduces spectrum utilization efficiency of the secondary network because the SUs can use the spectrum in a smaller area.

Here we propose an algorithm for preserving location privacy of PUs using transfiguration of protected contour. This algorithm replaces the circular protected contour of a PU with an N-sided polygon with an irregular shape. The number N controls the level of privacy, i.e. a smaller value for N guarantees a higher level of privacy but reduces efficiency of spectrum utilization for the SUs. As $N \to \infty$, the irregular polygon approaches the original circular shape of the protected contour.

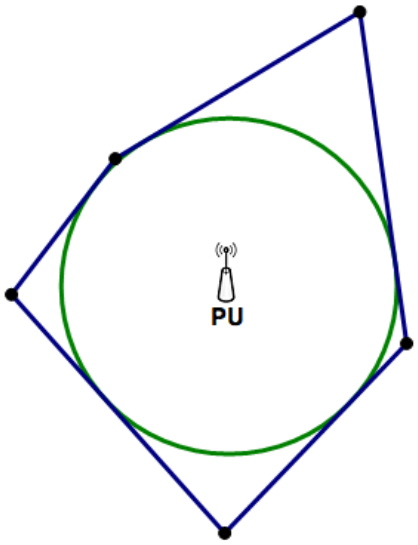


Figure 5.4: Perturbation using transfiguration.

Algorithm 8 Perturbation with Transfiguration Algorithm for Preserving Location Privacy

Input: Circular protected contour C_u of primary user u, number of polygon sides N.
Output: Coordinates of vertices of the N-sided polygon.

1: Divide C_u to N arcs of equal size, i.e. each arc is $\frac{360}{N}$ degrees.
2: Choose a random point q_i on each arc i.
3: **for** each i **do**
4: Compute l_i, the tangent line to C_u at point q_i.
5: **end for**
6: **for** each i **do**
7: Compute M_i, the point of intersection between l_i and l_{i+1}. $(l_{N+1} = l_1)$
8: **end for**
9: **return** the set of coordinates $\{M_i\}$.

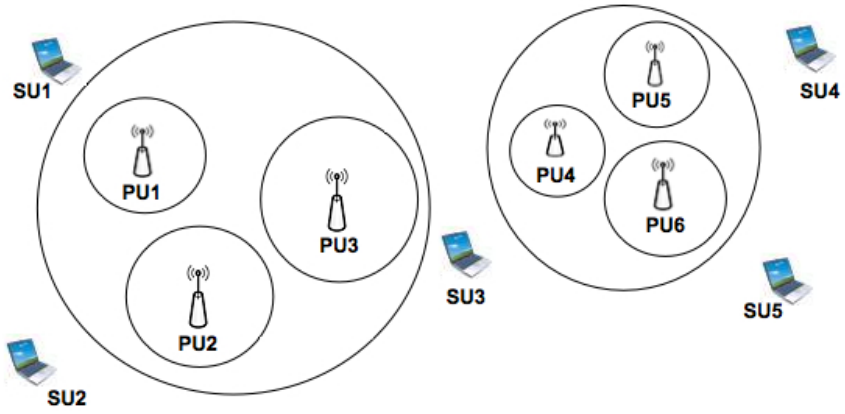


Figure 5.5: 3-anonymity for primary users' location privacy.

Algorithm 8 outputs coordinates for a set of N points that form a convex protected contour with irregular shape that envelops the original circular protected contour. The algorithm starts by dividing the circular protected contour into N arcs and then choosing one random point on each of these arcs. The tangent lines at these points form the irregular-shaped protected contour that will be replaced with the original circular shape. The way that we chose the random points ensures that the tangent lines will form a closed shape for the protected contour.

5.4.3 k-Anonymity

The concept of k-anonymity was originally introduced in the context of relational data privacy [52]. The motivating factor behind the k-anonymity model was the possibility of indirect identification of records from public databases via *quasi-identifiers* i.e., combination of multiple record attributes that can be used to identify individual records [53]. For instance, a medical institution may want to release a table of medical records with the names of the individuals replaced with dummy identifiers. However, some set of attributes (which are referred to as the *quasi-identifier*) can still lead to identity breaches. In the k-anonymity method, the granularity of data representation is reduced with the use of techniques such as

Table 5.1: Simplified geolocation database.

ID	2D Location	Radius of Protected Contour (km)
PU1	(1,2)	0.2
PU2	(3,-3)	0.25
PU3	(5,1)	0.3
PU4	(9,2)	0.1
PU5	(12,3)	0.15
PU6	(12,1)	0.2

generalization or suppression.

In the context of location privacy of PUs in database-driven spectrum sharing, we can achieve location k-anonymity by combining protected contours of k primary users that are closest together and creating a larger protected contour that works like the cloak box for Location-Based Services (LBSs) [54]. The secondary users are not allowed to transmit in the area covered by this larger protected contour. This increases the privacy of primary users but will decrease the performance of the secondary network. Figure 5.5 illustrates this idea for $k = 3$.

Consider the following simplified scenario. Suppose table 5.1 shows the location and the radius of the protected contour of the 6 PUs in figure 5.5. Also assume that all PUs are operating in the same band. To answer a SU's query, the database checks if the SU is located inside the protected contour of a PU or not.

To implement 3-anonymity, we replace Table 5.1 with Table 5.2, where the set of PUs {PU1,PU2,PU3} and {PU4,PU5,PU6} are replaced with $\overline{\text{PU1}}$ and $\overline{\text{PU2}}$, respectively. The location of the virtual PU that is replaced with 3 actual PUs is the circumcenter of the triangle that consists of the location of the 3 PUs (the circumcenter is the center of a circle that circumscribes the triangle formed by the three PUs). The radius of the protected contour is the distance between the circumcenter and one of the PUs plus the largest radius of protected contour of the three PUs.

Note that although 3-anonymity by spatial cloaking preserves the location privacy of PUs,

Table 5.2: Simplified geolocation database with 3-anonymity.

ID	2D Location	Radius of Protected Contour (km)
$\overline{\text{PU1}}$	(2.56,-0.28)	3.06
$\overline{\text{PU2}}$	(10.67,2)	1.86

it results in poor spectral efficiency for the secondary network, because the SUs cannot transmit in some regions that are not part of the protected contour of real PUs.

It is known that the optimal k-anonymity problem, even under simplifying assumptions, is NP-hard [55]. Therefore in order to have a practical k-anonymized privacy-preserving technique, we need to come up with heuristic algorithms that provide us with a good approximation of the optimal solution for the k-anonymity problem.

Here we propose an algorithm for classifying n primary users into $g = \lceil \frac{n}{k} \rceil$ groups of k PUs, such that the k PUs in each group are the k closest neighbors. After grouping the PUs, the algorithm replace each group $G = \{g_1, g_2, \cdots, g_k\}$ with a virtual primary user Q that minimizes the equation:

$$\max_{1 \leq i \leq k} (d(g_i, Q) + R_i), \tag{5.8}$$

where $d(A, B)$ is the Euclidean distance between A and B, and R_i is the radius of the protected contour of primary user i, and then we set the radius of protected contour of Q as:

$$R_Q = \max_{1 \leq i \leq k} (d(g_i, Q) + R_i). \tag{5.9}$$

In other words we replace the group of k PUs in each group with one virtual PU such that its protected contour is the smallest protected contour that envelops the protected contour of all the k PUs.

The complexity of first part of the Algorithm 9 (grouping the n PUs into groups of k PUs) is:

$$O(n + (n-k) + (n-2k) + \cdots + (n - \frac{n}{k}k)) = O(\frac{n^2}{k})$$

The complexity of finding a point Q that minimizes equation 5.8 is $O(k \log k)$, so the com-

Algorithm 9 k-anonymity Algorithm for Preserving Location Privacy

Input: Set of Primary users $U = \{u_1, u_2, \cdots, u_n\}$ and their corresponding radius of protected contour $\{R_1, R_2, \cdots, R_n\}$.
Output: Groups of k closest PUs.

```
while U is nonempty do
    if |U| ≤ k then
        Put all members of U into a group G.
        Remove members of G from U.
    else
        Choose one member u_i of U randomly.
        Compute its distance from all other members of U.
        Choose the k − 1 closest members to u_i, and put them along with u_i to a group G.
        Remove the k members of G from U.
    end if
    for each group G = {g_1, g_2, ··· , g_k} do
        Find point Q that minimizes Equation 5.8.
        Compute the protected contour radius R_Q using Equation 5.9.
        return Q and R_Q.
    end for
end while
```

plexity of the second part of the algorithm is $\frac{n}{k} \times O(k \log k) = O(n \log k)$. Therefore the total complexity of this algorithm is $O(\frac{n^2}{k} + n \log k)$.

Although k-anonymity is widely used for preserving location privacy, but limiting the database to group the PUs into groups of equal size, may result in significant utilization loss. Figure 5.6 illustrates this limitation when using 3-anonymity to preserve location privacy of six primary users. We observe that PU4 is much closer to PU1, PU2, and PU3, but since 3-anonymity forces the database to divide PUs into groups of size 3, the database will put PU4 with PU5 and PU6 into one group and the large protected contour that envelops these three PUs causes significant waste of available spectrum. To address this limitation of k-anonymity, we propose another privacy-preserving technique in the next section.

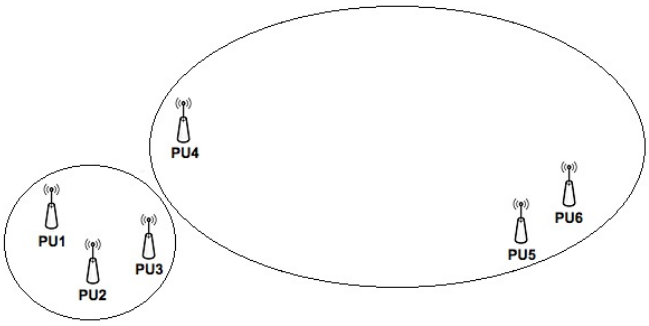


Figure 5.6: Limitation of using 3-anonymity for primary users' location privacy.

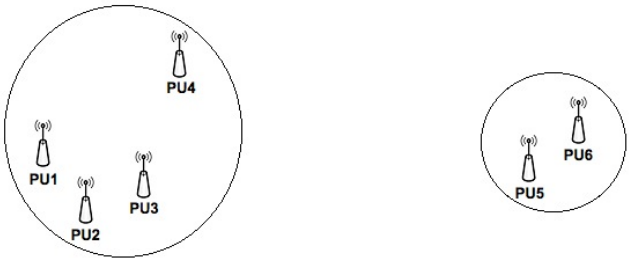


Figure 5.7: Using 2-clustering for preserving primary users' location privacy.

5.4.4 k-Clustering

In this section we propose another technique for preserving location-privacy using generalization that solves the problem of k-anonymity that we discussed in Section 5.4.3. In this technique, instead of classifying PUs into groups of k users, we group them into k clusters. Figure 5.7 illustrates applying 2-clustering algorithm, to the example mentioned in Figure 5.6.

Algorithm 10 works similar to Algorithm 9 but instead of dividing PUs into groups of equal size, divides them into k clusters of PUs that are closest together. These clusters may not be of equal size. The complexity of the Algorithm 10 (grouping the n PUs into k clusters of PUs) is $O(n^2)$.

The main disadvantage of using k-clustering instead of k-anonymity is that some PUs that

Algorithm 10 k-clustering Algorithm for Preserving Location Privacy

Input: Set of Primary users $U = \{u_1, u_2, \cdots, u_n\}$ and their corresponding radius of protected contour $\{R_1, R_2, \cdots, R_n\}$.
Output: k clusters of primary users.

1: Compute the distance d_{ij} between all pairs of PUs u_i and u_j.
2: Put d_{ij} values into a sorted array D.
3: Put each u_i into a single cluster.
4: **while** (number of clusters) $> k$ **do**
5: Choose the smallest value d_{ij} from array D.
6: Combine clusters of u_i and u_j.
7: **end while**
8: **for** each cluster $G = \{g_1, g_2, \cdots, g_m\}$ **do**
9: Find point Q that minimizes Equation 5.8.
10: Compute the protected contour radius R_Q using Equation 5.9.
11: **return** Q and R_Q.
12: **end for**

are far from other primary users may end up in a single cluster by itself. In other words we face the problem of providing unfair level of privacy for different PUs. This shortcoming can be addressed by using a hybrid approach and combining k-clustering with other privacy-preserving techniques such as perturbation with additive noise or transfiguration of protected contour that are designed for a single PU.

All of the privacy-preserving techniques that are proposed in this section preserve the location privacy of PUs at the cost of reducing spectrum utilization efficiency of SUs. In Section 5.6, we study this fundamental tradeoff further. But first we need metrics to quantify location privacy of stationary and mobile PUs to evaluate the impact of the inference attacks and the effectiveness of the countermeasures that we proposed.

5.5 Metric for Location Privacy of Primary Users

In [56], the authors discussed three general metrics for evaluating location inference attacks, namely *uncertainty*, *inaccuracy*, and *incorrectness*. In this section we define these metrics in the context of our problem to evaluate the PUs' location privacy.

Assume that $Y_{ij}^{(k)}$ is the database certain knowledge about the presence of a PU that operates on channel k in cell $c(i,j)$. $Y_{ij}^{(k)}$ is a deterministic function that is equal to 1 if a PU exists in cell $c(i,j)$ that operates on channel k, and is 0 otherwise.

Recall that $X_{ij}^{(k)}$ is a Bernoulli random variable that is equal to 1 if a PU exists in cell $c(i,j)$ that operates on channel k, and is 0 otherwise. This random variable shows the attacker's estimation of $Y_{ij}^{(k)}$. We see that the attacker uses the value, $p_{ij}^{(k)}$, to infer the location of the PUs.

5.5.1 Uncertainty

Suppose o denotes the observed sensory information (i.e., database's reply to the query), the attacker's extracted information is in the form of $p(X_{ij}^{(k)}|o)$, which is a probability distribution for the possible values of the PU's location given the observed information. *Uncertainty* is the ambiguity of this posterior distribution with respect to finding a unique answer (note that a unique answer need not be the correct one). We employ the concept of entropy to define $h_{ij}^{(k)}$, the uncertainty of the attacker about the presence of a PU that operates on channel k in cell $c(i,j)$:

$$h_{ij}^{(k)} = -p_{ij}^{(k)} \log(p_{ij}^{(k)}) - (1 - p_{ij}^{(k)}) \log(1 - p_{ij}^{(k)}). \tag{5.10}$$

Therefore the total uncertainty of the attacker about the location of primary users that are operating on channel k will be:

$$H^{(k)} = \sum_{i=1}^{M} \sum_{j=1}^{N} h_{ij}^{(k)}, \tag{5.11}$$

which we use as the metric for uncertainty of the attacker about the location of PUs that operate on channel k.

A bigger value for the attacker's uncertainty shows a better location privacy for the database. The uncertainty is maximized if the result of a location inference attack is a uniform distri-

bution of the locations.

We believe that uncertainty, which is a widely used metric for measuring privacy, is not suitable for location privacy of PUs. The reason is that the attacker might be certain about a wrong distribution. Figure 5.8 illustrates this shortcoming of uncertainty metric. The right figure shows the true distribution of the PUs, and the left figure shows the attackers estimation. Note that the attacker's uncertainty is zero, but this does not mean that the location privacy of the PU is threatened, because the attacker's estimation is wrong.

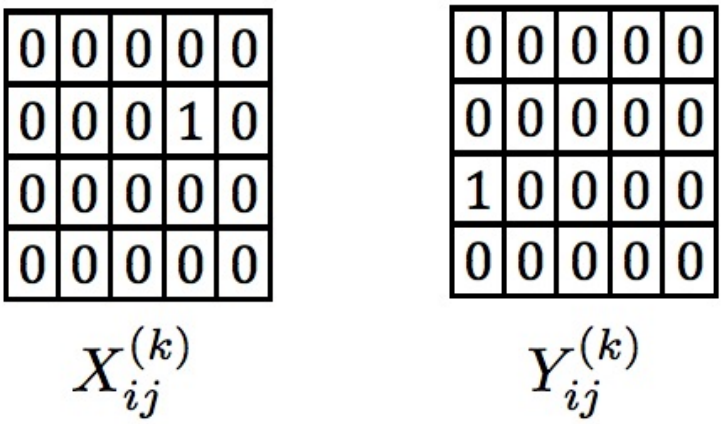


Figure 5.8: Example of disadvantage of using uncertainty metric.

5.5.2 Inaccuracy

Because the attacker does not have infinite resources, the result of a location inference attack is only an estimate, $p(X_{ij}^{(k)}|o)$, of the primary users' location distribution, $Y_{ij}^{(k)}$. *Inaccuracy* is the discrepancy between the distributions $p(X_{ij}^{(k)}|o)$ and $Y_{ij}^{(k)}$.

Here we define inaccuracy as:

$$IA^{(k)} = \sum_{i=1}^{M}\sum_{j=1}^{N}(p_{ij}^{(k)} - Y_{ij}^{(k)})^2. \tag{5.12}$$

Inaccuracy shows the deviation of attacker's estimation of PUs' location distribution from

the PUs' real location distribution.

We believe that inaccuracy is not an appropriate metric for quantifying location privacy of PUs. Figure 5.9 illustrates the shortcoming of inaccuracy metric. Assume that the $Y_{ij}^{(k)}$ is the real distribution of PUs and $X_{ij}^{(k)}$, and $\bar{X}_{ij}^{(k)}$ are two different estimations of $Y_{ij}^{(k)}$. The inaccuracy for both of these estimations is the same, but it is obvious that $\bar{X}_{ij}^{(k)}$ is a much better estimation.

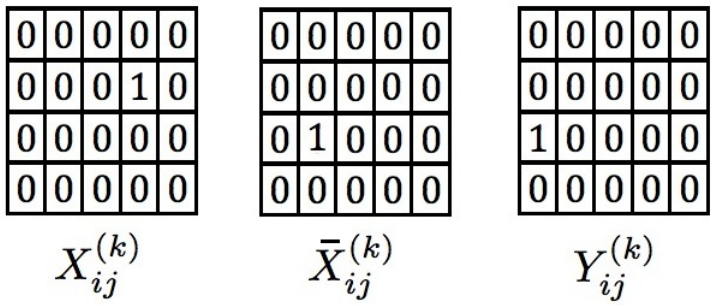


Figure 5.9: Example of disadvantage of using inaccuracy metric.

5.5.3 Incorrectness

We showed that uncertainty and inaccuracy are indirect measures for location privacy and can only be used to quantify the location privacy of a user relatively. The reason is that what matters for a user is whether the attacker finds the correct answer to his attack (i.e. its location), or alternatively, how close the attacker's output is to the correct answer. The database can calculate the distance (or expected distance) between the location inferred by the attacker and the true location. This distance is called the *incorrectness* of the attack.

We define incorrectness as:

$$IC^{(k)} = \sum_{i=1}^{M} \sum_{j=1}^{N} p_{ij}^{(k)} d_{(i,j)}^{k}, \tag{5.13}$$

where $d_{(i,j)}^{(k)}$ is the Euclidean distance between the cell $c(i,j)$ and the closest PU to it that operates on channel k.

In this dissertation, we use incorrectness as the metric for quantifying location privacy of PUs. The location privacy for stationary PUs, LP_s, is defined as:

$$LP_s = \sum_{k=1}^{C} IC^{(k)}. \tag{5.14}$$

To quantify the location privacy of mobile PUs, we need to consider the speed of the PU in addition to its location, thus we use a modified version of the incorrectness metric for mobile PUs. The location privacy of mobile PUs, LP_m, is defined as:

$$LP_m = \sum_{k=1}^{C} \sum_{i} w_i(k) d(x_i), \tag{5.15}$$

where $w_i(k)$ is the output of Algorithm 7 for the weight of particle x_i on channel k, and $d(x_i)$ is the distance between this particle and the particle that describes the actual kinematics of the closest PU's path of movement. Assuming that $x_i = (loc_i, vel_i)$ and the closest PU's kinematics is represented as (L, V), the distance $d(x_i)$ is defined as:

$$d(x_i) = d_e(loc_i, L) + |vel_i - V| \tag{5.16}$$

We use these metrics later in this chapter to evaluate the effect of the location inference attacks and effectiveness of the proposed countermeasures.

5.6 PLOT: A Technique for Mitigating Location Inference Attacks

In this section, we propose a privacy-preserving technique called *Primary users' Location Obfuscation Technique* ($PLOT$) that a database can employ to mitigate the location inference attacks that were introduced in Sections 5.2 and 5.3. Although the proposed countermeasure

was designed to counter those specific attacks, it is also effective in mitigating the effect of other attacks that use the same source of information to infer the location of PUs.

PLOT uses *query history retention*, which means that it requires the database to store a history of queries from a user or a group of users, and analyzes them in order to detect inference attacks. Identifying collusion attacks against a database is a nontrivial problem. Certain types of clustering algorithms can be used to group queriers that are likely to collude. These algorithms group queriers based on common or similar characteristics, such as the similarity in query times, their location, and resource usage [57], [58]. In this chapter, we assume that the database uses an effective method to identify queriers that are likely to collude. However, specifics of such a method are outside the scope of this paper. In our simulations, we assumed that all queriers belong to one collusion group.

The core idea of PLOT is to use the inference algorithms given in Algorithms 6 and 7 to "estimate" the amount of knowledge that, possibly colluding, attackers have accumulated about the location of a particular PU. If the database determines that the amount of accumulated knowledge is beyond a certain threshold, then it replies to a (suspected) attacker's query with false information using the perturbation with additive noise technique that is described in Section 5.4.1. Even if the attackers were to use location inference algorithms that are different from Algorithms 6 and 7 but keep using the same source of information (i.e., database's replies to SUs' queries), PLOT would be effective, because it contaminates the knowledge base of the location inference algorithm by providing false information. The proposed approach of using query history retention empowered by recursive Bayesian estimation was shown to be a powerful countermeasure against database inference attacks [57], [59].

Recall that the perturbation with additive noise technique preserves the privacy of PU's location by providing a *false positive* response to a query that corresponds to replacing the actual transmission power P_k with P'_k, where $P'_k \leq P_k$ for each k. Note that a false negative response, which corresponds to $P'_k > P_k$, cannot be used by the database because it will likely result in interference to PUs.

5.6.1 A Metric for Performance Loss

The proposed privacy-preserving scheme makes a tradeoff between the PUs' location privacy and the SUs' spectrum utilization. Specifically, the scheme decreases the SUs' allowed maximum transmission power to enhance the PUs' location privacy. This decrease in power can affect a SU network's performance in a number of different ways. For example, it can decrease a SU network's coverage area; cause a drop in throughput due to the use of a lower modulation rate or lower code rate forward error correction (FEC); or cause a reduction in network capacity. Therefore, we have chosen a simple yet direct approach for quantifying the impact of the decrease in transmission power. This impact is quantified using a metric called *performance loss* (PL) which is defined as

$$PL = \sum_{k=1}^{C} (P_k - P'_k). \tag{5.17}$$

5.6.2 Optimal Strategy for Mitigating Location Inference Attacks

We define the database's optimal strategy as a strategy in which the PUs' location privacy is maximized subject to upper bounding the performance loss by the value PL^{max}. This strategy is equivalent to solving the following optimization problem when the PUs are stationary.

$$\text{Maximize } \sum_{k=1}^{C} \sum_{i,j} p_{ij}^{(k)} d_{(i,j)},$$

$$\text{subject to } \sum_{k=1}^{C} (P_k - P'_k) \leq PL^{max},$$

where $p_{ij}^{(k)}$ is an output value of Algorithm 6 obtained using information from the query response $\{(ch_k, P'_k, t_k)\}$. For mobile PUs, the optimization problem is defined as follows:

$$\text{Maximize } \sum_{k=1}^{C} \sum_{i} w_i(k) d(x_i),$$

$$\text{subject to } \sum_{k=1}^{C} (P_k - P'_k) \leq PL^{max},$$

where $w_i(k)$ is an output of Algorithm 7 obtained using information for the query response $\{(ch_k, P'_k, t_k)\}$.

The difficulty of solving the above optimization problems depends on the relation between the maximum transmission power, P_k, and probabilities, $p_{ij}^{(k)}$ and $w_i(k)$, which in turn depends on the MTP function. For example, when a two-way propagation model is used for computing the MTP function, then $h(d) \propto d^2$ [41]; this makes the above problems convex optimization problems that can be solved in $O(C^3)$ time [39]. For our simulations, we used a simple function given by Equation 5.7 to obtain values for $h(d)$, which enable us to solve the optimization problems using a brute-force approach.

5.7 Evaluation of PLOT

In this section, we evaluate the efficacy of Algorithms 6 and 7, and also provide simulation results for PLOT. Note that in order to have a common scale for location privacy and performance loss metrics with different grid's sizes and maximum allowable transmission power (P^{max}) values, we use normalized values of the metrics defined in Equations 5.15 and 5.17.

In the first experiment, we use a 100×100 grid of square cells with a side length of 1 km. The available spectrum consists of 10 channels. We assume that there are 10 stationary PUs: there are two PUs operating in each of the channels 1, 2, and 8, and one PU is operating in

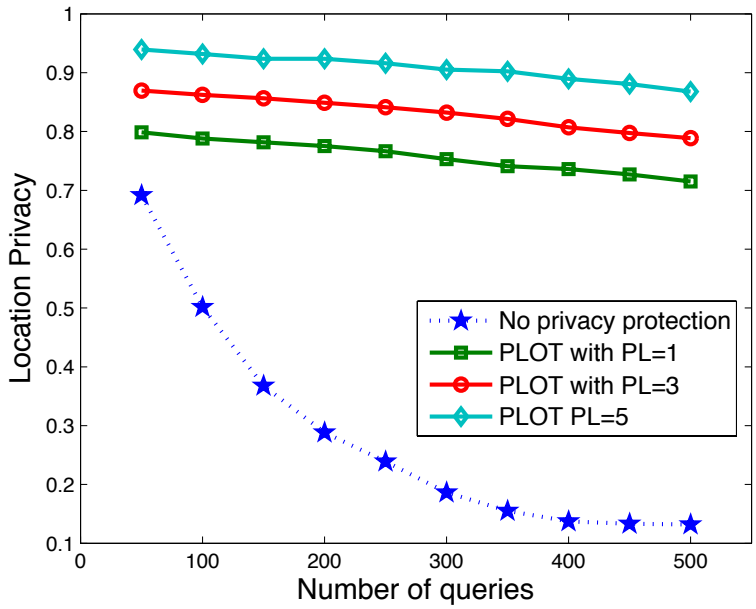


Figure 5.10: LP_s vs. number of queries for stationary primary users.

each of the channels 3, 5, 6, and 7. The maximum allowed transmission power is computed using Equation 5.7.

Figure 5.10 plots the privacy metric, LP_s, vs. the number of queries for stationary PUs. As expected, when the database does not use PLOT, the value of LP_s drops quickly as the number of queries increase. However, the figure shows that when PLOT is employed, the decrease in LP_s is much more gradual. The figure also shows that for a given number of queries, a higher value of LP_s can be achieved with an increase in the PL value. This result confirms that PLOT makes a tradeoff between the location privacy of the PUs and the SUs' spectrum utilization. Moreover, our results indicate that the location privacy of the incumbent systems cannot be assured without sacrificing spectrum utilization efficiency; in other words, there exists a fundamental tradeoff problem between the PUs' location privacy and spectrum utilization efficiency.

Figure 5.11 compares PLOT with a simple benchmark strategy that responds to queries with random false positives. To reply to a query, while performance loss is less that PL^{max}, this

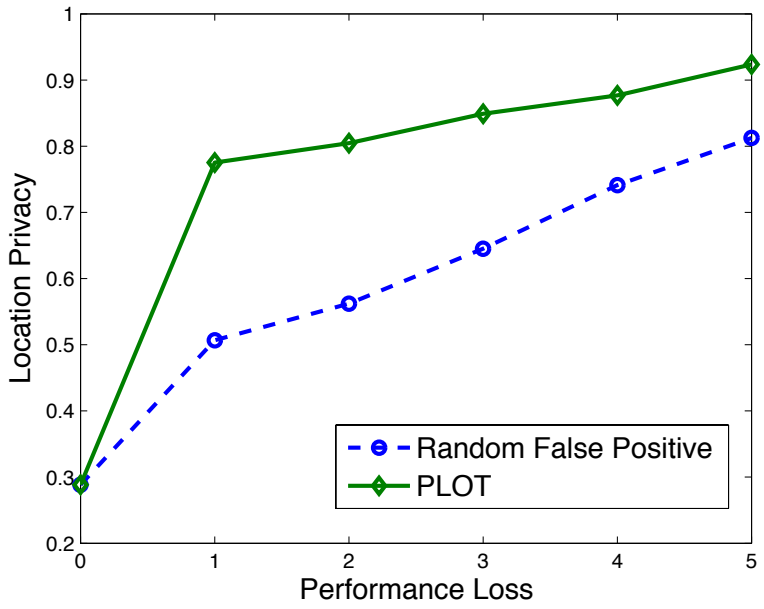


Figure 5.11: LP_s vs. performance loss for stationary primary users.

strategy selects a channel k randomly and replaces P_k with a random value P'_k such that $P'_k < P_k$. The figure shows the results after 200 queries. As it can be seen PLOT outperforms the random false positive strategy, especially when the limitation on the performance loss is more strict (e.g. PL = 1), the difference in the level of privacy that the two schemes can provide is much larger.

In order to assess the effectiveness of PLOT against the location inference algorithm 7 we setup the following experiment on a set of PUs with simple movement patterns. We use a 100×100 grid of square cells with a side length of 1 km. There are five PUs, three of them move from the cells (10,1), (50,1), and (90,1) move north, and the two of them move south from the cells (30,100) and (70,100), all with a speed of 1 km/min. Each PU is operating on a different channel. At each iteration of the location inference algorithm, a SU with a random location inside the grid queries the database and the database using the minimum distance between the location of the SU and the current location of all the PUs, computes the maximum allowed transmission power and returns it to the SU. At this step the database

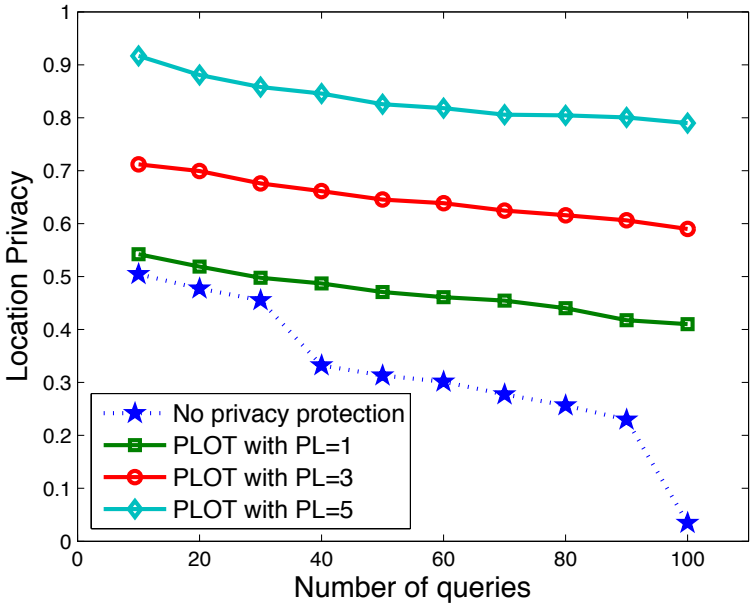


Figure 5.12: LP_m vs. number of queries for mobile primary users.

runs the defense strategy which decides on which channels the database should return a lesser allowed transmission power as a false positive. To compute the maximum allowed power, we use the same function that is given in Equation 5.7.

The location inference algorithm uses 90000 particles (x, y, v_x, v_y) where $1 \leq x, y \leq 100$ and $-1 \leq v_x, v_y \leq 1$ where all values are integers. The algorithm converges fast and if the database do not use a privacy protection mechanism, it find the location and speed of PU with a high accuracy. At each iteration of the attack, a SU with a random location on the grid, queries the database. Figure 5.12 plots the location privacy metric LP_m vs. the number of queries for mobile PUs. As expected, when the GDB does not use PLOT, the values of LP_m drops quickly. The two sudden drop in LP_m between 30th and 40th queries indicates that a location inference happened at this range (in fact the algorithm has found the location of two PUs moving south). The other sudden drop in LP_m between 90th and 100th queries) in case that PLOT is not applied show another location inference has happened between 90th and 100th queries. The figure also confirms the tradeoff between location privacy and

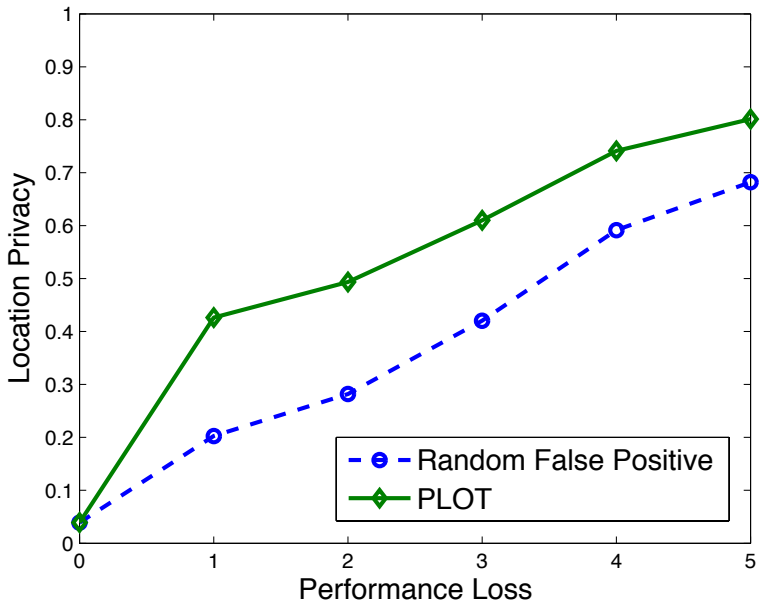


Figure 5.13: LP_m vs. performance loss for mobile primary users.

performance of secondary network, because for a given number of queries, a higher value of LP_m can be achieved with an increase in the PL value.

Figure 5.13 compares PLOT with a simple benchmark strategy that responds to queries with random false positives. As we can see, PLOT is superior to a random false positive strategy, especially where only a very limited amount of performance loss is allowed.

5.8 Summary

In this chapter, we addressed the problem of preserving the location privacy of PUs in a database-driven cognitive radio network. We identified two low-cost attacks against stationary and mobile PUs that can find the location of PUs using a set of seemingly innocuous queries and extracting information from the databases replies. In order to preserve the location privacy of PUs against these inference attacks, we proposed four privacy-preserving techniques. We also proposed a privacy protection mechanism that can mitigate the effect

of the identified location inference attacks by making an optimal tradeoff between location privacy and spectrum utilization efficiency.

Chapter 6

Conclusions and Future Research Directions

In this dissertation, we proposed ex ante approaches to help spectrum enforcement by mitigating security and privacy threats such as rogue transmissions, belief manipulation attacks, and location privacy violations.

6.1 Conclusions

In order to enforce trustworthy transmission behavior via spectrum access policies in policy-based cognitive radios, we introduced a novel policy reasoner (BRESAP) in chapter 2. The proposed policy reasoner uses multi-terminal binary decision diagrams (MTBDDs) to represent, interpret, and process policies in an efficient manner. We presented a systematic approach to translate policies into MTBDDs, merge policies into a single meta-policy, and compute opportunity constraints. We have also proposed three different graph-based algorithms for reasoning and computing the opportunity constraints.

The major drawback of BRESAP is that it predefines the attributes of the spectrum policies

and hard-wires them into the spectrum policy reasoner. BRESAP cannot process policies whose attributes have not been predetermined in advance. BRESAP can process such policies only after modifying the reasoning software itself. To address these drawbacks, we describe the design of a policy reasoner that processes ontology-based spectrum policies in chapter 3. The main advantage of using ontology-based policies is that the policy reasoner can understand and process any spectrum policies authored by any organization by relying on the spectrum ontologies. Based on the insights gained from the experiments that we run on our ontology-based policy reasoner, we also provided a set of guidelines for designing ontology-based spectrum policies.

Another important security issue in CR networks is to make the machine learning algorithms that are used in CR engine robust against belief manipulation attacks. In chapter 4, we investigated channel selection policies that use reinforcement learning algorithms. We showed that these algorithms are vulnerable to belief manipulation attacks and proposed a countermeasure to use randomness in order to make the algorithm robust against these attacks while keeping performance above a threshold.

Finally in chapter 5, we investigated the problem of preserving the location privacy of PUs in a database-driven cognitive radio network. Using two algorithms, we demonstrated that SUs can readily infer the locations of stationary and mobile incumbent systems using seemly innocuous database queries. Specifically, we described two location inference attacks. The first attack focused on locating PUs that are stationary, and the second attack used recursive Bayesian estimation techniques to find the location of a mobile PU. These attacks demonstrate the potential seriousness of the incumbents location privacy problem. In order to preserve the location privacy of PUs against these inference attacks, we proposed four privacy-preserving techniques. We also proposed a privacy protection mechanism called PLOT that can mitigate the effect of the identified location inference attacks by making an optimal tradeoff between location privacy and spectrum utilization efficiency. PLOT uses query history retention and location inference algorithms to estimate the amount of knowledge accumulated by the attackers, and then determines when and how to obfuscate the

database query replies by using perturbation with additive noise.

6.2 Future Research Directions

There are a number of challenges related to spectrum policies, including the development of advanced algorithms for executing policy inference and reasoning tasks carried out by policy-based cognitive radios. Despite the great potential of ontology-based spectrum policies, there is slow progress in integrating this concept into policy-based cognitive radios because of the complexity of policy inference and reasoning when the policies are ontology-based. The primary challenge in using ontology-based policies is meeting the real-time processing requirements of the radio. To date, ontology-based policies have been successfully applied to interactive, non-real-time applications, but not to real-time applications. Most of the policy inference and reasoning tasks carried out by a policy-based cognitive radio need to be executed within a very tight time window.

Although our ontology-based policy reasoner that is described in Chapter 3 can compute the transmission reply in real-time, it is not able to find and return transmission opportunity constraints within the required latency criteria of a CR. Our investigations show that the main reason is the slowness of the basic Rete algorithm (a.k.a. Rete I) [22] in the PR. The basic Rete algorithm is an efficient pattern-matching algorithm for implementing production rule systems that is freely available. Although Rete I is very good at handling large numbers of rules efficiently, it is less efficient when it is called on to handle large amounts of data or very rapidly changing data. Our experimental evaluation results show that in order to meet the real-time requirements of a cognitive radio, we need an alternative algorithm that performs 2-3 times faster than Rete I algorithm. Another approach for speeding up the ontology-based policy reasoner is using parallelism. It is proved that the Rete-class of algorithms is highly suitable for parallel implementation and its execution speed can be improved significantly if we evaluate activation of different nodes in the Rete network in parallel [60].

Identifying collusion attacks against location privacy of PUs in a geolocation database is another challenging problem that need to be addressed. Certain types of clustering algorithms can be used to group queriers that are likely to collude. These algorithms group queriers based on common or similar characteristics, such as the similarity in query times, their location, and resource usage [57], [58]. Designing such an effective algorithm to identify queriers that are likely to collude will significantly improve the optimal tradeoff between location privacy of PUs and spectrum utilization efficiency of SUs.

Another issue that need to be investigated in the future is the consideration of protection zones in evaluating the PUs' location privacy. All of the inference attacks and countermeasures that are proposed in this report use exclusion zones as the protected contour of primary users. In its final report, CSMAC proposed to eliminate exclusion zones entirely in favor of protection zones. This would allow SU operation as long as the aggregate received co-channel interference at the PU antenna is below a yet to be determined threshold. CSMAC claims that these protection zones are smaller than exclusion zones (14-95 km vs 72-121 km, depending on the location in question). According to the CSMAC report, the protection zones are smaller because they are based on more realistic propagation models rather than the worst case ones underlying the exclusion zones [61].

In Section 5.4 we proposed four privacy-preserving techniques for location privacy of PUs in a geolocation database. The effectiveness of these techniques highly depends on the geometrical arrangement of PUs in the grid. Figures 5.6 ad 5.7 illustrated an example of the impact of the orientation of PUs on the effectiveness of k-anonymity and k-clustering techniques. Finding a hybrid technique which uses a combination of the proposed privacy-preserving methods depending on the geometrical arrangement of PUs is another research direction that should be further investigated.

Bibliography

[1] N. O. Tippenhauer, K. B. Rasmussen, C. Popper, and S. Capkun, "Attacks on public WLAN-based positioning," in *ACM MobiSys*, 2009.

[2] A. Ginsberg, W. D. Horne, and J. D. Poston, "Community-based cognitive radio architecture: Policy-compliant innovation via the semantic web," in *IEEE DySPAN*, pp. 191–201, 2007.

[3] F. Perich and M. McHenry, "Policy-based spectrum access control for dynamic spectrum access network radios," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 7, pp. 21–27, 2009.

[4] A. Toninelli, J. Bradshaw, L. Kagal, and R. Montanari, "Rule-based and ontology-based policies: Toward a hybrid approach to control agents in pervasive environments," in *Semantic Web and Policy Workshop*, pp. 42–54, Sept. 2005.

[5] M. Kokar and L. Lechowicz, "Language issues for cognitive radio," *Proc. IEEE*, 2009.

[6] T. Clancy and N. Goergen, "Security in cognitive radio network.: Threats and mitigation," in *CrownCom*, pp. 1–8, May 2008.

[7] F. C. Commission, "Third order and memorandum opinion and order, in the matter of unlicensed operation in the tv broadcast bands, additional spectrum for unlicensed devices below 900 mhz and in the 3 ghz band," April 2012.

[8] D. Gurney, G. Buchwald, L. Ecklund, S. Kuffner, and J. Grosspietsch, "Geo-location database techniques for incumbent protection in the TV white space," in *IEEE DySPAN*, pp. 1–9, 2008.

[9] J. M. III, *Cognitive radio: an integrated agent architecture for software defined radio*. PhD thesis, KTH Royal Institute of Technology, 2000.

[10] D. Wilkins, G. Denker, M. Stehr, D. Elenius, and R. Senanayake, "Policy-based cognitive radios," *IEEE Wireless Communications Magazine*, vol. 14, pp. 41–46, 2007.

[11] D. X. W. Group, "The xg vision, request for comments," 2003.

[12] B. A. Fette, ed., *Cognitive Radio Technology*. Elsevier, 2009.

[13] D. Elenius, G. Denker, M. Stehr, R. Senanayake, C. Talcott, and D. Wilkins, "Coral - policy language and reasoning techniques for spectrum policies," in *Policies for Distributed Systems and Networks, POLICY'07*, pp. 261–265, 2007.

[14] P. F. P.-S. I. Horrocks, H. Boley, S. Tabel, B. Grosof, , and M. Dean, "Swrl: A semantic web rule language combining owl and ruleml." W3C Member submission, 2004.

[15] H. R. Andersen, "An introduction to binary decision diagrams." Lecture Notes, IT University of Copenhagen, 1997.

[16] M. Fujita, P. C. McGeer, and J. C.-Y. Yang, "Multi-terminal binary decision diagrams: An efficient datastructure for matrix representation," *Formal Methods in System Design*, vol. 10, pp. 149–169, 1997.

[17] P. Rao, D. Lin, E. Bertino, N. Li, and J. Lobo, "An algebra for fine-grained integration of xacml policies," in *14th ACM symposium on Access control models and technologies*, pp. 63–72, 2009.

[18] C. Llano, E. Asensio, and F. Fuentes, "Variable ordering schemes to apply to the binary decision diagram methodology for event tree sequences assessment," *Journal of Risk and Reliability*, vol. 222, pp. 7–16, 2008.

[19] M. Sauerhoff, I. Wegener, and R. Werchner, "Optimal ordered binary decision diagrams for fanout-free circuits," in *The sixth workshop on Synthesis And System Integration of MIxed Technologies (SASIMI '96)*, pp. 197–204, 1996.

[20] K. Fisler, S. Krishnamurthi, L. A. Meyerovich, and M. C. Tschantz, "Verification and change-impact analysis of access-control policies," in *International Conference on Software Engineering (ICSE)*, pp. 196–205, 2005.

[21] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, , and P. Schneider, *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.

[22] C. L. Forgy, "Rete: A fast algorithm for the many pattern/many object pattern match problem," *Artificial intelligence*, vol. 19, no. 1, pp. 17–37, 1982.

[23] N. Guarino, "Formal ontology and information systems," in *International Conference on Formal Ontology in Information Systems*, pp. 3–15, 1998.

[24] M. Kokar, D. Brady, and K. Baclawski, *Cognitive Radio Technology*, ch. Chapter 13: The role of ontologies in cognitive radios, pp. 401–428. Elsevier, 2009.

[25] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, "Pellet: A practical owl-dl reasoner," *Web Semantics: science, services and agents on the World Wide Web*, vol. 5, no. 2, pp. 51–53, 2007.

[26] P. V. Biron, K. Permanente, and A. Malhotra, "Xml schema part 2: Datatypes second edition." W3C recommendation, 2004.

[27] S. Bechhofer, F. V. Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein, "Owl web ontology language reference." W3C recommendation, 2004.

[28] S. Menard, "Jpype project," 2010.

[29] X. team of SRI, "Xg communication policies and ontologies."

[30] Y. Chen, Q. Zhao, and A. Swami, "Joint design and separation principle for opportunistic spectrum access in the presence of sensing errors," *IEEE Transactions on Information Theory*, vol. 54, no. 5, 2008.

[31] S. Ahmad and M.Liu, "Multi-channel opportunistic access: a case of restless bandits with multiple plays," in *Allerton Conference on Communication, Control, and Computing*, 2009.

[32] E. Gilbert, "Capacity of burst-noise channels," *Bell System Technical Journal*, vol. 39, no. 9, pp. 1253–1265, 1960.

[33] R. Smallwood and E. Sondik, "The optimal control of partially observable markov processes over a finite horizon," *Operations Research*, pp. 1071–1088, 1971.

[34] M. Li, I. Koutsopoulos, and R. Poovendran, "Optimal jamming attack strategies and network defense policies in wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 9, no. 8, pp. 1119–1133, 2010.

[35] A. Wald, *Sequential Analysis*. Wiley, 1947.

[36] Q. Zhao, B. Krishnamachari, and K. Liu, "On myopic sensing for multi- channel opportunistic access: structure, optimality, and performance," *IEEE Transactions on Wireless Communications*, vol. 7, no. 12, pp. 5431–5440, 2008.

[37] R. S. Sutton and A. G. Bareto, *Reinforcement Learning: An Introduction*. MIT press, 1998.

[38] S. Ahmad, M. Liu, T. Javidi, Q. Zhao, and B. Krishnamachari, "Optimality of myopic sensing in multi-channel opportunistic access," *IEEE Transactions on Information Theory*, vol. 55, no. 9, pp. 4040–4050, 2009.

[39] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[40] M. Ohya and M. Mizu, "An information-theoretical approach and its application to optimal problems," *Electronics and Communication in Japan*, vol. 76, no. 6, 1993.

[41] D. Gurney, G. Buchwald, L. Ecklund, S. Kuffner, and J. Grosspietsch, "Geo-location database techniques for incumbent protection in the tv white space," in *IEEE Dyspan*, 2008.

[42] F. C. Commission, "Enabling innovative small cell use in 3.5 ghz band nprm & order (fcc 12-148)," December 2012.

[43] NSF, "Enhancing access to the radio spectrum (ears)." Program Solicitation NSF 13-539.

[44] DARPA, "Baa: Shared spectrum access for radar and communications (ssparc)." DARPA-BAA-13-24, February 2013.

[45] C. M. Cordeiro, K. Challapali, and D. Birru, "IEEE 802.22: An introduction to the first wireless standard based on cognitive radios," *Journal of Communication*, vol. 1, pp. 328–337, April 2006.

[46] "Tv white space database." [Online]. Available: http://whitespaces.spectrumbridge.com/.

[47] N. Bergman, *Recursive Bayesian Estimation*. PhD thesis, Department of Electrical Engineering, Linköping University, 1999.

[48] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation: Theory Algorithms and Software*. John Wiley & Sons Inc., 2001.

[49] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002.

[50] A. Kong, J. S. Liu, , and W. H. Wong, "Sequential imputations and bayesian missing data problems," *Journal of the American Statistical Association*, vol. 89, no. 425, pp. 278–288, 1994.

[51] C. Aggarwal and S. Phillip, *Privacy-Preserving Data Mining: Models and Algorithms*, ch. A general survey of privacy-preserving data mining models and algorithms. Springer, 2008.

[52] P. Samarati and L. Sweeney, "Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression," in *IEEE Symposium on Research in Security and Privacy*, 1998.

[53] L. Sweeney, "k-anonymity: A model for protecting privacy," *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, vol. 10, no. 5, pp. 557–570, 2002.

[54] B. Gedik and L. Liu, "Protecting location privacy with personalized k-anonymity: Architecture and algorithms," *IEEE transaction on mobile computing*, vol. 7, no. 1, 2008.

[55] G. Aggrawal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu, "Approximation algorithms for k-anonymity," *Journal of Privacy Technology*, 2005.

[56] R. Shokri, G. Theodorakopoulos, J. Boudec, and J. Hubaux, "Quantifying location privacy," in *IEEE Symposium on Security and Privacy*, 2011.

[57] M. Hylkema, "A survey of database inference attack prevention methods," tech. rep., Boston University, December 2009.

[58] G. Palshikar and M. Apte, "Collusion set detection using graph clustering," *Data Mining and Knowledge Discovery*, vol. 16, no. 2, pp. 135–164, 2008.

[59] Y. Chen and C. Wesley, "Database security protection via inference detection," *Intelligence and Security Informatics*, pp. 452–458, 2006.

[60] A. Gupta, C. Forgy, A. Newell, and R. Wedig, "Parallel algorithms and architectures for rule-based systems," *ACM SIGARCH Computer Architecture News*, vol. 14, no. 2, pp. 28–37, 1986.

[61] M. Weiss, M. Altamimi, and M. McHenry, "Enforcement and spectrum sharing: A case study of the 1695-1710 MHz band," in *CrownCom*, 2013.