

Cognitive Networks: Foundations to Applications

Daniel H. Friend

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 and Computer Engineering

Dr. Allen B. MacKenzie, Chair

Dr. Charles Bostian

Dr. Michael Buehrer

Dr. Luiz A. DaSilva

Dr. Madhav Marathe

March 6, 2009

Blacksburg, Virginia

Keywords: cognitive networks, reasoning and learning, distributed optimization, genetic algorithm, Markov decision process, mobile ad hoc network, routing, dynamic spectrum access, channel allocation, multichannel topology control

Copyright 2009, Daniel H. Friend

Cognitive Networks: Foundations to Applications

Daniel H. Friend

(ABSTRACT)

Fueled by the rapid advancement in digital and wireless technologies, the ever-increasing capabilities of wireless devices have placed upon us a tremendous challenge – how to put all of this capability to effective use. Individually, wireless devices have outpaced the ability of users to optimally configure them. Collectively, the complexity is far more daunting. Research in cognitive networks seeks to provide a solution to the difficulty of effectively using the expanding capabilities of wireless networks by embedding greater degrees of intelligence within the network itself.

In this dissertation, we address some fundamental questions related to cognitive networks, such as “What is a cognitive network?” and “What methods may be used to design a cognitive network?” We relate cognitive networks to a common artificial intelligence (AI) framework, the multi-agent system (MAS). We also discuss the key elements of learning and reasoning, with the ability to learn being the primary differentiator for a cognitive network.

Having discussed some of the fundamentals, we proceed to further illustrate the cognitive networking principle by applying it to two problems: multichannel topology control for dynamic spectrum access (DSA) and routing in a mobile ad hoc network (MANET). The multichannel topology control problem involves configuring secondary network parameters to minimize the probability that the secondary network will cause an outage to a primary user in the future. This requires the secondary network to estimate an outage potential map, essentially a spatial map of predicted primary user density, which must be learned using prior observations of spectral occupancy made by secondary nodes. Due to the complexity of the objective function, we provide a suboptimal heuristic and compare its performance against heuristics targeting power-based and interference-based topology control objectives. We also develop a genetic algorithm to provide reference solutions since obtaining optimal solutions

is impractical. We show how our approach to this problem qualifies as a cognitive network.

In presenting our second application, we address the role of network state observations in cognitive networking. Essentially, we need a way to quantify how much information is needed regarding the state of the network to achieve a desired level of performance. This question is applicable to networking in general, but becomes increasingly important in the cognitive network context because of the potential volume of information that may be desired for decision-making. In this case, the application is routing in MANETs. Current MANET routing protocols are largely adapted from routing algorithms developed for wired networks. Although optimal routing in wired networks is grounded in dynamic programming, the critical assumption, static link costs and states, that enables the use of dynamic programming for wired networks need not apply to MANETs. We present a link-level model of a MANET, which models the network as a stochastically varying graph that possesses the Markov property. We present the Markov decision process as the appropriate framework for computing optimal routing policies for such networks. We then proceed to analyze the relationship between optimal policy and link state information as a function of minimum distance from the forwarding node.

The applications that we focus on are quite different, both in their models as well as their objectives. This difference is intentional and significant because it disassociates the technology, i.e. cognitive networks, from the application of the technology. As a consequence, the versatility of the cognitive networks concept is demonstrated. Simultaneously, we are able to address two open problems and provide useful results, as well as new perspective, on both multichannel topology control and MANET routing.

This work was supported by a Bradley Fellowship from the Bradley Department of Electrical and Computer Engineering at Virginia Tech, with additional funding provided by internships with MIT Lincoln Laboratory in Lexington, MA under the supervision of Dr. Andrew Worthen. This material is also based upon work supported by the National Science

Foundation under Grant No. 0448131. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

Acknowledgments

I've always felt that waiting until the end to thank God makes it look like an afterthought, so I would prefer to thank God up front. Why do people thank God in their acknowledgments? I always kind of wonder, since most people don't provide specifics. In my case, it's because I didn't really know what I was getting into when I started this program, but I knew that I would need His help along the way. Several times, after spending hours, days, and even weeks trying to find direction in my research or trying to solve some problem, I would suddenly realize that it was time to ask for help, and I would pray. He never let me down, and I hope that I will never let Him down.

When I first proposed the idea of returning to school for a Ph.D. to my wife, I was amazed that she was so supportive. Three and a half years later, I realize that I could have explained it a little differently, with perhaps a vastly different outcome. I could have said, "Sweetheart, I could buy you a Range Rover and we could travel the world, or I could quit my job and go back to school for a Ph.D." I've always seen this venture as an investment, but only now do I realize how much of that investment was made by others. Only time will tell, but I have faith that this investment will provide returns that could not have come in any other way, even if there never is a Range Rover in the driveway.

Before coming out for Graduate Recruiting weekend in 2005, I had never been to Virginia Tech. Had it not been for that trip, I may have made the tragic mistake of going somewhere else. That visit changed everything, and I returned home with a great admiration of the university and the people here. It's a beautiful place, full of honest and just plain good

people. I'm tremendously grateful to have been given a Bradley Fellowship to fund my research. It has been an honor and has both shortened the amount of time that it has taken to graduate and relieved some of the financial burden. MIT Lincoln Lab has filled in the summer gaps for the first two years. They've always done more for me than I asked of them, even though what I asked for was often more than I expected. I only wish that they were located in a more temperate climate.

When I was initially trying to choose an advisor, my primary concern was finding someone that would be supportive of my desire to take care of my family and have something of a normal work schedule. My initial impression was that Dr. MacKenzie was the kind of advisor who would be supportive of the needs of my family, and he has more than lived up to this. In fact, he has taught me even more about the importance of family and balancing the demands of life. I admire the many great qualities he possesses and appreciate the sincere concern that he displays for his students' welfare.

Contents

1	Introduction	1
2	Background on Cognitive Networks	5
2.1	Definition	6
2.2	Motivation and Requirements	6
2.3	Foundations and Related Work	7
2.3.1	Cognitive Radio	8
	Shared Attributes with Cognitive Networks	8
	Differences from Cognitive Networks	8
2.3.2	Cross-Layer Design	9
	Shared Attributes with Cross-Layer Design	9
	Differences from Cross-Layer Design	10
2.3.3	Cognitive Networks in the Literature	11
2.4	Generic Cognitive Network Hierarchy	11
2.4.1	User/Application/Resource Requirements	12
2.4.2	The Cognitive Process	13
2.4.3	Software Adaptable Network	15
3	Learning and Reasoning: Key Elements of Cognitive Networks	16
3.1	Frameworks for Learning and Reasoning	17
3.1.1	Linking Cognitive Architectures to Cognitive Network Architectures .	18
3.1.2	The Cognitive Network as a Multiagent System	19

3.2	Learning and Reasoning within a MAS Framework	20
3.2.1	Complexity and Exploitation of Domain Knowledge	21
3.2.2	Elements of Reasoning	22
3.2.3	Generic Methods for Reasoning	23
3.2.3.1	Distributed Constraint Reasoning	23
3.2.3.2	Bayesian Networks	24
3.2.3.3	Metaheuristics	26
	Parallel Genetic Algorithms	27
	Scatter Search	28
	Tabu Search	28
3.2.4	Reasoning by Heuristic	29
3.2.4.1	Job Shop Scheduling	30
3.2.4.2	Robot Motion Planning	30
3.2.4.3	Delay-constrained Least-cost Routing	32
3.2.5	Multiobjective Reasoning	33
3.2.6	Elements of Learning	33
3.2.7	Methods for Learning	35
3.2.7.1	Q-learning	36
3.2.7.2	Learning of Bayesian Networks	36
3.2.7.3	Case-based Reasoning	37
3.3	Sensory and Actuator Functions	39
3.4	Cognitive Node Architecture	41
3.4.1	Reconfigurable Platform, Stack Manager, and COD	41
3.4.2	Exchange Controller	43
3.4.3	Cognitive Controller and Distributed Optimization Process	43
4	Secondary Network Multi-channel Topology Control	45
4.1	Related Work	48
4.2	Channel and Network Models	49

4.2.1	Path Loss and Outage Probability	49
4.2.1.1	Calculating Outage Probability	50
4.2.2	Network Model	51
4.2.2.1	The MAC Protocol	53
4.2.2.2	Conflict Graph	53
	Maximal Independent Sets	54
4.3	Outage Potential Map	54
4.3.1	Maximum Likelihood Position Estimation for Outage Potential Maps	56
4.4	MinMaxEo Topology Control	57
4.4.1	Formalizing the Objective	57
4.4.2	Issues in Finding the Optimum Topology	58
4.4.3	Randomized Algorithm for Estimating Single-Channel Expected Out- age Potential	59
4.4.4	Single-channel Heuristic	65
4.4.5	Multi-channel Heuristic	67
4.4.5.1	Stage 1	67
4.4.5.2	Stage 2	67
4.4.5.3	Stage 3	68
4.5	A Power-based Multi-channel Topology Control Problem	70
4.5.1	Solving MinMaxP	71
4.5.2	Metaheuristic for Evaluating MinMaxP	75
4.5.3	Performance of the MinMaxP Heuristic vs. the GA	76
4.6	Performance Comparison	77
4.6.1	Single-channel Heuristic Performance	77
4.6.1.1	Single-channel Simulation Parameters	80
4.6.1.2	Single-channel Simulation Results	80
4.6.2	Multi-channel Heuristic Performance	85
4.6.2.1	Metaheuristic for Obtaining Comparison Solutions to Min- MaxEo	85

4.6.2.2	Performance of the Multi-channel Heuristic	86
4.7	How This Problem Solution is a Cognitive Network	91
5	Minimum Expected Cost Routing in Markovian MANETs	92
5.1	Related Work	94
5.2	Markovian Network Processes	95
5.3	Routing as a Markov Decision Problem	96
5.3.1	Selecting an Appropriate MDP Type	98
5.3.2	Characteristics of an Optimal Policy	99
5.3.3	Finding an Optimal Policy	100
5.3.4	The Suboptimality of Shortest-Path Routing in Markovian Networks	101
5.4	Relating Distance and Link State Information	102
5.4.1	Link Distance in SITI Networks	103
5.4.2	Link Distance in SITD Networks	104
5.4.2.1	Linear Networks	104
	Static Links	105
	Bounds on $v^{\pi^*}(s)$	106
5.4.2.2	Arbitrary Networks	108
5.4.3	Delayed State Information	109
5.5	The DEC-MDP Formulation	110
5.6	The Markov Routing DEC-MDP as a Cognitive Network	111
6	Conclusions, Contributions, and Future Work	113
6.1	Conclusions	113
6.2	Contributions of this Dissertation	115
6.2.1	Foundations	115
6.2.2	The DSA Application	115
6.2.3	The MANET Routing Application	116
6.3	Future Work	117

6.3.1	Directions for MinMaxEo Topology Control	117
6.3.2	MANET Routing Future Work	118
A	Proof of Lemma 4.5.1	120
B	Proof of Theorem 5.4.2	123
C	Proof of Theorem 5.4.3	129
D	Proof of Theorem 5.4.4	131
E	Acronyms	134
	Bibliography	145

List of Figures

2.1	The Cognitive Framework and Critical Design Issues ©2006 IEEE. Reprinted, with permission, from Thomas <i>et al.</i> , “Cognitive networks: Adaptation and learning to achieve end-to-end performance objectives,” <i>IEEE Communications Magazine</i> , vol. 44, pp. 51–57, Dec. 2006.	12
3.1	Cognitive Node Architecture ©2008 IEEE. Reprinted, with permission, from Friend <i>et al.</i> , “Architecture and performance of an island genetic algorithm-based cognitive network,” in <i>IEEE CCNC</i> , (Las Vegas, Nevada, USA), January 2008.	42
4.1	Histograms of $\hat{E}_o(M, H)/E_o(M, H)$ ©2008 IEEE. Reprinted, with permission, from Friend <i>et al.</i> , “Environmentally-friendly secondary network topology control for minimizing outage potential,” in <i>IEEE DySPAN</i> , (Chicago, IL), pp. 178–185, October 2008.	62
4.2	Performance of MinMaxP Heuristic	77
4.3	Performance of Randomly Selected Solution	78
4.4	Performance Comparison for 10 Node Networks ©2008 IEEE. Reprinted, with permission, from Friend <i>et al.</i> , “Environmentally-friendly secondary network topology control for minimizing outage potential,” in <i>IEEE DySPAN</i> , (Chicago, IL), pp. 178–185, October 2008.	81
4.5	Histograms of the Ratio of Heuristic Performance vs. Min-total Power Topology Control ©2008 IEEE. Reprinted, with permission, from Friend <i>et al.</i> , “Environmentally-friendly secondary network topology control for minimizing outage potential,” in <i>IEEE DySPAN</i> , (Chicago, IL), pp. 178–185, October 2008.	82
4.6	Histograms of the Ratio of Heuristic Performance vs. LIFE ©2008 IEEE. Reprinted, with permission, from Friend <i>et al.</i> , “Environmentally-friendly secondary network topology control for minimizing outage potential,” in <i>IEEE DySPAN</i> , (Chicago, IL), pp. 178–185, October 2008.	83

4.7	Performance of the MinMaxEo Heuristic	88
4.8	Performance of the MinMaxP Heuristic on MinMaxEo	89
4.9	Performance of Random Solutions on MinMaxEo	90
5.1	Example Markovian Network	102
5.2	Linear Markovian Network	105

Chapter 1

Introduction

Riding the back of the digital revolution, the ever-increasing capabilities of wireless devices have placed upon us a tremendous challenge – how to put all of this capability to effective use. The complexity of modern wireless networks far exceeds the ability of users and operators to optimally configure them, with the only viable solution being to incorporate greater degrees of intelligence within the network itself. Such a network would perform autonomously (or at least require little human intervention) and be equipped to solve difficult problems and to learn in the face of uncertainty.

A potential solution to these challenges is the emerging field of cognitive networks, which was originally introduced by Thomas *et al.* [1]. Cognitive networks are formed when a collection of communication nodes organize to achieve network-level goals with the aid of some form of cognitive processing. The concept is an outgrowth of research in cognitive radios, which couples intelligent, or cognitive, processing with a platform that provides the ability to both sense the radio environment and react by changing its configuration. The question that Thomas *et al.* sought to answer was how cognitive processing could be applied to problems at the network level.

As with any emerging research topic, much of the initial work must be performed at a high

level, answering such questions as “What is a cognitive network?” and “What methods may be used to design a cognitive network?”. Chapters 2 and 3 address these questions, with the focus of Chapter 2 being to establish some key characteristics of a cognitive network through comparison against cognitive radio and cross-layer design. As fields of research are usually built on prior, and perhaps concurrent, research topics, Chapter 2 describes similarities with these other areas as well as differences. In Chapter 3, we explore cognitive models and how cognitive models translate to computational methods. We identify learning and reasoning as key characteristics of cognitive networks and describe methods for implementing each of these functions. As cognition is most naturally associated with AI, we discuss the relationship between multiagent systems, a key AI concept, and cognitive networks.

Once a concept has been adequately defined, one’s attention turns to applications. Often, this emphasis on applications enables further refinement of the initial concept. The broad association of cognitive radio research with DSA motivated us to consider network-level DSA problems for application of a cognitive network. DSA seeks to use the radio spectrum more efficiently by taking advantage of variations in the spatio-temporal usage of spectrum owners, or primary users. A key enabler of DSA is the ability of detect primary users in a frequency band. The DSA and cognitive radio communities have been justifiably preoccupied with the detection and classification of primary user signals. However, limitations inherent in the detection of random processes mean that it is impossible to always detect primary users. Also, the potentially time-varying behavior of primary users, coupled with the non-zero amount of time that it takes to vacate channels, creates additional potential for interfering with primary user communications. Additive interference from secondary nodes (i.e. nodes that seek to use spectrum when primary users are inactive) is a network-level issue that is affected by network topology as well as the medium access control (MAC) protocol in use. Network topology has traditionally been defined by node transmit powers, but the advent of multi-channel radios has led to the formation of multichannel topology control problems. Multichannel topology control is a recent development, with few results currently in existence. Furthermore, topology control problems usually do not take into consideration

the effects of the MAC protocol in the topology control objective. In Chapter 4, we formulate a multichannel topology control problem that includes the effects of a CSMA-like MAC protocol as well as the stochastic nature of the wireless medium to minimize the potential for interfering with primary users. The secondary network uses past observations of primary user behavior to compute a spatial map representing the potential for primary users to appear in an area that included the secondary network. This learning of primary user behavior patterns, coupled with heuristic reasoning to minimize an interference objective, is an example of a cognitive network applied to a DSA problem.

While DSA has tremendous potential and has captured the interest of many researchers, emphasis on a particular application may lead to confusion of the technology with the application of that technology. For example, cognitive radio is often used synonymously with DSA, despite the fact that Mitola, the inventor of the cognitive radio concept, identified radio resource management, network management, services delivery, and download certification as potential application areas [2], signifying that the cognitive radio concept is more broad than just DSA. By focusing on a single application, we may leave the impression that the concept does not extend to other types of problems. To address this concern, Chapter 5 addresses the problem of routing in MANETs as another suitable application for a cognitive network. This problem is quite distinct from DSA, which emphasizes the versatility of the cognitive network concept. Routing in MANETs has proven to be a challenging problem, and despite the existence of literally hundreds of routing protocols, there is little work that can claim optimality of a proposed solution. Chapter 5 models a MANET as a Markovian random graph, for which the optimal routing policy is the solution to a Markov decision problem (MDP). However, the optimal policy cannot be executed by real networks due to physical limitations on the availability of link state information. This leads us to characterize the realistic routing problem as a decentralized decentralized partially observable Markov decision problem (DEC-MDP), which is also of significant interest in current MAS research. By treating each node in the network as an agent, the problem described in Chapter 5 fits perfectly with the description of a cognitive network as a MAS presented in Chapter 3.

Chapter 6 closes the dissertation by providing a summary of the conclusions that we have reached as a result of the work in preceding chapters. We also provide a summary of the main contributions of this work to the existing body of knowledge and describe opportunities for future research in cognitive networks as well as on the particular problems covered in Chapters 4 and 5.

The main contributions of this dissertation are to the development of the cognitive network concept and to the solution of the two applications that we study. In the cognitive network arena, we contribute to the concept definition and propose ways in which it may be tied to AI as well as methods for implementing the key elements of cognition, learning and reasoning. For the multichannel topology control application, we present polynomial-time heuristics that address our objective function and a polynomial-time heuristic that addresses a previously unstudied power-based multichannel topology control problem. For the MANET routing application, we provide one of the few examples of provably optimal routing policy and give analytical results for several special classes of Markovian random networks.

Chapter 2

Background on Cognitive Networks ¹

Cognitive networks are motivated by complexity. Particularly in wireless networks, there has been a trend towards increasingly complex, heterogeneous, and dynamic environments. While wired networks can also take on any of these characteristics and are not excluded from potential cognitive network applications, wireless networks are a natural focus of research in complex networks because of the significant additional complexity introduced by the wireless medium. Previous wireless research in cognitive radio and cross-layer design has addressed some of these issues, but has shortcomings from the network perspective. Cognitive networks represent a new approach to dealing with this complexity. This chapter is intended to provide a primer on the cognitive network concept. As such, it begins by first explaining the need for cognitive networks, how they are defined, and possible applications for the technology. Then we examine how cognitive networks are related to, but distinct from, previous work in cognitive radios and cross-layer design.

¹This chapter is based on the work published in [3] and is a result of joint work with Dr. Ryan Thomas.

2.1 Definition

We suggest the following definition for a cognitive network, first described by Thomas *et al.* in [1]: A cognitive network is a network with a cognitive process that can perceive current network conditions, and then plan, decide, and act on those conditions. The network can learn from these adaptations and use them to make future decisions, all while taking into account end-to-end goals.

The cognitive aspect of this definition is similar to that used to describe cognitive radio and broadly encompasses many simple models of cognition and learning. More critical to the definition are the network and end-to-end aspects. Without the network and end-to-end scope, the system is perhaps a cognitive radio or layer, but not a cognitive network. Here, end-to-end denotes all the network elements involved in the transmission of a data flow. For a unicast transmission, this might include the subnets, routers, switches, virtual connections, encryption schemes, mediums, interfaces, or waveforms, to mention just a few. The end-to-end goals are what give a cognitive network its network-wide scope, separating it from other adaptive approaches.

2.2 Motivation and Requirements

The overall goal of any technology is that it meet some need in the best way possible for the least cost. With the first half of this goal in mind, a cognitive network should provide, over an extended period of time, better end-to-end performance than a non-cognitive network. Cognition can be used to improve the performance of resource management, quality of service (QoS), security, access control, or many other network goals. Cognitive networks are only limited in application by the adaptability of the underlying network elements and the flexibility of the cognitive process.

In examining the second half of this goal, the cost (in terms of overhead, architecture, and

operation) must justify the performance. We would expect, in general, that implementing a cognitive network would require a system that is more complex than a non-cognitive network. Thus for cognitive networks to be justifiable, the performance improvement must outweigh these additional costs. For certain environments, such as static wired networks with predictable behavior, it may not make sense to convert to cognitive behaviors. Other environments, such as heterogeneous wireless networks, may be ideal candidates for cognition.

Cognitive networks should use observations of network performance as input to a decision-making process and then provide output in the form of a set of actions that can be implemented in the modifiable elements of the networks. Ideally, a cognitive network should be forward-looking, rather than reactive, and attempt to adjust to problems before they occur.

Cognitive networks require a software adaptable network (SAN) to implement the actual network functionality and allow the cognitive process to adapt the network. Similarly to cognitive radio, which depends on a software defined radio (SDR) to modify aspects of radio operation, a SAN depends on a network that has one or more modifiable elements. Practically, this means that a network must be able to modify one or several layers of the network stack in its member nodes. A simple example of a SAN could be a wireless network with directional antennas (antennas with the ability to direct their maximum receive or transmit gain to various points of rotation). A more complex example would incorporate more modifiable aspects at various layers of the protocol stack, such as MAC adaptations or routing control.

2.3 Foundations and Related Work

Having defined a cognitive network, it is helpful to review some existing research areas related to the cognitive network concept. We take a look at two areas in particular, cognitive radio and cross-layer design. We also examine other current research on the cognitive network

concept.

2.3.1 Cognitive Radio

Shared Attributes with Cognitive Networks The 50 percent correlation in nomenclature would of itself imply some degree of commonality, and it can certainly be argued that research in cognitive radio has sparked the formulation of the cognitive network concept. What cognitive radios and cognitive networks do share is the cognitive process that is the heart of the performance optimizations. An essential part of the cognitive process is the capability to learn from past decisions and use this learning to influence future behavior. Both are goal-driven and rely on observations paired with knowledge of node capabilities to decide on current actions and plan for the future. A cognitive radio requires tunable parameters which define the optimization space of the cognitive process. These tunable parameters are ideally provided by an SDR. The concept of the SAN is the cognitive network analog of SDR. Therefore, both technologies employ a software-configurable platform that is controlled by the cognitive process.

Differences from Cognitive Networks Cognitive networks are delineated from cognitive radios by the scope of the controlling goals. Goals in a cognitive network are based on end-to-end network performance, whereas cognitive radio goals are localized only to the radios user. These end-to-end goals may be derived from users, applications, and/or resource requirements. This difference in goal scope from local to end-to-end enables the cognitive network to operate more easily across all layers of the protocol stack. Current research in cognitive radio emphasizes interactions with the physical layer, which limits the direct impact of changes made by the cognitive process to the radio itself. Agreement with other radios on parameters that must match for successful link communication is reached through a process of negotiation. Since changes in protocol layers above the physical layer tend to impact more nodes in the network, the cognitive radio negotiation process would have to be

expanded to include all nodes impacted by the change. However, because the negotiation process is unable to assign precedence to radios desires without goals of a broader scope, achieving agreement among multiple nodes may be a slow process. Any resulting compromise may result in suboptimal network performance. In contrast, cognitive networks are more cooperative in nature, since the performance is referenced to the end-to-end goals and nodes within a single cognitive element must cooperate to enact decisions.

Another significant difference between cognitive radios and cognitive networks is the degree of heterogeneity that is supported. Cognitive networks are applicable to both wired and wireless networks whereas cognitive radios are inherently for wireless use. Since the cognitive network may span multiple wired and wireless mediums, it is useful for optimizing performance for these heterogeneous types of networks, which are generally difficult to integrate.

The fact that a cognitive network is composed of multiple nodes also adds a degree of freedom in how the cognitive processing is performed compared to cognitive radio. A cognitive network has the option to implement a centralized cognitive process, a fully distributed cognitive process, or a partially distributed cognitive process.

2.3.2 Cross-Layer Design

Shared Attributes with Cross-Layer Design Designs that violate the traditional layered approach by direct communication between non-adjacent layers or sharing of internal information between layers are called cross-layer designs [4]. Cognitive networks may directly or indirectly share information that is not available externally in the strict layered architecture. Therefore, cognitive networks may perform cross-layer design.

The common theme is that observations are made available externally and some adaptation is performed at a layer other than the layer providing the observation. In a cognitive network that included multiple layers in its SAN, protocol layers provide observations of current conditions to the cognitive process. The cognitive process then determines what is optimal

for the network and changes the configurations of network elements' protocol stacks.

Differences from Cross-Layer Design Despite similarities, cognitive networks reach beyond the scope of cross-layer designs. The cognitive network may support trade-offs between multiple goals and in order to do so may perform multi-objective optimization (MOO), whereas cross-layer designs typically perform single objective optimizations. Cross-layer designs perform independent optimizations that do not account for the network-wide performance goals. Trying to achieve each goal independently is likely to be suboptimal, and as the number of cross-layer designs within a node grows, conflicts between the independent adaptations may lead to adaptation loops [5]. This pitfall is avoided in a cognitive network by jointly considering all goals in the optimization process.

The ability to learn is another significant difference. The cognitive network learns from prior decisions and applies the learning to future decisions. Cross-layer designs are adaptations that respond the same way when presented with the same set of inputs, regardless of how poorly the adaptation may have performed in the past. The ability to learn from past behavior is particularly important in light of the fact that our understanding of the interaction between layers is limited.

As was the case when compared to cognitive radio, the scope of goals and observations sets cognitive networks apart from cross-layer design. The observations used by the cognitive process span multiple nodes, and the optimization is performed with a network-level objective in mind, whereas cross-layer design is node-centric. This global information allows the cognitive process to adapt in ways that are not possible when nodes have limited visibility into the state of other nodes or parts of the network, as is the case with cross-layer design.

2.3.3 Cognitive Networks in the Literature

Perhaps the first mention of a cognitive network-like concept was in the article by Clark *et al.* on the network knowledge plane, which they define as a “pervasive system within the network that builds and maintains high level models of what the network is supposed to do” [6]. Thomas *et al.* [1] were the first to define the term “cognitive network” and examine its functionality. More recently, the End-to-End Reconfigurability Project II [7], m@ANGEL platform [8], CTVR at Trinity College [9], and the Institute for Wireless Networks at RWTH Aachen University [10] have proposed architectures at various degrees of maturity for end-to-end oriented, autonomous networks. These architectures are typically focused on a particular application (such as 4G cellular or wireless), implementation (such as the cognitive mechanism or associated API), or problem (such as mobility or management).

2.4 Generic Cognitive Network Hierarchy

A common model of cognition is the three-level theory [11]. The model is often summarized as consisting of behavioral, functional, and physical layers. The behavioral level determines what observable actions the system produces, the functional layer determines how the system processes the information provided to it, and the physical layer comprises the neuro-physiology of the system.

From this concept, we derive a three-layer framework, with each layer corresponding roughly to the layers of the three-level theory. At the top layer are the goals of the system and elements in the network that define the behavior of the system. These goals feed into the cognitive process, which determines the actions that the system takes. The SAN is the physical control of the system, providing the action space for the cognitive process. This framework is illustrated in Fig. 2.1.

The framework allows a cognitive process to consist of one or more cognitive elements,

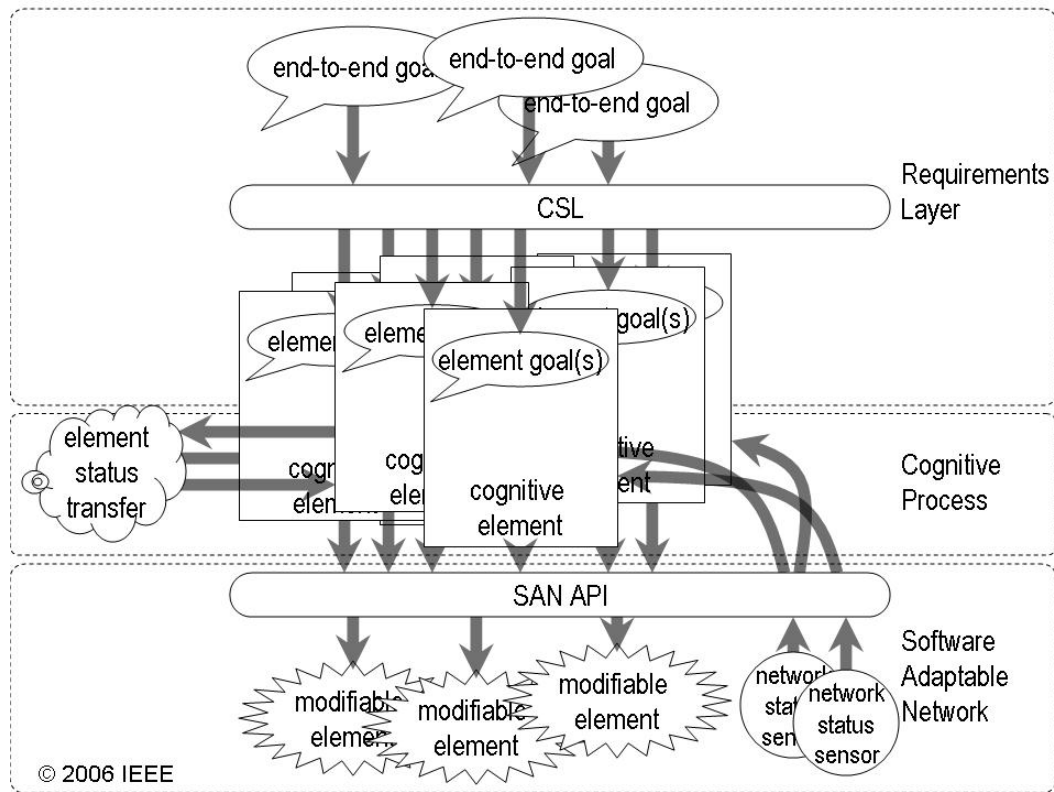


Figure 2.1: The Cognitive Framework and Critical Design Issues ©2006 IEEE. Reprinted, with permission, from Thomas *et al.*, “Cognitive networks: Adaptation and learning to achieve end-to-end performance objectives,” *IEEE Communications Magazine*, vol. 44, pp. 51–57, Dec. 2006.

operating in some degree between autonomy and full cooperation. If there is a single cognitive element, it may still be physically distributed over one or more nodes in the network. If there are multiple elements, they may be distributed over a subset of the nodes in the network, on every node in the network, or several cognitive elements may reside on a single node. In this respect, the cognitive elements operate in a manner similar to a software agent.

2.4.1 User/Application/Resource Requirements

The top-level component of the cognitive network framework includes the end-to-end goals, cognitive specification language (CSL), and cognitive element goals. The end-to-end goals,

which drive the behavior of the entire system, are put forth by the network developers, users, applications and/or resources. Without end-to-end goals guiding network behavior, undesired consequences may arise. This is a problem with many cross-layer designs and is explored in some depth in [5], which illustrates unintended end-to-end interactions in a MAC/PHY cross-layer design.

Like most engineering problems, there is likely to be a trade-off for every goal that is part of the optimization. When a cognitive network has multiple objectives, it may not be able to optimize all metrics indefinitely, eventually reaching a point in which one metric cannot be optimized without affecting another. This set of actions from which no goal can be improved without worsening another is called the Pareto optimal front.

To connect the goals of the top-level users of the network to the cognitive process, an interface layer must be developed. In a cognitive network, this role is performed by the CSL, which provides behavioral guidance to the elements by translating the end-end goals to local element goals. Less like the Radio Knowledge Representation Language (RKRL) proposed by Mitola for cognitive radio and more like a QoS specification language [11], the CSL maps end-to-end requirements to underlying mechanisms. Unlike a QoS specification language, the mechanisms are adaptive to the network capabilities as opposed to fixed. Furthermore, a CSL must be able to adapt to new network elements, applications, and goals, some of which may not even be imagined yet. Other requirements may include support for distributed or centralized operation, including the sharing of data between multiple cognitive elements.

2.4.2 The Cognitive Process

There is not a common, accepted definition of what cognition means when applied to communication technologies. The term cognitive, as used in this chapter, follows closely in the footsteps of the definition used by Mitola in [2] and the even broader definition of the FCC. The former incorporates a spectrum of cognitive behaviors, from goal-based decisions to proactive adaptation. Here, we associate cognition with machine learning, which is broadly

defined in [12] as any algorithm that “improves its performance through experience gained over a period of time without complete information about the environment in which it operates.” Underneath this definition, many different kinds of artificial intelligence, decision making, and adaptive algorithms can be placed, giving cognitive networks a wide scope of possible mechanisms to use for reasoning and learning.

Learning serves to complement the decision-making of the cognitive process by retaining the effectiveness of past decisions under a given set of conditions and/or by revealing behavioral patterns in the network environment that aid in planning and future decision-making. Gauging the effectiveness of past decisions requires a feedback loop to measure the success of the chosen solution in meeting the objectives defined. This is retained in memory, so that when similar circumstances are encountered in the future, the cognitive process will have some idea of where to start or what to avoid. Learning behavioral patterns may be accomplished by using observations to update a deterministic or stochastic model of the environment. This model is used as an input to the reasoning process, so that as the accuracy of the model improves as a result of learning, reasoning performance also improves.

The effect of a cognitive process’s decisions on the network performance depends on the amount of network state information available to it. In order for a cognitive network to make a decision based on end-to-end goals, the cognitive elements must have some knowledge of the network’s current state or the state of the network environment. If a cognitive network has knowledge of the entire state, decisions at the cognitive element level should be at least as good, if not better (in terms of the cognitive element goals) than those made in ignorance. For a large, complex system such as a computer network, it is unlikely that the cognitive network would know the total system state. There is often a high cost to exchange this information, and it may even be impossible to know the instantaneous network state due to an inability to communicate state information before it changes. This implies that a cognitive network will have to work with partial state information.

The performance costs of a distributed system knowing less than the whole state of the

system could be termed the *price of ignorance*. Of course, the price of ignorance would only account for costs due to decisions made on insufficient information. Other issues that may arise in systems with limited information, such as the reduction in overhead from reduced data collection and distribution or the increase in critical nodes knowledge of important information represent different design issues in the engineering trade-space.

2.4.3 Software Adaptable Network

The SAN consists of the application programming interface (API), modifiable network elements, and network status sensors. The SAN is really a separate research area, just as the design of the SDR is separate from the development of the cognitive radio, but at a minimum the cognitive process needs to be aware of the API and the interface it presents to the modifiable elements. Just like the other aspects of the framework, the API should be flexible and extensible.

Another responsibility of the SAN is to provide observations of the network state or surrounding environment to the cognitive process. These observations are the source of the feedback used by the cognitive process. Possible observations may be local (such as bit error rate, battery life, or data rate), nonlocal (such as end-to-end delay and clique size), external (such as spectrum scanning), or compilations of different observations.

Overall, the SAN enables the higher-level cognitive processing that occurs in cognitive elements. The control of modifiable elements in the SAN, as well as observations made by the SAN, are used in reasoning and learning. As reasoning and learning are the key elements in cognitive processing, we devote the next chapter to a discussion of these elements.

Chapter 3

Learning and Reasoning: Key Elements of Cognitive Networks ¹

The emerging research area of cognitive networks offers a potential solution for dealing with the increasing complexity of communication networks by empowering networks with learning and decision-making capabilities. A key feature of a cognitive network, as described in Chapter 2, is the cognitive process, which is responsible for the learning and reasoning that occurs in the cognitive network. This chapter investigates the underlying mechanisms for the cognitive process and the trade-offs involved in selecting and implementing these mechanisms. We restrict our focus to selected methods that appear to be applicable to cognitive networks and do not attempt to evaluate all possible methods for reasoning and learning.

The terms learning and reasoning are difficult to define for general use, making it important to explain what these terms mean to us. Following the line of thought in [15], we consider reasoning to be the immediate decision process that must be made using available historical knowledge as well as knowledge about the current state of the world. The primary respon-

¹The majority of this chapter is based on the sections of [13] that were written by me. Section 3.4, which is based on [14], is the result of joint work with Mustafa ElNainay and Yongsheng (Sam) Shi.

sibility of reasoning is to choose a set of actions. Learning is a long-term process consisting of the accumulation of knowledge based on the perceived results of past actions. Cognitive nodes learn by enriching the knowledge base to improve the efficacy of future reasoning. It is not always easy to separate reasoning and learning along these lines since the two may be tightly coupled. We will endeavor to point out cases in which they cannot be easily separated as they are encountered.

Having established our position on the nature of learning and reasoning in a cognitive network, we proceed to discuss frameworks for learning and reasoning in Section 3.1, where we also give our reasons for viewing the multi-agent system (MAS) as a fitting framework for bridging the gap between AI and cognitive networks. Section 3.2 goes into detail on some of the methods that may be used for learning and reasoning in a cognitive network. We then turn our attention in Section 3.3 to the sensor and actuator functions that close the cognitive loop and give our perspective on how existing research in sensor networks may be used to benefit cognitive networks. Section 3.4 concludes the chapter with a presentation of a cognitive node architecture that facilitates the elements of cognitive processing described in this chapter and in Chapter 2.

3.1 Frameworks for Learning and Reasoning

As a starting point for investigating systems that are capable of learning and reasoning, it is helpful to consider general cognitive architectures. More specifically, we are interested in cognitive architectures that are intended for implementation in a computing device. A variety of such architectures exist with varying degrees of complexity. At the higher complexity end of the scale are the so-called universal theories of cognition such as ACT-R [16], Soar [17], and ICARUS [18]. Each of these architectures seeks to capture all of the components of the human mind and the interactions necessary for cognition. Toward the other end of the complexity scale are the OODA and CECA loops [19]. These two architectures do not

attempt to incorporate all of the elements of human thinking but are intended to model the decision-making process to provide a basis for improved command and control in military decision-making. A cognitive architecture that is closely related to the OODA loop is the cognition cycle proposed in [2], which is the basis for many of the architectures used in cognitive radio research.

3.1.1 Linking Cognitive Architectures to Cognitive Network Architectures

While it is tempting to convert the most human-like cognitive architectures to cognitive network architectures, we must not forget that the purpose of the cognitive network is to exchange data between users and applications within the network. By directly converting human cognitive models for use in cognitive networking, we may introduce unnecessary complexity. Therefore, it may be appropriate to simplify a human-like cognitive architecture to remove the elements that are superfluous to the purposes of networking. On the other hand, some cognitive architectures may be oversimplified and lack key elements for a successful cognitive network architecture. An example of this may be the OODA loop, which in its simplest incarnation [19] does not include learning. It is conceivable for a cognitive network to be designed without the capacity to learn; however, cognitive networks are most likely to be applied to complex problems that are difficult or infeasible to completely characterize at design time. Therefore, we expect learning to play a central role in a cognitive network.

More recently, there have been proposals for cognitive network architectures in [9, 10]. The architectures presented in [9, 10] are based on the cognition cycle of Mitola [2] and focus on the inner workings of a cognitive node. In [10], Mahonen *et al.* propose that a toolbox of methods from machine learning, mathematics, and signal processing be available for matching to the needs of the decision-making process. In [9], Sutton *et al.* describe a reconfigurable node architecture for use in cognitive networks. The emphasis is on enabling reconfiguration at all layers of the protocol stack, with reconfiguration varying from changing parameters within

a layer to replacement of the entire layer with a different protocol.

3.1.2 The Cognitive Network as a Multiagent System

Discussion of cognition, learning, and reasoning in a computational environment inevitably leads to a discussion of artificial intelligence. One of the largest subfields within AI is the multi-agent system (MAS). Although a MAS has no universally accepted definition, we will adopt the definition in [20], which relies on three concepts: situated, autonomous, and flexible. *Situated* means that agents are capable of sensing and acting upon their environment. The agent is generally assumed to have incomplete knowledge, partial control of the environment, or both limitations [21]. *Autonomous* means that agents have the freedom to act independently of humans or other agents though there may be some constraints on the degree of autonomy each agent has. *Flexible* means that agents' responses to environmental changes are timely and pro-active and that agents interact with each other and possibly humans as well in order to solve problems and assist other agents. A special case of a MAS is the study of cooperative distributed problem solving (CDPS), which differs from a MAS in that, in addition to the three concepts describing a MAS, CDPS also possesses inherent group coherence [22]. In other words, motivation for the agents to work together is inherent in the system design. This simplifies the problem but leaves the challenging issue of how to ensure that this benevolence is put to good use.

In [15], Dietterich describes a standard agent model consisting of four primary components: observations, actions, an inference engine, and a knowledge base. In this agent model, reasoning and learning are a result of the combined operation of the inference engine and the knowledge base. By our definition, reasoning is the immediate process by which the inference engine gathers relevant information from the knowledge base and sensory inputs (observations) and decides on a set of actions. Learning is the longer term process by which the inference engine evaluates relationships, such as between past actions and current observations or between different concurrent observations, and converts this to knowledge

to be stored in the knowledge base. This model fits well within most of the cognitive architectures previously mentioned, and we shall use it as our standard reference for our discussion of learning and reasoning.

Returning to the cognitive network architecture in Figure 2.1, we will relate this architecture to the agent model just described. Sensory inputs (network status sensors) in the cognitive network architecture are received from nodes in the network through the SAN API. Actions are taken on the modifiable elements using the SAN API as well. Inference engines and knowledge bases are contained within the cognitive elements. In distributed implementations, each cognitive element consists of multiple agents (i.e. network nodes) that learn and reason cooperatively to achieve the common element goals. Interactions between groups of agents belonging to different cognitive elements can be opportunistic, allowing competition between cognitive elements.

At this point it is clear that a cognitive network can be framed as a MAS. This is perhaps no surprise due to the generality of the MAS definition, but we will continue a step further and explain why a cognitive network is more aligned to the CDPS framework when agents share the same set of end-to-end goals. A key component of the definition of a cognitive network is the existence of end-to-end goals that drive its behavior. These goals provide a common purpose for each agent and motivate them to work together. Referring to Figure 2.1, the greatest alignment occurs among agents within the same cognitive element since they have the same element goals. Across cognitive elements, the element goals are derived from the network's end-to-end goals. This implies that there is some degree of coherence across cognitive elements.

3.2 Learning and Reasoning within a MAS Framework

Having established the framework within which we will treat learning and reasoning for a cognitive network, we turn our attention to methods that allow agents to learn and reason

and the factors involved in selecting a method. The number of methods that we could discuss is much larger than this chapter can accommodate, so we have simply tried to provide a cross-section of possible methods that are suitable for use in cognitive networks. We have divided the reasoning methods into two categories: generic methods and heuristics. The motivation for doing so is rooted in computational complexity and the trade-off between general versus limited-scope methods.

3.2.1 Complexity and Exploitation of Domain Knowledge

Many existing network optimization problems have been shown to be NP-hard. In fact, networking research seems to attract much attention from theoretical computer scientists due to the rich variety of computational complexity issues that arise from inherently combinatorial problems. Since cognitive networks are intended to address the inherent complexity in networking problems, we would expect the problems addressed by cognitive networks to frequently be NP-hard. Since such problems are unlikely to admit polynomial time optimal solutions, researchers must devise non-optimal methods. Frequently, a heuristic algorithm can be found that provides reasonable average performance and, in some cases, algorithms can be proven to provide solutions that are within some factor of the optimal (these are known as approximation algorithms). Another approach is to use a more general problem-solving method such as a metaheuristic. These methods are applicable to a wider variety of problems; therefore, we refer to them as generic methods.

In the context of a cognitive network, the flexibility of a generic reasoning method is more aligned with the concept of human cognition in that a single “brain” is able to solve problems of various types, whereas a heuristic is more limited in the range of problems that it may address. In fact, a heuristic may become worthless with even a slight change in the problem definition. This is because heuristics tend to exploit domain-specific knowledge (i.e. the specifics of the problem) to a greater degree than generic reasoning methods. This exploitation may provide significant improvements in performance, but at the cost of flexibility. The

decision between the degree of flexibility needed for the reasoning method is driven by the intended application. However, heuristics may be able to provide some flexibility to the cognitive network if they form a toolbox of reasoning methods, with the appropriate heuristic being selected at runtime, as described in [10].

3.2.2 Elements of Reasoning

The primary objective of reasoning within a cognitive network is to select an appropriate set of actions in response to perceived network conditions. This selection process ideally incorporates historical knowledge (often referred to as short term and long term memories) as well as current observations of the network's state.

Often reasoning is categorized as either inductive or deductive. Inductive reasoning forms hypotheses that seem likely based on detected patterns whereas deductive reasoning forgoes hypotheses and only draws conclusions based on logical connections. Due to the size of the cognitive network state space, which is likely to grow combinatorially with the number of network nodes, the cognitive process must be capable of working with partial state information. Since the cognitive process always sees a limited view of the network state, it is difficult to draw certain conclusions as required by deductive reasoning. The approach in inductive reasoning of generating a best hypothesis based on what is known to be more conducive to the limited observations available to the cognitive process.

Reasoning (or decision-making) can also be categorized as one-shot or sequential [15]. A one-shot decision is akin to single-shot detection in a digital communications receiver. A final action is selected based on immediately available information. Conversely, sequential reasoning chooses intermediate actions and observes the response of the system following each action. Each intermediate action narrows the solution space until a final action can be chosen. This is a natural approach to problem diagnosis. Sequential reasoning may also be especially useful for proactive reasoning where time constraints are more relaxed and there are only indications of an impending problem. However when immediate action is needed,

such as a response to congestion at a particular network node, a one-shot decision is more expedient.

The formulation of our approach to reasoning in a cognitive network is partly driven by the depth of our understanding of relationships between the parameters we can control and the observations that we have available to us. Treating the network as a system, our actions are the inputs to the system and the observations are the outputs. If we cannot mathematically represent relationships between the inputs and outputs, then we may select a reasoning method that is capable of dealing with uncertainty, such as Bayesian networks. This approach relies heavily on the learning method to uncover the relationships between inputs and outputs, implying the need for a large number of observations to reveal statistical relationships. If we can link inputs and outputs mathematically, then we can use methods based on the optimization of objective functions such as distributed constraint reasoning or metaheuristics. In this case, learning may be used to reduce the time required to find solutions that are within an acceptable range of the optimum.

3.2.3 Generic Methods for Reasoning

3.2.3.1 Distributed Constraint Reasoning

Distributed constraint reasoning problems can be further classified as either a distributed constraint satisfaction problem (DisCSP) or a distributed constraint optimization problem (DCOP). A DisCSP attempts to find any of a set of solutions that meets a set of constraints, whereas a DCOP attempts to find an optimal solution to a set of cost functions. The cost functions in a DCOP essentially replace the constraints in a DisCSP, which allows for a more general problem solving method. Since a DCOP always admits a solution whereas a DisCSP may not be satisfiable, we focus our attention on the application of a DCOP to cognitive networks.

Following the definition given in [23], each agent in a MAS DCOP has control over a variable

x_n . Each variable has a finite and discrete domain D_n . A set of cost functions $f_{i,j}$ are defined over pairs of variables x_i and x_j . The sum of the cost functions provides an objective function that the DCOP seeks to minimize.

A well-known algorithm for solving a DCOP is Adopt [23]. The goals of Adopt are to limit the amount of inter-agent communication, allow flexibility in the accuracy of the solution to speed execution time, provide theoretical bounds on the worst case performance, and allow completely asynchronous operation. Adopt first forms a tree structure based on the cost functions. This tree structure is used to determine which agents communicate and what messages will be transmitted between agents. Agents then iteratively choose solutions for their variables based on local knowledge and exchange these partial solutions in the search for a global solution. Upper and lower bounds on the cost functions are computed and continually refined by the agents as new information is received. Tracking the lower and upper bounds allows early termination of the algorithm when some amount of error in the solution is tolerable.

The desirable characteristics for using Adopt in a cognitive network are asynchronous message passing, autonomous control of each agent's variable, and the flexibility of choosing solution accuracy. Additional advantages of Adopt are its potential for robustness to message loss and its ability to reduce inter-agent communications through a timeout mechanism [24]. These are especially important for wireless cognitive networks. Two limitations of DCOP algorithms are that cost functions are functions of only two variables and that each agent is assigned only one variable. However, procedures for extension to n-ary cost functions and multiple variables per agent are presented in [23] (without evaluation) with further work on n-ary cost functions in [25].

3.2.3.2 Bayesian Networks

The Bayesian network (BN) is a method of reasoning under uncertainty. The uncertainty may be a result of limited observations, noisy observations, unobservable states, or uncertain

relationships between inputs, states, and outputs within a system [26]. All of these causes for uncertainty are common in communications networks. In particular, the ability of cognitive networks to potentially control parameters at different layers in the protocol stack gives rise to concern over interactions between different protocol layers, interactions that are currently not well understood [27]. BNs provide a means for dealing with this uncertainty probabilistically.

BNs decompose a joint probability distribution (JPD) over a set of variables (i.e. events) into a set of conditional probability distributions defined on a directed acyclic graph (DAG). Each node represents a variable in the JPD. The directionality of the edges in the DAG represents parent-child relationships where, conditioned on the knowledge of its parents, a child is independent of all other non-descendents in the network [26]. Each node contains a conditional probability distribution, which is the probability distribution of the variable at that node conditioned on the variables at all parents of the node. The JPD can then be reconstructed as the product of all of the conditional probability distributions.

BNs are an example of a method that incorporates both reasoning and learning. Learning in a BN is accomplished through belief updating. Belief updating is the process of modifying the conditional probability distributions based on observations of variables in the JPD. Knowledge is contained within the conditional probability distributions as well as the DAG structure itself. Reasoning in a BN consists of producing probability estimates of sets of variables based on the current JPD and possibly observed variables as well. BNs only satisfy part of our definition of reasoning since they do not specify a method for selecting an action. To completely satisfy our definition of reasoning, BNs may be paired with a decision making method such as multi-criteria decision making (see [28] for an example).

A well known method for distributed BNs is the multiply sectioned Bayesian network (MSBN) [26]. The MSBN is constructed by sectioning the BN into subdomains that are then assigned to agents in an MAS. Each agent's subdomain is organized as a junction tree in order to simplify belief updating. The overall network structure is known as a linked junction forest. Organizing the original DAG as a linked junction forest provides guarantees

that cannot be made with arbitrary decompositions of the DAG.

There are two guarantees provided by the MSBN framework that are of particular interest to its application in a cognitive network. First, an agent is only allowed to directly communicate with other agents that share one or more of its internal variables. This ensures that communications are localized with respect to the variable dependencies. Second, the global JPD is consistent with the agents' conditional JPDs, which means that the MSBN is equivalent to a centralized BN.

Perhaps the first question that arises when considering an MSBN for application to a cognitive network is what the variables should be. There are two scenarios for the types of variables in the network. The variables may inherently be tied to each agent (in which case dynamic cognitive networks such as mobile wireless networks are more easily dealt with), or a subset of the variables may be independent of the number of agents in the MAS (in which case these variables must be protected from loss as agents enter and leave the cognitive network). It is likely that the dependence structure is unknown at design time and therefore must be learned (see Section 3.2.7.2 on learning the DAG structure of a BN). Also, a high average node degree in the DAG implies greater complexity in the computation of conditional distributions as well as increased communication between subdomains in the MSBN. A sparse DAG is ideal from both a computational as well as a communications standpoint.

3.2.3.3 Metaheuristics

Some of the most generic reasoning methods fall under the category of metaheuristics. A metaheuristic is an optimization method that teams simpler search mechanisms with a higher level strategy that guides the search. Metaheuristics commonly employ randomized algorithms as part of the search process [29]. This means that, given the same search space, the metaheuristic may arrive at a different solution each time it is run. Metaheuristics are commonly applied to problems for which finding an exact solution is infeasible. For these types of problems, the time required to find a globally optimum solution grows exponentially in

the dimension of the search space. Metaheuristics trade off solution optimality for a feasible computation time. Since metaheuristics are adept at handling complex problems of various types, they are of interest for cognitive networks.

Parallel Genetic Algorithms Parallel genetic algorithms are members of a class of biologically-inspired metaheuristics called evolutionary algorithms (EA). This class also includes evolutionary programming and evolution strategies. Because all three types of algorithms share a similar approach, we will focus only on the parallel genetic algorithm (PGA). The PGA evolves a population of candidate solutions through crossover and mutation operations. Parameters of the search space are encoded in chromosomes that are represented by binary vectors. The objective function is used to evaluate candidate solutions (i.e. chromosomes) and produce a fitness for each candidate solution. The fitness values are used to decide which members of the population survive and/or are crossbred. Randomness in the crossover and mutation operations seeks to explore the search space and provide diversity within the population.

Parallel implementations generally fall into three categories: master-slave, island (also distributed or coarse-grain), and cellular (or fine-grain) [29]. The PGA is centrally controlled in the master-slave model while the island and cellular models give agents more autonomy. Island PGAs appear to hold the most promise for cognitive network applications because the migration policy can be controlled whereas cellular PGAs require local communication at all agents for each generation of the PGA and master-slave PGAs require constant communication between master and slaves. Migration in island models occurs infrequently, which implies reduced communications between agents. Also, the migration in island PGAs can be asynchronous [30], which is a better fit for MAS than the synchronous requirements of cellular or master-slave PGAs. Additional flexibility is available in the injection island genetic algorithm (iGA), which allows island populations to have different resolutions in the binary encoding of parameters in a chromosome [30]. This may be useful in a cognitive network with heterogeneous nodes that have different computational abilities or different amounts of

energy available for computation.

Scatter Search Another member of the EA family is scatter search. Scatter search differs from a genetic algorithm in that new generations are constructed (deterministically or semi-deterministically) rather than randomly generated via mutation and crossover. The basic procedure for a scatter search is to select an initial population, choose a reference set of good and diverse solutions, and then enter an iterative loop of combining and improving solutions to create a new population from which a new reference set is selected [31]. If the reference set can no longer be improved, the algorithm may add diversity to the reference set and continue iterating.

Parallel scatter search is a recent development. Three initial approaches to parallelization given in [31] are synchronous parallel scatter search (SPSS), replicated combination scatter search (RCSS), and replicated parallel scatter search (RPSS). SPSS and RCSS are centrally controlled algorithms that require synchronous operation and the distribution of the reference set at each iteration. RPSS is similar to iGAs in that multiple populations exist simultaneously. Because there is no mechanism specified in [31] for exchanging subsets of the population (i.e. migration), we conclude that the algorithm is asynchronous, which is desirable for cognitive network applications.

A potential drawback of parallel scatter search (PSS) is that the combination and improvement operations are problem-dependent [32], which restricts the flexibility of a PSS. Particularly, it may be difficult to adapt a PSS to changes in the objective function. Since allowing changes in the end-to-end goals of a cognitive network implies changing the objective function, a PSS-based cognitive network may be restricted to having fixed end-to-end goals.

Tabu Search One of the most prominent metaheuristics is tabu search. The key elements of tabu search are a short-term memory called a tabu list, an intermediate-term memory, a

long-term memory, and a local transformation that defines the neighborhood of the current solution [33]. The use of memory in the search process is a major factor in the success of tabu search and distinguishes it from the metaheuristics we have discussed. The tabu list prevents the search from backtracking into territory that has been previously explored. The intermediate-term memory directs the search toward promising areas of the search space (intensification), and the long-term memory directs the search toward unexplored regions of the search space (diversification). By using these memories, tabu search learns from its exploration of the search space. When this learning is tied to the agent's knowledge base for use in future searches, tabu search is an example of a method that incorporates both learning and reasoning.

A taxonomy for parallel tabu search (PTS) is given in [34]. This taxonomy defines two classes of algorithms that have distributed control and asynchronous communications: *collegial* and *knowledge collegial*. Collegial identifies those PTSs in which a process (or agent) simply communicates promising solutions to other processes that make up the PTS. This class is similar to iGAs. Knowledge collegial is more complex in that agents communicate promising solutions but also try to infer global properties from the communicated solutions, such as common characteristics of good solutions.

3.2.4 Reasoning by Heuristic

Thus far, we have focused on methods that are general in nature and may be applied in a variety of circumstances with some tailoring of algorithm details. While metaheuristics in particular are often used to tackle problems that are known to be NP-hard, researchers frequently develop heuristic techniques that perform better by exploiting problem-specific attributes. In fact, this was our motivation for employing heuristics to address the problem in Chapter 4. As heuristics are generally problem-specific, the following examples do not necessarily directly translate into approaches that are meaningful for addressing networking problems, but are intended to show how heuristics exploit problem-specific attributes.

3.2.4.1 Job Shop Scheduling

Job shop scheduling is a classical NP-hard problem [35] in which a finite set of jobs is to be completed on a finite set of machines, with each job consisting of a set of operations that can only be performed on particular machines and in a prescribed order. The objective is to find a schedule of minimum length. A number of heuristic algorithms for job shop scheduling provide approximate solutions that are provably within some factor, ρ , of the optimum solution.

One such approximation algorithm, provided by Schmoys *et al.* [36], uses a randomized procedure to obtain a ρ -approximation with high probability. The jobs are first aligned so that they run to completion simultaneously. Since this schedule may require multiple jobs to run on the same machine at once, making the schedule infeasible, a randomized delay is prepended to each job, reducing the probability of conflict. In the final stage, the schedule is “flattened” so that any conflicts for the same machine are resolved by sequential scheduling of the operations.

Randomization is a powerful technique that is often used in heuristic algorithm development as well as in metaheuristics. Schmoys *et al.* are able to exploit the fact that introducing random delays to the job shop scheduling problem results in a reduction in the probability of conflict. Then, the ability of each machine to sequentially process any number of jobs is exploited to eliminate any conflicts that may remain.

3.2.4.2 Robot Motion Planning

Robot motion planning is the problem of, given a destination or final goal, determining the series of motion controls that will result in reaching the destination while avoiding potential obstacles along the way. The obstacles may be fixed in their location, or they may be moving, which leads to the dynamic motion planning problem. The two-dimensional dynamic point robot motion planning problem with bounded velocity, which has been shown to be NP-

hard [37], models the robot as a point in the plane that has a finite maximum velocity and assumes that obstacles are moving with fixed linear trajectories without rotating.

Kant and Zucker [38] tackle the two-dimensional dynamic robot motion planning problem with a two-level heuristic approach in which one level addresses global path planning while the other level performs local path planning. At the global level, the problem is broken down into path planning and trajectory planning, collectively referred to as path-velocity decomposition (PVD), where path planning is the process of breaking the overall path into straight-line segments and trajectory planning involves assigning a velocity to the robot for each segment in the path. The local-level heuristic is then left to address imminent collisions. When local sensors detect a nearby obstacle, the robot employs a repulsion-based algorithm to avoid a collision. If the avoidance algorithm causes the robot to be unable to continue the current global plan, the PVD stage is repeated to establish a new plan.

Kant and Zucker's heuristic contains elements of cognitive behavior such as sensing, actuation, reasoning, and planning. It exploits a geometric model of the robot's environment and simplifies path planning by constraining the plan to be a set of constant-velocity line segments. Since this linear path plan is not necessarily collision-free, Kant and Zucker add a local path-planning (i.e. collision avoidance) heuristic that handles situations in which the global path plan is leading the robot toward a collision. By constraining the global path plan to a finite set of constant velocity sub-plans and not requiring the global plan to be collision-free, the problem difficulty is greatly reduced. Then, the robot is supplied with a collision avoidance algorithm to handle situations in which the global path plan fails. This approach of first providing a best solution based on a subspace of the original problem and then providing a failure-recovery mechanism to handle issues that arise when implementing this solution may be useful as an approach for cognitive network heuristics.

3.2.4.3 Delay-constrained Least-cost Routing

In the standard routing problem, the goal is to deliver packets over paths that minimize a cost function. This problem is solvable in polynomial time using dynamic programming approaches. QoS routing imposes additional constraints that often increase the complexity of the problem. For example, delay-constrained least-cost routing, which routes a packet over the least-cost path that satisfies a delay constraint, is known to be NP-complete [39]. Chen and Nahrstedt tackle the delay-constrained least-cost routing problem in wireless networks using a ticket-based probing heuristic, in which the number of candidate routing paths that are searched is limited by the number of tickets issued by the source node [40]. For tighter delay constraints, more tickets are issued by the source.

Two types of tickets are issued: yellow tickets, which favor satisfying the delay constraint, and green tickets, which favor least-cost paths. The number of each type of ticket is based on a linear function of the delay constraint. Nodes that receive probes containing tickets split the tickets among outgoing links using an inverse proportional weighting of estimated delay for yellow tickets and an equivalent function using cost instead of delay for green tickets.

The reasoning behind the heuristic is that green tickets are used to aggressively search for a low-cost path that satisfies the delay constraint, with a high probability of failure, while yellow tickets serve as the backup by emphasizing discovery of paths that satisfy the delay constraint but may have much higher total cost. This approach is similar to that of the heuristic for the robot motion planning problem in Section 3.2.4.2 in that the heuristic first reduces the size of the problem space (by constraining the number tickets issued) and then provides a backup mechanism (yellow tickets) in case the primary mechanism (green tickets) fails.

3.2.5 Multiobjective Reasoning

The discussion so far has focused on cases in which the cognitive network has a single goal (or objective function), and perhaps additional constraints. However, it is also possible for the cognitive network to have multiple, potentially competing, goals. Reasoning then becomes a multiobjective problem. Multiobjective optimization attempts to first find members of the Pareto-optimal set, and then decide on a single solution from the set of Pareto-optimal solutions. Multiobjective DCOP is a recent development. Results in this area can be found in [25,41,42]. Multiobjective optimization is a well studied topic for evolutionary computation with some results for distributed multiobjective genetic algorithms in [43,44]. Bayesian networks can be teamed with multi-criteria decision making to handle multiple objectives [28].

3.2.6 Elements of Learning

Thus far we have discussed some powerful methods for reasoning based on current knowledge. However, these methods are severely impaired when there is no knowledge base available to guide reasoning. Teaming learning with reasoning makes the core of the cognitive process complete.

Learning methods may be classified as supervised, unsupervised, or rewards-based [45]. Supervised learning requires that the learning method be trained using known inputs and outputs provided by an expert. The expert is usually human, and the process of generating training data sets is generally laborious, if not infeasible. Rewards-based learning uses a feedback loop in which the feedback consists of measurements of the utility (i.e. reward) of prior actions. Rewards-based methods must then infer the correctness of prior actions based on the utility measurements. Unsupervised learning operates in an open loop without any assistance from an expert or utility measurements. Due to the difficulty of applying supervised and unsupervised learning methods to MASs, the majority of research on learning

in a MAS environment has focused on rewards-based learning [45]. It may be reasonable to use rewards-based learning for learning in cognitive networks as well since the possible variations in the network environment are too large to make supervised learning feasible, whereas performance measurements are readily available at all layers of the protocol stack.

One of the major issues encountered in developing a rewards-based learning system for a MAS is the credit assignment problem (CAP). The CAP is the problem of determining how to assign credit or blame to agents for the results of prior actions. It can be broken into two parts: the correlation of past actions with observations and the assignment of credit. The first stage is particularly difficult in a cognitive network because of the variation in response time to changes made at different layers of the protocol stack. For example, a change in transmit power may be observed within milliseconds as degraded signal-to-interference-and-noise ratio (SINR) at neighboring nodes. The same change in transmit power may also result in congestion apparent at the transport layer; however, the response time of the congestion observation is likely to be on the order of seconds.

For MASs in general, there are two levels of credit assignment that must be made: *inter-agent* CAP and *intra-agent* CAP [22]. Inter-agent CAP deals with assigning credit to agents based on the level of responsibility that each agent has for taking the actions. Since a MAS may not be cooperative, one agent may be more responsible for a particular action than any other and should therefore be assigned more credit or more blame. An example of this is the ACE algorithm [46] in which agents compete for the right to take actions based on estimates of their action's usefulness. Since only winning agents in ACE are allowed to implement their actions, credit should only be assigned to winning agents. The inter-agent CAP in a cognitive network may be simpler if agents take actions based on global agreement. In this case, a uniform credit assignment may be appropriate. This leaves intra-agent CAP, which is the process of determining the knowledge or inference that led to the action under consideration. To perform intra-agent CAP, a short-term history of past actions and their bases must be kept until actions can be correlated with their results. Agents must determine whether the action was due to inference from observations, knowledge in the knowledge base,

or some combination of the two and then respond appropriately.

3.2.7 Methods for Learning

Methods for learning should be considered along with the reasoning method. As we have seen with Bayesian networks, the coupling is sometimes so tight as to make the distinction between learning and reasoning difficult. While not all feasible pairings of learning and reasoning methods are this closely coupled, there is an inescapable dependency that necessitates their joint selection. The dependency comes from the need to apply what the learning algorithm has stored in the knowledge base. If this knowledge is not used, then learning is superfluous. This dependency motivates us to consider learning as paired with one of the methods of reasoning discussed previously.

In line with our discussion of rewards-based learning, we will discuss Q-learning and case-based reasoning. Reinforcement learning, of which Q-learning is an example, has been a popular subject for learning in MASs. For our discussion, we have paired Q-learning with DCOP. Case-based reasoning may seem misplaced for a section on learning, but, in fact, it is a framework that includes both reasoning and learning. We will discuss learning within the case-based reasoning framework and pair it with a metaheuristic for reasoning.

Bayesian networks do not fit cleanly within the supervised, unsupervised, and rewards-based classification of learning methods. This stems from being a probabilistic network. Learning in Bayesian networks can be thought of as occurring in two stages: first, learning of the network structure and initial conditional distribution estimates, and second, continual belief updating of the conditional distributions. In this section we focus on the first stage since the second stage is part of the previously discussed MSBN algorithm.

3.2.7.1 Q-learning

Q-learning (and reinforcement learning in general) models the world as a discrete-time, finite state, Markov decision process with unknown transition probabilities [47]. Actions result in an observable scalar reward that reveals something about the transition probabilities. Q-learning seeks to determine a policy that will maximize the expected reward. A policy, or decision rule, is a mapping from the observation/state space to the action space. The optimal policy is learned by an iterative process of first selecting an action based on the current state, a set of available actions, and a probability distribution and then updating the current policy based on the reward for the selected action (see [48] for a tutorial on Q-learning). Learning and reasoning are tightly coupled in Q-learning in that actions and the corresponding rewards are used to learn how to reason.

In distributed Q-learning, agents can be classified as performing joint action learning (JAL) or independent learning (IL) [49]. In JAL the agents are fully aware of the actions of all agents, whereas in IL the agents are only aware of their own actions. The JAL case implies communication between agents to share their actions. The IL approach is clearly beneficial when trying to minimize communication overhead in a cognitive network. However, convergence to the optimum learned policy is much more difficult to guarantee. An approach to this problem for stochastic environments is given in [50], where implicit coordination is used to overcome the lack of information about other agents' actions and converge to the optimal joint policy.

3.2.7.2 Learning of Bayesian Networks

As we mentioned in the discussion of BNs, the structure of the dependency graph may not be known a priori. An obvious consequence of this is that the conditional distributions are unknown as well. This issue is normally addressed by learning the structure of the DAG and then estimating the conditional distributions. The process requires a set of data consisting

of past combinations of inputs and outputs of the system. This data set is not training data in the sense of a supervised learning method because there is no concept of a correct output for any particular input. It consists of samples of the system's behavior.

The ability to learn the structure and conditional distributions of a BN allow a cognitive network to construct a set of beliefs about the network by observing the behavior of the network. This could be beneficial for networks that may be deployed in a variety of scenarios but cannot be fully characterized beforehand. One example of this is wireless networks for emergency response, where the geographic location for deployment of the network is not known until a disaster occurs. An additional benefit of learning a BN is that prior knowledge (such as that generated by an expert) can be incorporated into the BN to improve the learning process [51].

Learning a BN imposes a heavy computational burden, and this has led researchers to find ways to parallelize the process. For structures of moderate dimensionality, an asynchronous complete search such as in [52] may be feasible. For larger networks, researchers have turned to metaheuristics such as ant colony optimization [53], variable neighborhood search [54], and evolutionary algorithms [55] to provide approximate solutions to the BN structure. However, work on distributed metaheuristics for learning BNs seems to be in the early stages with one of the first results being reported in [56].

3.2.7.3 Case-based Reasoning

An example of a method that combines reasoning and learning is case-based reasoning (CBR). In CBR, the knowledge base (or case base) contains cases that are representations of past experiences and their outcomes. Dynamic networks will inevitably lead to disparity in the contents of agents' case bases. However, the structured contents of the case base can easily be shared among agents. This allows agents that have recently joined the network to benefit from the experience of other agents. This sharing of case bases also makes the cognitive network more robust to loss of agents, since the case base can be essentially a

distributed, overlapping database.

A four-stage cycle for CBR is presented in [57], with the stages being *retrieve*, *reuse*, *revise*, and *retain*. When a new case is encountered, the retrieve stage selects the most similar case from the knowledge base. The reuse stage combines the new case with this similar case to form a proposed solution. The efficacy of the proposed solution is evaluated in the revise stage, and the case is repaired if it failed to achieve the desired outcome. The final stage, retain, extracts any meaningful results and either enters the learned case into the knowledge base or updates existing cases. Based on this decomposition of CBR, reasoning consists of the retrieve and reuse stages while learning consists of the revise and retain stages.

Distributed case-based reasoning is discussed in [58, 59] where agents have individual responsibility for maintaining their case bases as well as sharing cases with other agents to improve global solutions. We conceptually pair case-based reasoning with a metaheuristic (this combination has been used in [60] for a single agent scenario using the genetic algorithm as the metaheuristic). The metaheuristic revises retrieved cases using information from past searches. This allows the metaheuristic to partition the search space so that it can either focus on finding a better solution in a region that has already produced good solutions or avoid trying to optimize dimensions of the search space that have little effect on solution quality.

The learning stages (revise and retain) are generally problem-specific when using case-based reasoning. For the cognitive network problem, learning may consist of evaluating how well a solution achieved the end-to-end goals. A similar problem to credit assignment occurs when evaluating the success of a solution. The response time of the network to the actions of the cognitive process may have different delays just as with Q-learning. This makes learning more difficult when there are multiple end-to-end goals (i.e. multiobjective optimization), especially when the goals apply to different protocol layers. Learning may also extend to the parameters used by the metaheuristic, in which case the cognitive network is learning how to search more efficiently.

3.3 Sensory and Actuator Functions

The MAS characteristic of *situated* encompasses the ability of agents to sense their environment as well as take actions on their environment. The ability to sense the environment provides the reasoning method with up-to-date information on the state of the network. These sensory inputs also provide the feedback necessary for learning. Actuator functions provide the ability to enact the solution or decision reached after reasoning. Therefore, sensory and actuator functions are vital to learning and reasoning.

Sensory functions lead to parallels between cognitive networks and sensor networks. Sensor networks are deployed to observe changes in an environment. In a cognitive network, the environment consists of the network itself as well as its surroundings. The cognitive network can be conceptually thought of as having a sensor network embedded within it that interacts with the cognitive elements through the SAN API (see Figure 2.1) to provide the status of the network. Sensor networks are usually considered to be limited in terms of available energy and processing power at each sensor node. The energy constraint drives sensor networks to limit communication in the network. In the cognitive network, it is also desirable to limit communication in the network, though the reason is to allow the communication channels in the network to serve users and applications as much as possible.

The processing limitations are less applicable to the sensors within cognitive networks. However, the cost of computation versus communication is leading some researchers to consider performing in-network computation [61] to reduce the amount of communication necessary to obtain the information of interest. It was pointed out in [61] that applications in sensor networks are often interested in aggregate queries such as averages or maximums. This is also relevant to cognitive networks where the observation of interest may be the average delay for flows in the network or the maximum transmit power over all the nodes. Thus, it is equally valuable to perform in-network computation in cognitive networks in order to reduce the required communication needed to obtain aggregates.

One point of difference between typical sensor networks and a cognitive network is that there is usually a single point in a sensor network that requires the aggregate observation, whereas in a cognitive network multiple nodes may need to know this result. One question that may be asked is whether this problem is even scalable in the number of nodes in the network. It has been shown in [62] that (using joint data compression) it is scalable for local observations to be globally distributed. In-network computation then eliminates redundant data and further reduces the load on the network.

Two methods that may be considered as types of in-network computation and data compression, respectively, are *filtering* and *abstraction* [3]. Filtering gives agents some autonomy over what local observations should be communicated to one or more agents. The learning process will ideally provide a means for agents to learn which observations have strictly local impacts and which observations need to be shared. When the nature of an observation is such that geographical proximity leads to correlation in agents' observations, agents may learn that only a subset of the cognitive network needs to report this observation. Abstraction is a common method of representing data in a more compact form. This is analogous to data compression in that redundant information has been removed.

The actuator functions are the means for the cognitive network to change its state as a result of reasoning and learning. Changes are enacted in the modifiable elements using the SAN API and must be coordinated by agents so that actions are coherent. Some changes, such as modulation type, addressing scheme, and MAC protocol, will require a degree of synchronization to prevent interruption to application sessions. In the worst case, poorly implemented actuator functions may lead to network instability. In this regard, research in distributed control may be beneficial.

3.4 Cognitive Node Architecture

A cognitive network achieves network-level cognition by integrating the cognitive cycle across layers in the protocol stack and throughout the nodes of the network. Such cognitive nodes require an architecture that supports observation of the state of the network, collective reasoning to achieve end-to-end network goals, learning from past actions, and reconfiguration of cognitive nodes based on collective decisions. In this section, we describe a cognitive node architecture that is intended to support cooperative distributed cognition.

The architecture presented herein has its roots in the work presented in [9,60] and is therefore derived from research in cognitive radio. The work in [9] provides a platform that allows reconfiguration of the whole protocol stack. We assume the presence of the type of platform described in [9] and focus on adding the ability to perform distributed learning and reasoning.

The cognitive node architecture is shown in Fig. 3.1. There are six major components of the architecture: the reconfigurable platform, stack manager, configuration and observation database (COD), exchange controller, distributed optimization process, and cognitive controller.

3.4.1 Reconfigurable Platform, Stack Manager, and COD

The reconfigurable platform and stack manager are inspired by the like-named components presented in [9]. Thus, each cognitive node has a flexible platform that allows the cognitive controller to choose from a variety of stack configurations in response to network conditions. The stack manager constructs the stack and reconfigures protocol layers.

The COD, which is a relational database, is the main repository for observations and configuration information. By keeping this information in a database, rather than internal to the cognitive controller, the information can be accessed by the exchange controller without interrupting the cognitive controller.

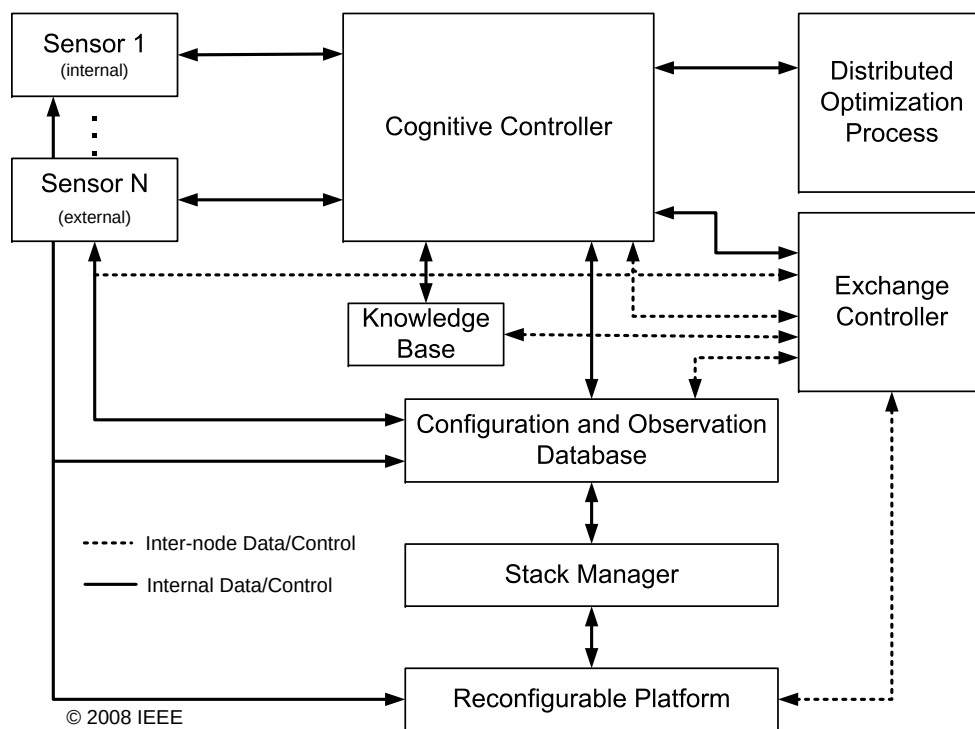


Figure 3.1: Cognitive Node Architecture ©2008 IEEE. Reprinted, with permission, from Friend *et al.*, “Architecture and performance of an island genetic algorithm-based cognitive network,” in *IEEE CCNC*, (Las Vegas, Nevada, USA), January 2008.

3.4.2 Exchange Controller

The exchange controller offloads communication and management overhead from the cognitive controller. Policies for exchanging observations between cognitive nodes are set by the cognitive controller and stored and enforced within the exchange controller. Thus, when another cognitive node requests data from the COD or knowledge base, the exchange controller responds according to established policy and without intervention from the cognitive controller. Requests from internal sensors for external data pass through the exchange controller so that multiple requests may be packaged in an efficient way. It is likely for observations to be on the order of a few bytes; therefore, the protocol overhead required to transport a single observation could be an order of magnitude larger than the observation itself if we assume a network protocol such as Universal Datagram Protocol/Internet Protocol (UDP/IP) in conjunction with MAC frame headers. By packaging observations appropriately, the exchange controller alleviates some of the protocol overhead in obtaining observations from other nodes.

The exchange controller acts as an application in communicating with the exchange controllers of other cognitive nodes, which explains its connection to the reconfigurable platform in Fig. 3.1. Therefore, application layer protocol processing occurs within the exchange controller. This creates greater modularity in the architecture and simplifies the design of the cognitive controller. Part of the application layer overhead in the exchange controller is providing secure communications between cognitive nodes so that the integrity of critical data, such as new configurations, is maintained. The exchange controller may also offload some of the coordination tasks associated with distributed reasoning.

3.4.3 Cognitive Controller and Distributed Optimization Process

The cognitive controller and distributed optimization process are the heart of the cognitive node. The cognitive controller manages the flow of information to the distributed optimiza-

tion process, deciding when it is necessary to re-optimize the current network configuration, as well as how the distributed optimization process should be configured. The cognitive controller also learns from past experience and stores the knowledge acquired by learning into the knowledge base.

Observations and experiences inherently contain uncertainty. This uncertainty may be related to sensing limitations (e.g. noise, nonlinear channels), but they may also be security-related, such as the trustworthiness of observations from an unknown source. As such, the cognitive controller should be based on a method that directly incorporates uncertainty into its processing.

In the following chapter, we change our focus from describing the cognitive network at a functional and architectural level to application of the cognitive network concept. By focusing on applications, we begin to provide greater clarity to how a cognitive network may operate on real problems.

Chapter 4

Secondary Network Multi-channel Topology Control for Minimizing Expected Outage Potential ¹

Chapters 2 and 3 have focused on higher levels of abstraction with regard to cognitive networks. To deal with the specifics of a cognitive network, thereby helping to solidify the ideas discussed in Chapters 2 and 3, it is useful to define the application to which the cognitive networking concept is to be applied. Hence, this chapter focuses on a particular application of cognitive networking. Our presentation first describes the problem and its associated model, followed by a description of the heuristic approach we have developed for solving the problem. We conclude by discussing how the problem solution satisfies the criteria for being a cognitive network.

In this chapter, we have chosen to apply cognitive networking to DSA. This decision is motivated by the wide-ranging association of cognitive radio with DSA as well as the close ties between cognitive radio and cognitive networks. These associations make it natural to ask how a cognitive network may be applied to DSA problems. We select an area within

¹This chapter is based on the work contained in [63,64].

DSA that has thus far been largely neglected and show how cognitive networks enable a solution approach.

To date, much of the focus of DSA research has been on detecting primary users and then establishing a network using spectrum that has been identified as unoccupied. Such focus on these areas is justified and must continue; however, due to the stochastic nature of wireless channels and the dynamic behavior of primary users, there exist scenarios where interference to primary users may occur, despite efforts to avoid it. Two such scenarios are:

- Despite an increased emphasis on distributed detection for DSA (see, e.g. [65,66]) and the accompanying improvement in detection capabilities, there will remain some non-zero probability of missed detection. Also, the missed detection probability will vary with the positions of the secondary network nodes relative to primary users, due to resulting variations in the received signal-to-noise ratio (SNR) of the primary user's signal, making it difficult to maintain a fixed probability of missed detection. This problem is compounded in fading/shadowing environments.
- Primary users may move into the proximity of secondary nodes, or primary user devices may be powered on while the secondary network is in operation on the primary user's channel. Once the secondary network is operating on a channel, there will be some delay before a newly appearing primary user is detected by the secondary network and further delay while the secondary network vacates the channel. For example, the channel abandonment time requirement for the DARPA XG DSA tests reported in [67] was 500 ms, with a network of up to 6 nodes requiring approximately 450 ms to abandon the channel with high probability. Assuming that not all secondary nodes can detect primary users with high probability (indeed, this is the motivating assumption behind distributed detection for DSA), we would naturally expect the channel abandonment time to increase with secondary network size due to forwarding delays in communicating both detection information (assuming distributed detection is used) and the decision to vacate the channel.

Since in either of these scenarios the secondary network may cause service interruptions, we are motivated to consider how the secondary network topology may be proactively chosen to be *environmentally friendly* to the primary network. In this chapter, we develop a quantitative measure of environmental friendliness based on outage probability that indicates the potential for a secondary network to cause outages to primary users, and using this measure, we formulate an objective function for selecting an optimal topology for the secondary network. The measure of environmental friendliness, called *expected primary outage potential* and signified by E_o , requires the secondary network to estimate the potential for primary users to appear and/or to have gone undetected within a region surrounding the secondary network.

This chapter addresses the single-channel case [63] separately from the multi-channel case [64] with heuristics for each. In addressing the multi-channel case, we have also developed a new power-based topology control problem along with a heuristic for solving it and metaheuristics for providing comparison solutions since complete search is impractical. The single-channel heuristic is compared against two well-known single-channel topology control algorithms, one that seeks to minimize total power and one that seeks to minimize interference.

In the following section, we discuss related work on multi-channel topology control as well as outage probability in networks. Then, in Section 4.2, we lay out the stochastic path loss model and describe the way in which a graph-based description for the multi-channel topology is obtained, as well as the way in which the MAC protocol is used to obtain a set of conflict graphs. In Section 4.3, we discuss the outage potential map and provide a simple method for generating the map. Section 4.4 presents the topology control objective function, describes the algorithm for estimating the expected outage potential for a given multi-channel topology, and presents our heuristics for the single- and multi-channel topology control problems. Section 4.5 describes the power-based multi-channel topology control algorithm and a heuristic for finding the min-max power multi-channel topology, which we then compare against our multi-channel heuristic for **MinMaxEo** topology control in Section 4.6. Finally, in Section 4.7, we describe how the solution to this DSA problem may

be viewed as a cognitive network.

4.1 Related Work

Much work has been done in outage probability for cellular networks (e.g. [68, 69]). More recently, outage probability has been used to analyze multihop spread spectrum networks for DSA [70]. In both the cellular and spread spectrum network models, network nodes are assumed to be constantly transmitting. Since packet networks tend to be bursty, our work focuses on outage probability in random access multi-hop packet networks by explicitly considering the effects of the MAC protocol on outage probability. Additionally, we do not make the simplifying assumption that the interferers are identically distributed at the receiver, as is often done for the cellular case. We further differentiate our work from existing work in this area by the fact that we do not assume that the location of the primary user is known. Instead, we focus on the total impact of the secondary network on its surrounding geographic region.

Topology control in wireless networks has also received a great deal of attention since the influential paper of Ramanathan and Rosales-Hain [71]. However, multi-channel topology control has been less emphasized. We consider the multi-channel topology control problem to be one in which both power and channel is assigned on a per-node or per-link basis (i.e. two degrees of freedom), subject to some constraint(s), to achieve some objective for the network. Some authors use similar terminology, but with a different meaning. For example, Marina and Das [72], Zhu *et al.* [73], and Naveed *et al.* [74] address topology control in multi-channel networks; however, none of these papers allow the node power to vary, making these forms of topology control more akin to channel assignment or logical topology control.

Thomas *et al.* [75] address what we consider to be a multi-channel topology control problem; however, they seek to minimize the number of channels used by the network without any constraint on the number of channels used, whereas we assume that a fixed set of channels are

unoccupied by primary users, making them available to the secondary network. Digham [76] also investigates a joint power and channel assignment problem in cognitive radio networks; however, the network consists of nodes communicating directly with an access point, making the network more similar to a cellular rather than ad hoc network. Additionally, our work differs significantly from the cited works in topology control in two ways: first, we directly account for the MAC protocol; and secondly, the objective function is referenced to an external entity (*exogenous*), whereas topology control objectives generally refer to the performance of the network being optimized (*endogenous*).

4.2 Channel and Network Models

In this section we present the assumptions and models for the wireless channel as well as the structure of the secondary network. Included in the network model is a graph-based model of the MAC protocol.

4.2.1 Path Loss and Outage Probability

Within the secondary network, we assume that the path loss between arbitrary pairs of secondary nodes is known (via measurement) for any pair of nodes with path loss small enough such that, when transmitting at maximum power, the received power level meets a detection power threshold, μ_d . However, we assume that path loss is unknown between a secondary user and any other point in space that is not occupied by another secondary user. Therefore, specifying the path loss model between secondary nodes and arbitrary points in space is essential for considering the effect on the primary network of energy radiated by secondary nodes.

While the use of deterministic path loss in networking research is widespread, wireless channels are most commonly described using stochastic models, such as lognormal shadowing,

Ricean fading, or composite random variables. We assume that unknown channel gains have a lognormal distribution. The lognormal distribution is a common model for capturing the medium-scale random variations in received signal strength. Finer-scale variations are often modeled by superimposing Ricean, Rayleigh, or Nakagami random variables over lognormal shadowing; however, since we feel it is impractical to compute outage probability on such a fine scale (e.g. every few centimeters), we have chosen a model that reflects more of an average fading effect. Therefore, we model channel gains as [77]

$$g = 10^{x/10} \frac{\lambda_o}{(d/d_o)^\gamma}, \quad (4.1)$$

where λ_o is a reference path loss at fixed distance d_o in the far-field of the transmitter, x is a zero-mean Gaussian random variable with standard deviation σ in dB (generally in the range of 4 to 12 dB), d is the distance between the transmitter and the point at which interference is being quantified, and γ is the path loss exponent (generally in the range 2 to 4). We refer to the (known) channel gain when secondary node i transmits to node j as g_{ij} .

4.2.1.1 Calculating Outage Probability

Since we have chosen a stochastic path loss model, the amount of interference at an arbitrary point in the plane due to secondary network transmissions is a random variable. This leads us to use outage probability to measure interference, where outage probability is simply, given the distances and transmit powers of a set of secondary nodes, the probability that the total interference caused by the secondary nodes will exceed a power level, μ_{out} , called the outage threshold. Formally, this is stated as

$$\rho = \Pr\left(\sum_k I_k > \mu_{out}\right), \quad (4.2)$$

where the I_k are assumed to be independent and are described by

$$I_k = 10^{x_k/10} \frac{\lambda_o P_k}{(d_k/d_o)^\gamma}$$

with P_k the transmit power of the k th interferer, d_k the distance from node k to the point at which interference is being quantified, and all other parameters as in (4.1). It should be noted that for lognormal random variables, the distribution of the sum of interferers, $\sum_k I_k$, is not known in closed form. However, various methods have been proposed for accurately estimating outage probability either via numerical methods (e.g. [68]) or by approximating the sum with another random variable (e.g. [78]).

4.2.2 Network Model

We assume that there are N nodes in the secondary network and that each node can transmit at any power in the set $\{0 \cup [P_{min}, P_{max}]\}$ on a per-link basis (i.e. node transmission power depends on the node to which it is transmitting). Directional link l_{ij} , originating at node i and terminating at node j , is included in the topology if $P_{ij} \cdot g_{ij} \geq \mu_c$, where P_{ij} is the power used when transmitting on l_{ij} , and μ_c is the link connectivity threshold. We begin by defining the maximal communications graph, $\mathcal{G}_{max} = (\mathcal{N}, \mathcal{L})$, which is the graph that results when all nodes transmit at P_{max} . Any other topology is necessarily a subgraph of $\mathcal{G}_{max}$. We designate the total number of possible links as $L = |\mathcal{L}|$.

Additionally, we assume that there is a set of channels, $\mathcal{H}$, that the secondary network has determined are available for use and that each node is capable of using any available channel on any of its originating links. The set of channels is assumed to be the same for all nodes, as may be the case when nodes perform distributed detection, thereby making collective decisions as to the availability of channels. Nodes possess multi-channel radios, meaning that they can transmit and/or receive on multiple channels simultaneously. Such multi-channel capability through a wideband transceiver has been emphasized in software radio

research over the last decade (for examples, see [79–81]).

The assignment of channels is transparent to the flow of data so that communication graphs, including $\mathcal{G}_{max}$, are defined solely by link power levels. A feasible topology is defined by the assignment of a valid power level (perhaps zero) and channel to each link in $\mathcal{G}_{max}$, resulting in the link power assignment vector $\bar{P}$. Each $\bar{P}$ induces a topology represented by the graph $\mathcal{G}_{\bar{P}} = (\mathcal{N}, \mathcal{L}_{\bar{P}})$ where

$$\mathcal{L}_{\bar{P}} = \{l_{ij} | P_{ij} \cdot g_{ij} \geq \mu_c\},$$

with P_{ij} being the power assigned to link l_{ij} in $\bar{P}$. We restrict the elements of $\bar{P}$ in the following way:

$$P_{ij} \in \{0, \max(P_{min}, P_{ij} | P_{ij} \cdot g_{ij} = \mu_c)\}, \quad (4.3)$$

to ensure that any link included in the topology will always use the minimum power necessary to close the link, thereby contributing the least possible interference on a per-link basis.

A feasible multi-channel topology consists of the power assignment vector $\bar{P}$ along with the channel assignment vector $\bar{H}$. The channel assignment for link l_{ij} is denoted by H_{ij} . A channel assignment vector is feasible if each element, H_{ij} , of the vector $\bar{H}$ satisfies $H_{ij} \in \mathcal{H}$. Combining channel and power assignment results in a feasible space consisting of $|\mathcal{H}|^L$ possible multi-channel topologies, where $|\mathcal{H}|$ is the number of channels in $\mathcal{H}$.

For practical purposes (e.g. to allow link layer acknowledgments and standard routing protocols), we restrict ourselves to topologies that include only bi-directional pairs of links (i.e. if l_{ij} is included in the topology, then l_{ji} is also included). However, we describe the network as a collection of directed links because the directionality is necessary for calculation of outage probabilities, as we will describe in Section 4.2.2.2. Our use of bi-directional link pairs leads to the following definition of connectivity:

Definition 4.2.1. *A directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{L})$, consisting of a set of nodes $\mathcal{N}$ and a set of directed links $\mathcal{L}$, is bi-directionally connected if the following condition holds:*

For every pair of nodes $n_1, n_2 \in \mathcal{N}$, there exists at least one directed path from

n_1 to n_2 such that, for every $l_{ij} \in \mathcal{L}$ on the path, it is also true that $l_{ji} \in \mathcal{L}$.

4.2.2.1 The MAC Protocol

We assume that all nodes in the secondary network use a CSMA-like MAC protocol. In the MAC protocol, a node listens to the channel assigned to l_{ij} before attempting to transmit on l_{ij} . Node j can detect that node i is transmitting (to node k) if

$$P_{ik} \cdot g_{ij} \geq \mu_d,$$

where μ_d is the detection threshold. We consider an idealized version of CSMA in that we assume: it takes zero time to detect a transmitting node, there is zero signal propagation delay between nodes, and links will not be simultaneously active if they *conflict*. We say that a pair of links, l_{ij} and l_{pq} , conflict if $H_{ij} = H_{pq}$ and either $P_{ij} \cdot g_{ip} \geq \mu_d$ or $P_{pq} \cdot g_{pi} \geq \mu_d$. Additionally, links originating at the same node and assigned the same channel will always conflict.

4.2.2.2 Conflict Graph

We may use the channel assignment vector to partition $\mathcal{G}_{\bar{P}}$ into subgraphs, one for each channel. The subgraph for channel $H \in \mathcal{H}$ is defined as $\mathcal{G}_{\bar{P},H} = (\mathcal{N}, \mathcal{L}_{\bar{P},H})$ where

$$\mathcal{L}_{\bar{P},H} = \{l_{ij} \in \mathcal{L}_{\bar{P}} | H_{ij} = H\}.$$

Using the MAC protocol, we can construct a set of undirected conflict graphs, one for each $\mathcal{G}_{\bar{P},H}$. The conflict graph for channel H is denoted by $\mathcal{C}_{\bar{P},H} = (\mathcal{V}_{\bar{P},H}, \mathcal{E}_{\bar{P},H})$, which describes the sets of links that may be active simultaneously on channel H (a single-channel version of a conflict graph is used in [82]). To avoid confusion, we use the graph-theoretic terms *vertex* and *edge* when referring to a conflict graph and the terms *node* and *link* when referring to a

communications graph. To construct the conflict graph, we take the directional links of $\mathcal{G}_{\bar{P},H}$ (i.e. those links with non-zero power in $\bar{P}$ that have been assigned to channel H) and make them the vertices of $\mathcal{C}_{\bar{P},H}$. An edge exists between two vertices of $\mathcal{C}_{\bar{P},H}$ if the corresponding links in $\mathcal{G}_{\bar{P},H}$ conflict with each other, indicating that they cannot be active simultaneously.

Maximal Independent Sets The notion of an independent set is important in determining outage probabilities in the secondary network. An independent set of the graph $\mathcal{C}_{\bar{P},H}$ is a set of vertices that share no incident edges. A maximal independent set (MIS) is an independent set that is not a proper subset of any other independent set. For the conflict graph $\mathcal{C}_{\bar{P},H}$, the set of all MISs is denoted by $\mathcal{M}_{\bar{P},H}$.

The concept of a MIS is important for our problem because it describes the largest sets of interferers that are allowed by the MAC protocol on a particular channel. By focusing on MISs as the candidate sets of interferers, we are choosing a per-channel worst-case scenario in that, although independent sets are equally valid under the MAC protocol, we only consider outage probabilities that result from the largest sets of interferers allowed by the MAC protocol in a particular channel. As such, the outage probability at a point in the plane, ξ , resulting from the MIS $M \in \mathcal{M}_{\bar{P},H}$ is given by re-writing (4.2) in the form

$$\rho_{\xi}(M) = \Pr \left(\sum_{l_{ij} \in M} 10^{x_i/10} \frac{\lambda_o P_{ij}}{(d(i, \xi)/d_o)^{\gamma}} > \mu_{out} \right),$$

where $d(i, \xi)$ is the distance from node i to the point ξ .

4.3 Outage Potential Map

While we may construct an interesting topology control objective using only our network model combined with outage probabilities, it would be better to combine these with knowledge of primary user tendencies gained from prior observations. We use this additional

knowledge to estimate the *outage potential* at a discrete set of points in the geographic neighborhood of the secondary network, called the region of influence (ROI). The term outage potential is meant to reflect the fact that these estimates are forward-looking, in that outages are not necessarily occurring at the present time, and some points in the plane may be more likely than others to have either missed the presence of primary users or for primary users to appear in the future.

This knowledge is geographically oriented, meaning that outage potential is estimated for a discrete set of points in the geographic neighborhood of the secondary network. Such an approach is along the lines of the spectrum hole probability grid used by Shared Spectrum for their XG architecture [83] and the radio environment map proposed by Zhao et al. [84]. Almost 20 years ago, Elfes [85] introduced another closely related concept called the *occupancy grid*, which has been successfully applied for mobile robot navigation in unknown environments using observations made by sonar and/or stereo ranging sensors. Elfes uses Bayesian methods for updating the occupancy grid, whereas others use Dempster-Shafer theory [86], which is also used in updating the spectrum hole probability grid [83].

We assume that the secondary network has estimated the outage potential at a discrete set of points in its neighborhood and for each channel of interest. These points, designated by ψ_i , are the tile centers of a tiling of the ROI, with the complete set of tiles denoted by $\mathcal{A}$. Each tile has a set of channel-dependent weights, $w_i(H)$, that reflect the secondary network's estimate of the potential for primary users to have gone undetected or to reappear in channel H and within that tile. In a slight abuse of notation, we use ψ_i to denote both the point at the center of a tile as well as the tile itself.

4.3.1 Maximum Likelihood Position Estimation for Outage Potential Maps

The procedure for estimating the outage potential map warrants considerable research of its own. Therefore, in this chapter, we only provide an outline of one rather simplistic way for estimating the outage potential grid. In our outline, we assume that the secondary network employs a distributed detection algorithm to increase the probability of correct detection. In a distributed detection setting, each secondary network node makes a local observation of a channel, after which these observations are combined in some manner to reach a final decision as to whether the channel is occupied.

Since the observations gathered by secondary nodes for primary user detection are either in the form of received signal strength (RSS) or require raw data from which RSS is readily obtained, we leverage the availability of RSS observations to perform position estimation. Patwari *et al.* [87] investigate the problem of how to determine location estimates for a set of nodes, referred to as the blindfolded devices, for which position is unknown, using RSS measurements from a set of reference nodes for which position is known. Their maximum likelihood (ML) position estimate can be adapted for use in estimating the position of a primary user by treating the primary user as a blindfolded device and treating the secondary nodes as the reference nodes. In this case, the planar maximum likelihood estimate, $\{x^*, y^*\}$, is the solution to the following:

$$\{x^*, y^*\} = \arg \min_{x \in \mathcal{X}, y \in \mathcal{Y}} f(x, y) \quad (4.4)$$

where

$$f(x, y) = \frac{b}{8} \sum_{i \in \mathcal{R}} \ln^2 \frac{\hat{d}_i^2}{d_i^2(x, y)} - \sum_{i=1}^N \ln \left[Q \left(\frac{b}{2} \ln \frac{d_{thr}^2}{d_i^2(x, y)} \right) \right], \quad (4.5)$$

$\mathcal{R}$ is the set of secondary nodes that detected the primary user, $\hat{d}_i$ is the expected distance between secondary node i and the primary user based on the received power level at node i , $d_i(x, y)$ is the distance from the secondary node i if the primary user is assumed to be at

position $\{x, y\}$, d_{thr} is the distance beyond which the expected received power falls below the detection threshold of reference nodes' receivers, and Q is the complement of the standard normal cumulative distribution function. Additionally, the estimate assumes a lognormal shadowing path loss model that is identical to (4.1) so that b is given by

$$b = \frac{10\gamma}{\sigma \ln(10)}, \quad (4.6)$$

where γ and σ are as in (4.1). The maximum likelihood estimate uses the observed power at the set of nodes that detected the primary user (represented by the left sum in (4.5)) as well as the lack of detection by the remaining nodes (represented by the right sum in (4.5)). After differentiating (4.5), a conjugate gradient algorithm may be used to find the minimum.

We assume that spectrum sensing is performed on a periodic basis. Then, by keeping track of the ML position estimates every time a primary user is detected on each channel, we compute $w_i(H)$ as the number of ML position estimates that fall within ψ_i and on channel H over some fixed interval of time, τ , normalized by the total number of observations made over τ . Using this approach, we may interpret $w_i(H)$ as an estimate of the probability that a primary user will appear within ψ_i and on channel H in a future observation. Admittedly, this approach is simplistic; however, it is sufficient for our purposes.

4.4 MinMaxEo Topology Control

In this section, we formalize the **MinMaxEo** topology control problem and describe a heuristic procedure for the problem.

4.4.1 Formalizing the Objective

Having described our network model, the outage potential map, and how to compute outage probabilities, we now use these to describe the objective of our topology control problem.

Our objective is to select the topology that causes the min-max per-channel expected outage potential, subject to the constraint that the topology be bi-directionally connected.

To formally state the objective, we start with the single-channel expected outage potential, which is written as

$$E_o(\bar{P}, H) = \sum_{\psi_i \in \mathcal{A}} \left(\frac{w_i(H)}{|\mathcal{M}_{\bar{P}, H}|} \sum_{M \in \mathcal{M}_{\bar{P}, H}} \rho_{\psi_i}(M) \right), \quad (4.7)$$

where $w_i(H)$ is the outage potential of channel H in tile ψ_i , $\rho_{\psi_i}(M)$ is the outage probability at the center of tile ψ_i when the MIS M is active, $\mathcal{M}_{\bar{P}, H}$ is the set of MISs of conflict graph $\mathcal{C}_{\bar{P}, H}$, and $|\mathcal{M}_{\bar{P}, H}|$ is the total number of MISs. The equal weighting of MISs in (4.7) reflects ignorance of the traffic patterns in the secondary network. The overall measure of a multi-channel topology is the maximum per-channel expected outage potential

$$E_o(\bar{P}, \bar{H}) = \max_H E_o(\bar{P}, H). \quad (4.8)$$

Topology $(\bar{P}_1, \bar{H}_1)$ is better than topology $(\bar{P}_2, \bar{H}_2)$ if $E_o(\bar{P}_1, \bar{H}_1) < E_o(\bar{P}_2, \bar{H}_2)$ because, on average, $(\bar{P}_1, \bar{H}_1)$ has less potential to cause outages. Naturally, we seek the topology that has the least potential to cause outages. Therefore, the topology control objective is to find the topology $(\bar{P}^*, \bar{H}^*)$ that satisfies

$$E_o(\bar{P}^*, \bar{H}^*) = \min_{\bar{P}, \bar{H}} E_o(\bar{P}, \bar{H}). \quad (4.9)$$

We call the min-max per-channel expected outage potential topology control problem **Min-MaxEo**.

4.4.2 Issues in Finding the Optimum Topology

Direct evaluation of (4.9) is problematic on multiple fronts:

1. The size of the feasible multi-channel topology space is exponential in the number of links L , meaning that it is also exponential in N .
2. For practical values of P_{max} , P_{min} , μ_c , and μ_d , experimental results show that the size of $\mathcal{M}_{\bar{P},H}$ is also exponential in the number of links assigned to H . Therefore, any algorithm requiring enumeration of all MISs cannot be performed in polynomial time.
3. Since interference is the sum of lognormal interferers, which has no closed-form distribution function, computing each $\rho_{\psi_i}(M)$ involves either numerical methods or approximating the sum of interferers by another random variable, either of which is computationally expensive and must be performed repeatedly for every MIS and every tile ψ_i .

Each of these issues is clearly a challenge and provides motivation for using a non-optimal heuristic. Therefore, we have developed a heuristic algorithm for finding approximate solutions to (4.9), the details of which are given in the following sections. This heuristic is derived by addressing each of the three main issues listed above, resulting in a polynomial-time algorithm.

4.4.3 Randomized Algorithm for Estimating Single-Channel Expected Outage Potential

As was mentioned in Section 4.4.2, there are multiple difficulties with evaluating the expected outage potential of any multi-channel topology, much less finding an optimum topology. In fact, the MIS enumeration problem is known to be NP-hard in the general case [88]. Since evaluating candidate solutions is a necessary intermediate step in our heuristic, we are in need of an efficient means of estimating (4.8). Because (4.8) is easily calculated from the set of single-channel expected outage potentials, we focus on estimating (4.7).

We first rewrite (4.7) as

$$E_o(\bar{P}, H) = \frac{1}{|\mathcal{M}_{\bar{P}, H}|} \sum_{M \in \mathcal{M}_{\bar{P}, H}} \sum_{\psi_i \in \mathcal{A}} \rho_{\psi_i}(M) w_i(H), \quad (4.10)$$

which tells us that we can compute $E_o(\bar{P}, H)$ by first computing the channel's expected outage potential on a per-MIS basis and then averaging over all MISs. This alone is not very helpful, but if we are able to approximate the per-MIS calculations using per-link calculations, then we no longer have to repeatedly apply numerical methods to calculate outage probabilities for sums of lognormal random variables. To develop this part of the approximation, we define two quantities related to (4.10), the channel-dependent per-MIS expected outage potential, given by

$$E_o(M, H) = \sum_{\psi_i \in \mathcal{A}} \rho_{\psi_i}(M) w_i(H), \quad (4.11)$$

and the per-link expected outage potential, given by

$$E_o(l_{ij}, H) = \sum_{\psi_i \in \mathcal{A}} \rho_{\psi_i}(l_{ij}) w_i(H), \quad (4.12)$$

where $\rho_{\psi_i}(l_{ij})$ is the outage probability in tile ψ_i when only link l_{ij} is active. This leads to the following approximation:

$$\hat{E}_o(M, H) = \sum_{l_{ij} \in M} E_o(l_{ij}, H). \quad (4.13)$$

By using (4.13), we can take advantage of the fact that outage probability from a single lognormal interferer can easily be calculated using the cumulative distribution function for a Gaussian random variable. We also gain the advantage of being able to pre-compute (4.12) once for each link-channel pair.

As this link-sum approximation will be used as a building block for estimating (4.10), we

must first evaluate its accuracy under various lognormal shadowing models. Fig. 4.1 shows histograms of the ratio of the approximation in (4.13) to the actual values for various lognormal shadowing parameters in sets of 100 uniform randomly generated 15-node networks. The number of MISs per network ranges from several hundred to several thousand so that each histogram contains tens of thousands of samples. The approximation is fairly concentrated around the zero error point (a ratio of 1 implies zero error) for the three lognormal parameter sets $(\sigma=7.1, \gamma=3.8)$, $(\sigma=8.3, \gamma=2.5)$, and $(\sigma=9.6, \gamma=2.8)$, which reflect real measurements reported in [89]. The parameter sets $(\sigma=12.0, \gamma=4.0)$ and $(\sigma=4.0, \gamma=2.0)$ are used to show the performance at the extremes of σ and γ , and while the lower extreme results in some of the most accurate approximations of all parameter sets shown, we see that at the upper limit the approximation is consistently overestimating the actual value. Since $(\sigma=12.0, \gamma=4.0)$ is an extreme case, we interpret the estimation errors for these path loss parameters as a bound on the approximation's performance.

Although (4.13) allows us to avoid the numerical computation necessary for outage probability due to the sum of lognormal interferers, we still need to address the problem of MIS enumeration. To do this, we insert (4.13) back into (4.10) to obtain

$$\hat{E}_o(\bar{P}, H) = \frac{1}{|\mathcal{M}_{\bar{P}, H}|} \sum_{M \in \mathcal{M}_{\bar{P}, H}} \sum_{l_{ij} \in M} E_o(l_{ij}, H). \quad (4.14)$$

If we define the counting function

$$c(l_{ij}, \mathcal{M}_{\bar{P}, H}) = \sum_{M \in \mathcal{M}_{\bar{P}, H}} \mathbf{1}(l_{ij}, M),$$

where $\mathbf{1}(l_{ij}, M)$ is the indicator function

$$\mathbf{1}(l_{ij}, M) = \begin{cases} 1 & \text{if } l_{ij} \in M \\ 0 & \text{otherwise} \end{cases},$$

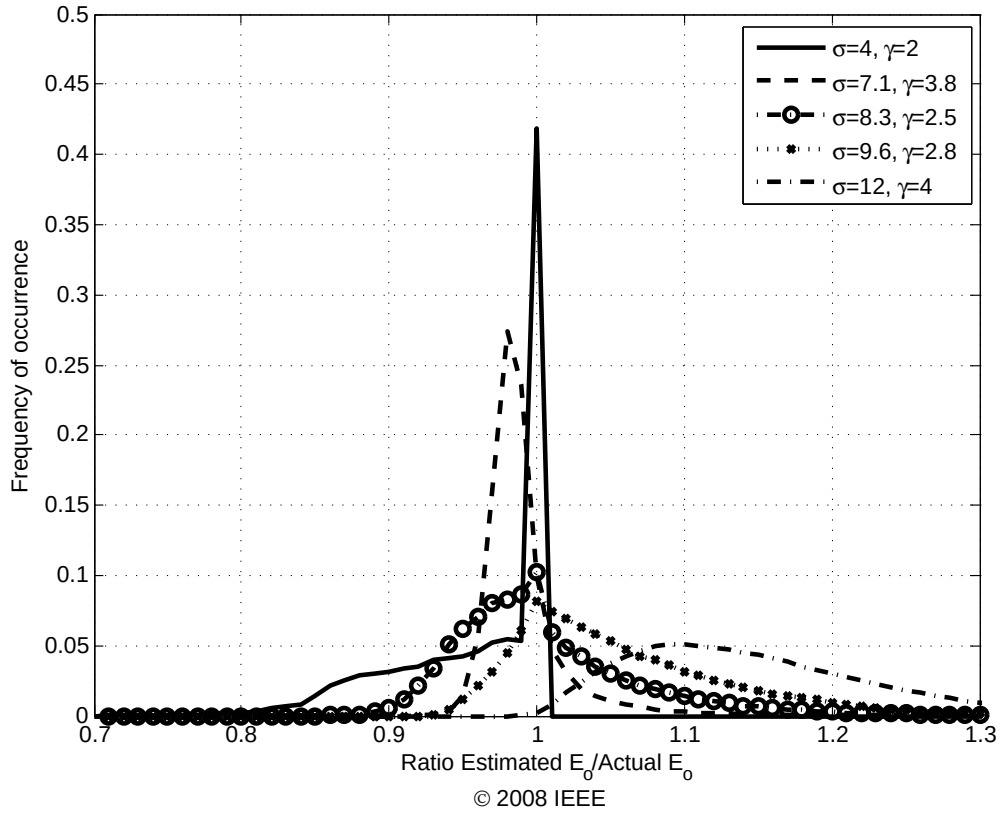


Figure 4.1: Histograms of $\hat{E}_o(M, H)/E_o(M, H)$ ©2008 IEEE. Reprinted, with permission, from Friend *et al.*, “Environmentally-friendly secondary network topology control for minimizing outage potential,” in *IEEE DySPAN*, (Chicago, IL), pp. 178–185, October 2008.

then we can rewrite (4.14) as

$$\hat{E}_o(\bar{P}, H) = \sum_{l_{ij} \in \mathcal{G}_{\bar{P}, H}} \frac{c(l_{ij}, \mathcal{M}_{\bar{P}, H}) E_o(l_{ij}, H)}{|\mathcal{M}_{\bar{P}, H}|}. \quad (4.15)$$

This indicates that instead of enumerating all MISs, we can use the link counts of $\mathcal{M}_{\bar{P}, H}$ along with the total number of MISs to obtain $\hat{E}_o(\bar{P}, H)$.

Unfortunately, despite significant research in graph theory on finding the maximum length of a MIS as well as the number of MISs for a particular graph, we are aware of no results (exact or bounds) related to finding the distribution of links in the set of MISs by any method other than enumeration, and existing bounds on $|\mathcal{M}_{\bar{P}, H}|$ are not sufficiently tight as to be useful [90]. However, after defining the functions for the ratio of per-link count to the total link count

$$r(l_{ij}, \mathcal{M}_{\bar{P}, H}) = c(l_{ij}, \mathcal{M}_{\bar{P}, H}) / \sum_{l_{ij}} c(l_{ij}, \mathcal{M}_{\bar{P}, H}),$$

and the average number of links per MIS

$$m(\mathcal{M}_{\bar{P}, H}) = \sum_{l_{ij}} c(l_{ij}, \mathcal{M}_{\bar{P}, H}) / |\mathcal{M}_{\bar{P}, H}|,$$

we see that (4.15) can be rewritten as

$$\hat{E}_o(\bar{P}, H) = m(\mathcal{M}_{\bar{P}, H}) \sum_{l_{ij} \in \mathcal{G}_{\bar{P}, H}} r(l_{ij}, \mathcal{M}_{\bar{P}, H}) E_o(l_{ij}, H),$$

which shows that we may use $m(\mathcal{M}_{\bar{P}, H})$ and $r(l_{ij}, \mathcal{M}_{\bar{P}, H})$ instead of absolute link counts.

For topologies with a large number of MISs, our intuition suggests that a random sampling of the MISs can provide estimates of the link count ratios as well as the average number of links per MIS. In constructing a randomized algorithm for generating MISs, it is important that there be as little bias as possible in the sampling method. Ideally, the algorithm would generate any particular MIS with probability $1/|\mathcal{M}_{\bar{P}, H}|$. Jain et al. [82] provide a simple

randomized algorithm for finding MISs in which links are randomly ordered by assigning a uniform random number to each link and then sorting. The MIS is constructed by selecting the link with smallest random value, eliminating all conflicting links, and repeating until there are no links remaining. However, this method clearly does not uniformly sample $\mathcal{M}_{\bar{P},H}$. Consider a link that conflicts with all other links, in which case it will only appear in a single MIS. Jain's algorithm would generate this link's MIS with probability $1/L$, which is generally much larger than $1/|\mathcal{M}_{\bar{P},H}|$.

Intuitively, we would expect links to appear in MISs with a frequency that is roughly inversely proportional to the number of other links with which it conflicts. In graph-theoretic terms, the number of links with which l_{ij} conflicts is the *degree*, $\deg(v_{ij})$, of the corresponding vertex in the conflict graph. Therefore, we propose the following algorithm for randomly constructing an MIS of $\mathcal{C}_{\bar{P},H}$:

- 1: Assign value $p_{ij} = 1/\deg(v_{ij})$ to each v_{ij} ;
- 2: Normalize p_{ij} by $\sum_{l_{ij}} 1/\deg(v_{ij})$ so that $\sum p_{ij} = 1$;
- 3: Initialize $M = \emptyset$, $\mathcal{V} = \mathcal{V}_{\bar{P}}$;
- 4: **while** $\mathcal{V} \neq \emptyset$ **do**
- 5: Randomly select one vertex, $v^* \in \mathcal{V}$, each with probability p_{ij} ;
- 6: Add v^* to M and remove it from $\mathcal{V}$;
- 7: Remove any links that conflict with v^* from $\mathcal{V}$;
- 8: Re-normalize each p_{ij} by $\sum_{v_{ij} \in \mathcal{V}} 1/\deg(v_{ij})$;
- 9: **end while**

The procedure is repeated for a constant multiple of the number of links, and estimates for link count ratios and average MIS length are calculated from the set of MISs. Using the estimates for the link count ratios $\hat{r}(l_{ij}, \mathcal{M}_{\bar{P},H})$ and the average MIS length $\hat{m}(\mathcal{M}_{\bar{P},H})$, we obtain the estimate

$$\check{E}_o(\bar{P}, H) = \hat{m}(\mathcal{M}_{\bar{P},H}) \sum_{l_{ij} \in \mathcal{G}_{\bar{P},H}} \hat{r}(l_{ij}, \mathcal{M}_{\bar{P},H}) E_o(l_{ij}, H). \quad (4.16)$$

We repeat (4.16) for each channel and then take the maximum of the per-channel estimates to obtain $\ddot{E}_o(\bar{P}, \bar{H})$.

4.4.4 Single-channel Heuristic

Our work in [63] addressed the single-channel case of (4.9), which is equivalent to (4.7). We use the algorithm described in Section 4.4.3 for estimating $E_o(\bar{P}, H)$, but we use a slightly different heuristic from the multi-channel case because there is no need to determine how to assign links to channels. For simplicity and to indicate that we are addressing the single-channel case, we drop the H from our equations in this section.

Since issues 2) and 3) of Section 4.4.2 are addressed by the algorithm in Section 4.4.3, we are left to deal with issue 1), which is the exponential size of the solution space. For this, we reason that we would expect the optimal solution to (4.7) to contain the spanning tree of $\mathcal{G}_{max}$ with the lowest E_o . Since the number of spanning trees of $\mathcal{G}_{max}$ is far too large for a complete search to find the one with minimum E_o , we instead construct the $E_o(l_{ij})$ -weighted minimum spanning tree (MST). The MST is generally defined on undirected graphs, which does not present a problem for us since each bi-directional topology can be easily converted to an undirected graph. The weight of each link in the undirected graph becomes $E_o(l_{ij}) + E_o(l_{ji})$. Well-known polynomial-time algorithms for finding the MST include Prim's algorithm and Kruskal's algorithm.

Since the $E_o(l_{ij})$ -weighted MST is not necessarily the same as the MST with minimum E_o , we perform a local search in the neighborhood of the $E_o(l_{ij})$ -weighted MST using single-link-pair replacement to move from one spanning tree to another with lower $\ddot{E}_o(\bar{P})$. We first find all link pairs for which either $E_o(l_{ij}) < \ddot{E}_o(\bar{P})$ or $E_o(l_{ji}) < \ddot{E}_o(\bar{P})$. These are the candidate replacement links. One-by-one, we replace a link in $\mathcal{G}_{\bar{P}}$ with a candidate replacement link and compute $\ddot{E}_o(\bar{P}')$ for any replacement that results in a connected topology. From the set of replacements for which $\ddot{E}_o(\bar{P}') < \ddot{E}_o(\bar{P})$, we choose the one with lowest $\ddot{E}_o$ and modify the topology accordingly. This procedure is repeated until we either cannot find a topology

that decreases $\ddot{E}_o$ or we select a topology that has been previously chosen. Alternatively, a fixed number of iterations may be used, though in our simulations, this procedure usually terminated in less than 20 iterations.

In general, the optimal solution need not be a spanning tree. So, instead of stopping after computing the MST, we estimate $E_o(\bar{P}_{mst})$ using (4.16) and compare the result with the per-link E_o values of the links that are not included in $\bar{P}_{mst}$. We iteratively add the pair of links with smallest $E_o(l_{ij}) + E_o(l_{ji})$ to $\mathcal{G}_{\bar{P}}$ and re-compute (4.16). If $\ddot{E}_o$ decreases, then we include the pair of links in the final topology. This procedure is described in the following algorithm:

- 1: Compute $E_o(l_{ij})$ for $l_{ij} \in \mathcal{L}_{max}$
- 2: Find the $E_o(l_{ij})$ -weighted MST, $\mathcal{G}_{\bar{P}_{mst}}$
- 3: Calculate $E_o = \ddot{E}_o(\bar{P}_{mst})$
- 4: Find $\mathcal{E} = \{(l_{ij}, l_{ji}) | E_o(l_{ij}) < E_o(\bar{P}_{mst}), l_{ij} \notin \mathcal{L}_{\bar{P}_{mst}}\}$
- 5: Initialize $\mathcal{L} = \mathcal{L}_{\bar{P}_{mst}}$, $\bar{P} = \bar{P}_{mst}$
- 6: **while** $\mathcal{E} \neq \emptyset$ **do**
- 7: Choose $(l'_{ij}, l'_{ji}) = \min_{\mathcal{E}} E_o(l_{ij}) + E_o(l_{ji})$
- 8: $\bar{P}' = \text{update } \bar{P} \text{ to include } (l'_{ij}, l'_{ji})$
- 9: Calculate $E'_o = \ddot{E}_o(\bar{P}')$
- 10: **if** $E'_o \leq E_o$ **then**
- 11: Set $E_o = E'_o$, $\bar{P} = \bar{P}'$, $\mathcal{L} = \{\mathcal{L} \cup l'_{ij}, l'_{ji}\}$
- 12: **end if**
- 13: $\mathcal{E} = \{\mathcal{E} - (l'_{ij}, l'_{ji})\}$
- 14: **end while**
- 15: **return** $\mathcal{G} = (\mathcal{N}, \mathcal{L})$

4.4.5 Multi-channel Heuristic

The multi-channel heuristic for finding an approximate solution to (4.9) consists of three stages, of which the final two incorporate the estimation procedure in Section 4.4.3.

4.4.5.1 Stage 1

In the first stage, we find the MST of an undirected graph (thereby guaranteeing that the bi-directional connectivity constraint is met) with link weights being set to $\min_H E_o(l_{ij}, H) + \min_H E_o(l_{ji}, H)$. Computing the link weights requires a number of computations that is polynomial in $|\mathcal{L}|$, $|\mathcal{H}|$, and $|\mathcal{A}|$, and the MST then can be found in time that is polynomial in $|\mathcal{L}|$. The resulting MST defines the power vector $\bar{P}$, leaving channel assignment for stages 2 and 3.

4.4.5.2 Stage 2

In stage 2 of the heuristic, we take a first cut at choosing $\bar{H}$ by assigning pairs of conflicting links greedily to the channel that has the lowest $\check{E}_o(\bar{P}, H)$ after adding the links. The algorithm is as follows:

- 1: Initialize $\mathcal{L}' = \mathcal{L}_{\bar{P}}$, $\check{E}_o(\bar{P}, H) = 0 \forall H$, and $\check{E}'_o(\bar{P}, H) = 0 \forall H$;
- 2: Index the links in $\mathcal{L}'$ in descending order by $\min_H E_o(l, H)$;
- 3: **while** $\mathcal{L}' \neq \emptyset$ **do**
- 4: Take the lowest index link $l_{max} \in \mathcal{L}'$;
- 5: Find the set of links $\mathcal{K} = \{l_{i,j} | l_{i,j} \in \mathcal{L}', l_{i,j} \text{ conflicts with } l_{max}\}$;
- 6: **for all** $H_{cur} \in \mathcal{H}$ **do**
- 7: Choose $l_{min} \in \mathcal{K}$, which satisfies $E_o(l_{min}, H_{cur}) \leq E_o(l, H_{cur}) \forall l \in \mathcal{K}$;
- 8: Compute $\check{E}'_o(\bar{P}, H_{cur})$;
- 9: **end for**
- 10: Find the channel $H_{min} | \check{E}'_o(\bar{P}, H_{min}) \leq \check{E}'_o(\bar{P}, H) \forall H$;

- 11: Choose $l_{min} \in \mathcal{K}$, which satisfies $E_o(l_{min}, H_{min}) \leq E_o(l, H_{min}) \forall l \in \mathcal{K}$;
- 12: Assign l_{max} and l_{min} to channel H_{min} ;
- 13: Update the estimate $\check{E}_o(\bar{P}, H_{min})$;
- 14: Remove l_{max} and l_{min} from $\mathcal{L}'$;
- 15: **end while**

The idea behind this algorithm is that we want to assign the links with worst-case (i.e. max-min) $E_o(l, H)$ as early as possible and to the channel for which it causes the smallest overall change in $E_o(\bar{P}, \bar{H})$, but rather than assigning the links one at a time, we pair each high E_o link with a link that has smaller E_o to balance out the effect on a channel's $E_o(\bar{P}, H)$. Otherwise, the high E_o link may appear in most or all of the MISs of that channel's conflict graph, resulting in high channel E_o . This algorithm provides a base channel assignment, which we then seek to improve in stage 3 of the heuristic.

4.4.5.3 Stage 3

Stage 3 is a local search procedure that takes the output from stage 2 and either moves high E_o links out of the max- E_o channel or moves low E_o links into the max- E_o channel. It may seem counterproductive to move links into the max- E_o channel, but there are two reasons why this may improve the channel's E_o . First, the channel E_o is an average over all MISs, so adding low E_o links may improve the average. Additionally, if the new link conflicts with one or more high- E_o links currently assigned to the channel, the effect will be to reduce the percentage of MISs that include the high- E_o links with which the newly added link conflicts. Since changing the channel assignment for a single link may result in a new channel having the max- E_o , channel assignments are changed for only one link at a time. The stage 3 algorithm is as follows:

- 1: Initialize $\check{E}_o(\bar{P}, H)$ with the estimates from stage 2;
- 2: **while** Able to improve by either adding or removing a link **OR** Reached iteration limit **do**

```

3:   Find  $H_{max} = \arg \max_H \check{E}_o(\bar{P}, H)$ ;
4:   Sort  $\mathcal{L}' = \{l_{ij} | H_{ij} = H_{max} \text{ in descending order};$ 
5:   while No improvement from removing link from channel  $H_{max}$  do
6:       Choose highest  $E_o$  link from  $\mathcal{L}'$ , call it  $l_{ij}$ ;
7:       Set  $H_{ij} \neq H_{max}$ ;
8:       Estimate  $\check{E}'_o(\bar{P}, H_{max})$ ;
9:        $H_{min} = H_{max}$ ;
10:    if  $\check{E}'_o(\bar{P}, H_{max}) \leq \check{E}_o(\bar{P}, H_{max})$  then
11:        for all  $H_{cur} \neq H_{max}$  do
12:            Set  $H_{ij} = H_{cur}$ ;
13:            Estimate  $\check{E}'_o(\bar{P}, H_{cur})$ ;
14:        end for
15:        if  $\check{E}'_o(\bar{P}, H) < \check{E}_o(\bar{P}, H_{max})$  for some  $H$  then
16:            Assign  $H_{ij}$  to the channel that has lowest  $\check{E}'_o(\bar{P}, H)$ ;
17:            Update  $\check{E}_o(\bar{P}, H)$  with  $\check{E}'_o(\bar{P}, H)$ ;
18:        end if
19:    end if
20: end while
21: if Unable to improve  $\check{E}_o(\bar{P}, \bar{H})$  by removing link from  $H_{max}$  then
22:    while No improvement from adding link to channel  $H_{max}$  do
23:        Find  $H_{max} = \arg \max_H \check{E}_o(\bar{P}, H)$ ;
24:        Sort  $\mathcal{L}' = \{l_{ij} | H_{ij} \neq H_{max}, E_o(l_{ij}, H_{max} < \check{E}_o(\bar{P}, H_{max}))\}$  in ascending order;
25:         $H_{cur} = H_{ij}$ ;
26:        Set  $H_{ij} = H_{max}$ ;
27:        Estimate  $\check{E}'_o(\bar{P}, H_{cur})$ ;
28:        Estimate  $\check{E}'_o(\bar{P}, H_{max})$ ;
29:        if  $\check{E}'_o(\bar{P}, H_{cur})$  and  $\check{E}'_o(\bar{P}, H_{max})$  less than  $\check{E}_o(\bar{P}, H_{max})$  then
30:            Assign  $H_{ij}$  to  $H_{max}$ ;

```

```

31:         Update  $\check{E}_o(\bar{P}, H)$  with  $\check{E}'_o(\bar{P}, H)$ ;
32:     end if
33: end while
34: end if
35: end while

```

As written, the algorithm terminates when no link can be moved into or out of channel H_{max} or after having reached a fixed number of iterations. The fixed iteration limit is included because the randomness of the estimation procedure makes it possible, though improbable, that the algorithm will not terminate otherwise. However, our experience in simulations is that the algorithm terminates in fewer iterations than there are links.

4.5 A Power-based Multi-channel Topology Control Problem

As discussed in Section 4.1, most work using the term multi-channel topology control does not allow both power and channel assignments to vary. The few algorithms that do allow both to vary are incompatible with the assumptions that we have made. Therefore, to the best of our knowledge, the existing literature does not provide a topology control objective or algorithm that we can directly compare with the problem we have presented. To address this issue, we begin with the work of Lloyd *et al.* [91] and one of the topology control objectives presented therein, namely connected min-total power.

Lloyd *et al.* assumed that power was assigned on a per-node basis and sought to minimize the total node transmit power, subject to a connectivity constraint. As we are focused on the potential interference of the secondary network on primary users, we assign power on a per-link basis so that nodes always transmit with the minimum power necessary to communicate with another node. Links are directional to allow channels to be assigned in a

more general way and to accommodate possible channel asymmetry. To be compatible with the assumptions in Section 4.2.2, our multi-channel adaptation of min-total power topology control assumes the following:

1. A directed graph $\mathcal{G}_{max} = (\mathcal{N}, \mathcal{L})$ is given.
2. Each $l_{ij} \in \mathcal{L}$ is assigned a power level, P_{ij} , as defined by (4.3), resulting in a power assignment vector $\bar{P}$.
3. Each non-zero power link, l_{ij} , is assigned a channel, H_{ij} , resulting in channel assignment vector $\bar{H}$. Any link may be assigned to any channel in the set of available channels, $\mathcal{H}$.
4. The total per-channel power of channel H is the sum of the P_{ij} for all l_{ij} assigned to H .

The min-max total per-channel power topology control problem, or **MinMaxP**, is to find the power and channel assignment vectors that satisfy

$$\begin{aligned} & \min_{\bar{P}, \bar{H}} \max_{H \in \mathcal{H}} \sum_{H_{ij}=H} P_{ij}, \\ \text{s.t. } & \mathcal{G}_{\bar{P}} \text{ is bi-directionally connected,} \end{aligned} \tag{4.17}$$

where $\mathcal{G}_{\bar{P}}$ and bi-directional connectivity are as defined in Section 4.2.2.

4.5.1 Solving MinMaxP

We believe that this multi-channel topology control problem is a natural extension of the min-total power problem when we are primarily focused on the interference caused by the secondary network. To our knowledge, this problem has not been previously researched, which means that there are no known algorithms for solving the problem. Our first concern is whether this problem can be solved in polynomial time. As previously stated, an optimal

solution for the single-channel case can be found in polynomial time by finding the MST. However, we are concerned with the cases where $|\mathcal{H}| \geq 2$. First, let us assume that $\bar{P}$ is fixed so that the problem reduces to

$$\min_{\bar{H}} \max_{H \in \mathcal{H}} \sum_{H_{ij}=H} P_{ij}. \quad (4.18)$$

We recognize (4.18) as similar to the objective of the minimum makespan problem on uniform processors, which is known to be NP-complete for 2 or more processors [39].

The minimum makespan problem on uniform processors, (**MinMake**), assumes that there is a set of k jobs with completion times $\{\tau_1, \tau_2, \dots, \tau_k\}$, where $\tau_i \in \mathbb{Q}^+$, and a set of processors $\{M_1, M_2, \dots, M_m\}$. Each job is assigned to one of the processors so that the time that it takes a particular processor to complete its jobs is just the sum of the completion times of all jobs assigned to that processor. The objective is to find a job assignment that minimizes the time it takes for the last processor to finish (i.e. the makespan).

The parallel between **MinMake** and the reduced version of **MinMaxP** is apparent when we equate job completion times and processors in **MinMake** to link transmit powers and channels in **MinMaxP**, respectively. In general, the number of jobs (or links) can be odd; however, to simplify the reduction from **MinMake** to **MinMaxP** for proving NP-hardness, we provide the following lemma, in which **MinMake** is restricted to an even number of jobs:

Lemma 4.5.1. ***MinMake** ($m > 2$) is NP-hard when the number of jobs, k , is restricted to even numbers.*

Proof. See Appendix A. □

This lemma supports the following theorem:

Theorem 4.5.2. *The multi-channel topology control problem **MinMaxP** is NP-hard for $|\mathcal{H}| \geq 2$.*

Proof. Given an instance of **MinMake** with a set of jobs $\{\tau_1, \tau_2, \dots, \tau_k\}$, $k = 2\kappa$, and a set of processors $\{M_1, M_2, \dots, M_m\}$, $m \geq 2$, the reduction of the problem instance to a **MinMaxP** problem instance is as follows. First, we choose the set of channels so that $|\mathcal{H}| = m$. Then, we arrange $\kappa + 1$ nodes in a star topology, with n_0 in the center. We choose $P_{\min} = \min_i \tau_i$, $P_{\max} = \max_i \tau_i$, and $\mu_c < P_{\min}$. Then we choose the channel gains as $g_{i,0} = \mu_c / \tau_i$, $i = \{1, 2, \dots, \kappa\}$, for the links terminating at n_0 and $g_{0,i} = \mu_c / \tau_{i+\kappa}$, $i = \{1, 2, \dots, \kappa\}$, for the links originating at n_0 . The resulting channel gains are guaranteed to be less than 1 because $\mu_c < \min_i \tau_i$, but they are also large enough that, for each link to/from n_0 , there is a power level $P_{ij} \in [P_{\min}, P_{\max}] \subset \mathbb{R}^+$ that satisfies $P_{ij}g_{ij} \geq \mu_c$.

For links that do not originate or terminate at n_0 , we choose the channel gains $g_{ij} < \mu_c / P_{\max}$, which ensures that for all l_{ij} , $i, j \neq 0$, there is no link transmit power that can satisfy $P_{ij}g_{ij} \geq \mu_c$. The net result is that $\mathcal{G}_{\max}$ consists only of the links between n_0 and the other nodes, with each link to/from n_0 having power $P_{ij} = \mu_c / g_{ij}$. This implies that there is a single bidirectionally connected topology, namely $\mathcal{G}_{\max}$. Therefore, the optimum solution to this instance of **MinMaxP** can be written as in (4.18). Since each link power in $\mathcal{G}_{\max}$ is equal to some τ_i , it is clear that a channel assignment that satisfies (4.18) also minimizes the makespan, and vice versa. $\square$

In the absence of any known algorithms that address **MinMaxP**, we have developed a heuristic based on the MST and the so-called longest processing time first (LPT-first) scheduling algorithm. The algorithm is as follows:

- 1: Find the MST of the undirected, power weighted graph;
- 2: Convert the undirected links to directed links;
- 3: Sort the links by power in descending order;
- 4: **while** Not all links have been assigned a channel **do**
- 5: Assign the highest power unassigned link to the channel with lowest total power;
- 6: Update the channel's total power;
- 7: **end while**

Graham [92] has shown that the LPT-first solution is within $\frac{4}{3} - \frac{1}{3|\mathcal{H}|}$ of optimal. However, we have observed in our simulations that the LPT-first solution is usually within 1% of the average total per-channel power. Since the average total per-channel power is a lower bound on the optimum solution, this implies that the LPT-first solution is usually within 1% of the optimum channel assignment.

Nevertheless, we cannot claim that the sequential heuristic for solving **MinMaxP** performs this close to optimum based on the strength of the channel assignment algorithm alone. Ideally, we would conduct a complete search to obtain the optimum for comparison to our heuristic; however, this is impractical for moderate sized networks. For example, a bi-directionally connected tree for a 20-node network has 38 links. If $|\mathcal{H}| = 3$ channels, then for a single tree there are on the order of 10^{18} channel assignment combinations. Such a network can easily have 10^{15} spanning trees, bringing the total number of multi-channel tree topologies to 10^{33} .

We have often referred to *the* optimal **MinMaxP** solution, but in general there may be more than one optimal solution. Also, this set of optimal solutions may contain topologies that have more links than a tree. We justify our focus on bi-directional trees with the following theorem.

Theorem 4.5.3. *The set of optimal solutions to **MinMaxP** contains at least one bi-directional tree.*

Proof. Suppose that we have an optimal solution, $(\bar{P}^*, \bar{H}^*)$, to **MinMaxP** that contains more links than a bi-directional tree. By assumption, the graph $\mathcal{G}_{\bar{P}^*}$ contains more links than are necessary to satisfy the bi-directional connectivity constraint. We can simply remove link pairs from $\mathcal{G}_{\bar{P}^*}$, thereby removing them from the channels to which they are assigned, until we are left with a bi-directional tree. The remaining multi-channel topology is still optimal since removing links cannot increase (4.17). $\square$

4.5.2 Metaheuristic for Evaluating MinMaxP

Since a complete search is impractical for network sizes of interest to us, we turn to another method that is widely used for obtaining good solutions to combinatorial problems, the genetic algorithm (GA). We use the GA simply for comparison purposes. We do not treat it as a competitor to the heuristic because the length of time we run the GA is approximately 4 orders of magnitude greater than what is required by the heuristic. Since it is easy to design a poorly performing GA, this section describes the GA that we have developed for obtaining comparison solutions.

The basic components of our GA are a population of candidate solutions (individuals), a method for evaluating candidate solutions, a crossover function, and a mutation function. We use a single population GA in which solutions consist of the vector pairs $(\bar{P}, \bar{H})$. Each solution is evaluated using $\max_{H \in \mathcal{H}} \sum_{H_{ij}=H} P_{ij}$. Crossover is a binary operation where two parent solutions are combined to produce a new feasible solution. The parent population is determined by calculating $E_o(\bar{P}, \bar{H})$ for each individual, taking a fixed number of individuals with the lowest $E_o(\bar{P}, \bar{H})$, denoted by β_{par} , and using the parent population to generate new individuals to replace those not in the parent population. Parents are chosen uniformly at random and crossover is performed. Due to the bi-directional connectivity constraint, crossover is performed by first taking the union of all links in both solutions, sorting the links according to their weight, and removing links (and their directional pair) that do not break network connectivity, starting with the highest weight link.

Once crossover has generated enough individuals to replenish the population, mutation is performed on the population with a fixed uniform probability, denoted by α_{mu} , excepting the individual with lowest $E_o(\bar{P}, \bar{H})$. Mutation consists of channel mutation followed by link mutation. For channel mutation, we select links uniformly with fixed probability α_{ch} and change its channel to a new randomly selected channel. For link mutation, we select a random number of links that are not included in the individual's topology, sort all links by power level, and remove link pairs that do not break the connectivity constraint, starting

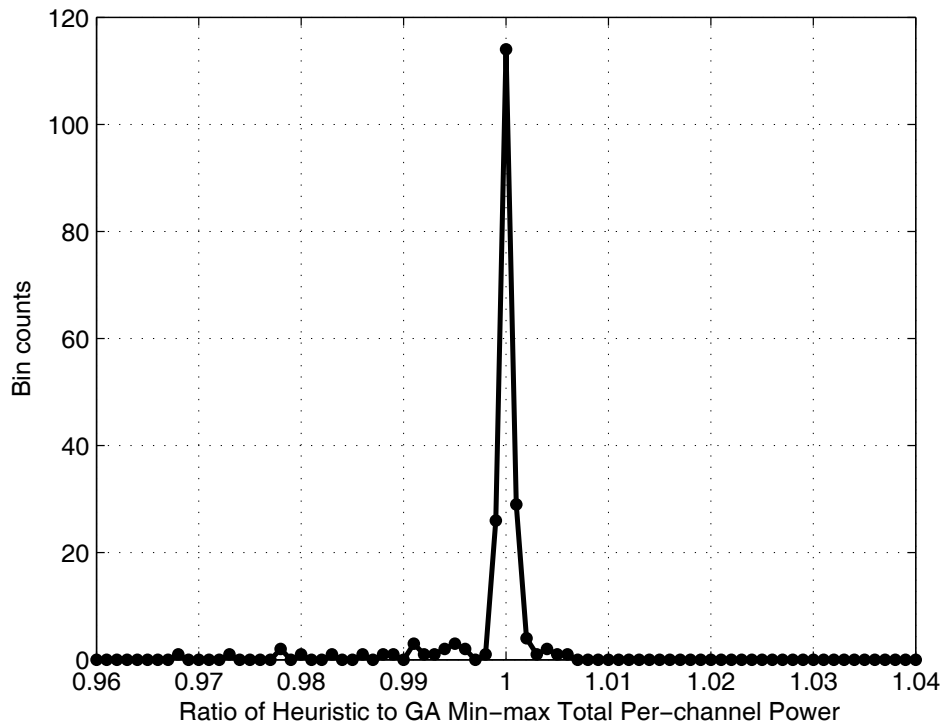
with the highest power link pair, until we are again left with a tree.

To allow the GA to run for an extended period of time, we implement a re-initialization method that is based on the change in the lowest $E_o(\bar{P}, \bar{H})$ for the population from one iteration to the next. If, after a fixed number of iterations β_{st} , the lowest $E_o(\bar{P}, \bar{H})$ in the population has not changed, we compare the current population's best solution with the best known solution since starting the GA, replacing the best known if the current population improves upon it, and then we regenerate the entire population. This method prevents wasted generations due to convergence of the population to a small number of solutions.

4.5.3 Performance of the MinMaxP Heuristic vs. the GA

We have simulated the **MinMaxP** heuristic for 4-channel, 25-node networks with nodes placed uniformly at random on a 1km^2 square with path loss parameters ($\sigma=7.1, \gamma=3.8$), and $d_o = 100\text{m}$. The communication threshold is $\mu_c = -97\text{dBm}$, which is based on the thermal noise floor for a 1 MHz channel, an 8 dB receiver noise figure, and a 9 dB SNR. The GA has a population size of 50 with $\beta_{par} = 25$, $\alpha_{mu} = 0.2$, $\alpha_{ch} = 0.2$, and $\beta_{st} = 40$. The GA runs for 100,000 generations.

Fig. 4.2 shows a histogram of the ratio of the maximum per-channel power achieved by the heuristic in Section 4.5.1 versus the maximum power of the best solution discovered by the GA for 200 network scenarios. We display the results as histograms because of the random nature of network generation. We see from the figure that the heuristic's performance is nearly identical to that of the GA, falling within 1% of the GA performance for most cases and bettering the GA by a few percent in a few cases. To provide a frame of reference, we have also simulated the performance of an algorithm that chooses solutions at random. This algorithm starts with all feasible links in the network and removes directional link pairs randomly, subject to the condition that connectivity be maintained, until it is left with a tree. Channels are then randomly assigned to these links. The histogram of the ratio of this algorithm's performance to that of the GA is shown in Fig. 4.3. The difference in

Figure 4.2: Performance of **MinMaxP** Heuristic

performance is readily apparent, with the randomly selected solution having on average more than 30 times worse performance than the heuristic.

4.6 Performance Comparison

4.6.1 Single-channel Heuristic Performance

For performance evaluation, we compare the results of our heuristic with those obtained using two other single-channel topology control objectives: the min-total power objective, in which the goal is to minimize the total transmission power of the nodes [71], and the minimum interference objective, in which the goal is to minimize a particular definition of interference [93]. In both cases, we must adapt the original algorithms, though for different

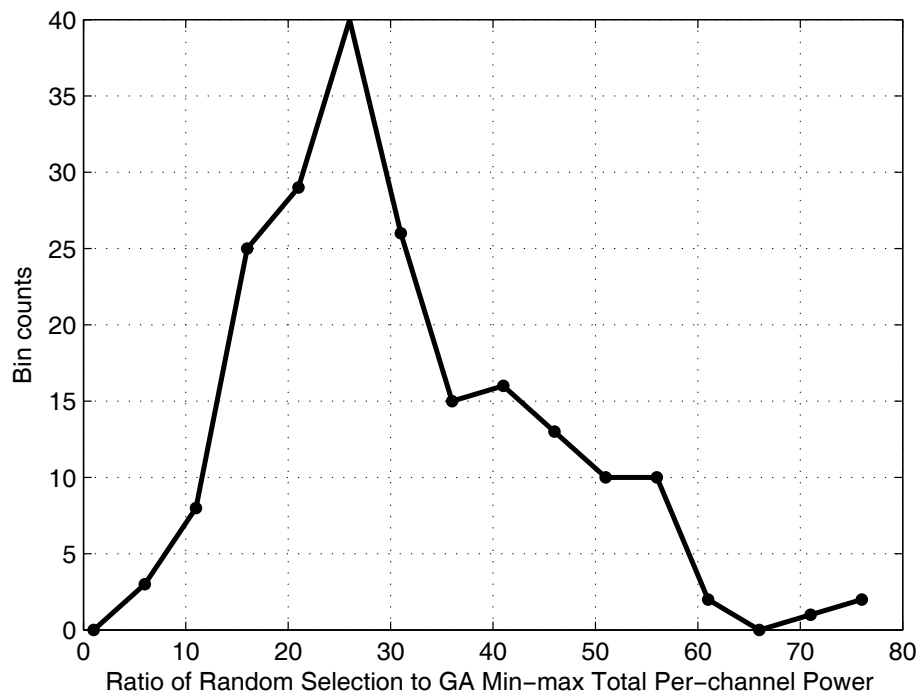


Figure 4.3: Performance of Randomly Selected Solution

reasons.

Typically, min-total power topology control assigns a single power level to a node, as opposed to assigning power on a per-link basis. To provide a fair comparison of min-total power topology control with our algorithm, we simply assume that power is assigned on a per-link basis, with the objective being to minimize the sum of powers assigned to links. This is easily solved by a link-power-weighted MST.

The LIFE algorithm developed by Burkhart et al. [93] finds the minimum interference topology. They define the interference of a graph to be the maximum coverage of the links in a graph, where the coverage is defined as

$$\begin{aligned} Cov(l_{ij}) := & |\{k \in \mathcal{N} : k \text{ covered by } D(i, |l_{ij}|)\} \cup \\ & \{k \in \mathcal{N} : k \text{ covered by } D(j, |l_{ij}|)\}|, \end{aligned} \quad (4.19)$$

with $|l_{ij}|$ being the distance between nodes i and j , $D(i, |l_{ij}|)$ the disk of radius $|l_{ij}|$ centered at node i , and k covered by $D(j, |l_{ij}|)$ meaning that node k falls within $D(i, |l_{ij}|)$.

The reason why we feel the need to adapt (4.19) is because it is implicitly based on a deterministic path loss model. Since we use a stochastic path loss model, we propose the following modification to (4.19) for a given link power vector $\bar{P}$:

$$\begin{aligned} Cov_{st}(l_{ij}) := & |\{k \in \mathcal{N} : P_{ij} * g_{ik} \geq \mu_c\} \cup \\ & \{k \in \mathcal{N} : P_{ji} * g_{jk} \geq \mu_c\}|. \end{aligned} \quad (4.20)$$

Since the idea behind (4.19) is to quantify the minimum number of nodes that will be affected by transmissions on either l_{ij} or l_{ji} , we believe that (4.20) is an appropriate modification for comparison purposes. The network-level interference measure is the same as in [93], with $Cov(l_{ij})$ replaced by $Cov_{st}(l_{ij})$. The minimum interference topology control objective is then

to find

$$\bar{P}^* = \arg \min_{\bar{P} \in \mathcal{P}} \max_{l_{ij} \in \mathcal{L}_{\bar{P}}} Cov_{st}(l_{ij}). \quad (4.21)$$

4.6.1.1 Single-channel Simulation Parameters

For simulations purposes, a fixed number of nodes is placed uniformly at random within a 1 km² square. The ROI extends this square by 500 m on each side for a total area of 4 km². The channel gains between nodes are randomly generated using (4.1) with $\sigma = 7.1$, $\gamma = 3.8$, $[\lambda_o]_{dB} = -71.9$ and $d_o = 100$ m [89], with this same set of parameters being used to calculate outage probabilities. We assumed a receiver noise floor at -106 dBm, which is the thermal noise floor when assuming a 1 MHz channel bandwidth and an 8 dB receiver noise figure. Additionally, we use the values $\mu_c = -97$ dBm, $\mu_d = -110$ dBm, and $\mu_{out} = -93$ dBm.

Key inputs for our simulations are the set of w_i 's capturing outage potential, as well as the tile density. With regard to tile density, we space the tile centers by 50 m so that outage probability at the center of a tile is used to approximate outage probability as much as $25\sqrt{2}$ m away. As for the outage potential map, we randomly generate a spatial delta-like map in which most tiles are given a w_i of 0.1 while a few tiles, chosen at random, are given a w_i of 10. While simulations show that there is benefit to using our topology control objective in situations where nothing is known about the primary user (i.e. all w_i are the same), forcing some variation in the w_i reflects a scenario where the secondary network has some knowledge of the history of primary user tendencies. It is not intended as a representative sample of how an outage potential map would actually look, as this is the subject of current research.

4.6.1.2 Single-channel Simulation Results

Due to the difficulty with obtaining the optimal solution to (4.9) for networks with any sizable number of nodes, we are only able to compare our algorithm against optimal results for 10-node networks. Figure 4.4 plots the average $E_o(\bar{P})$ (not the estimate $\ddot{E}_o$) over 10 runs

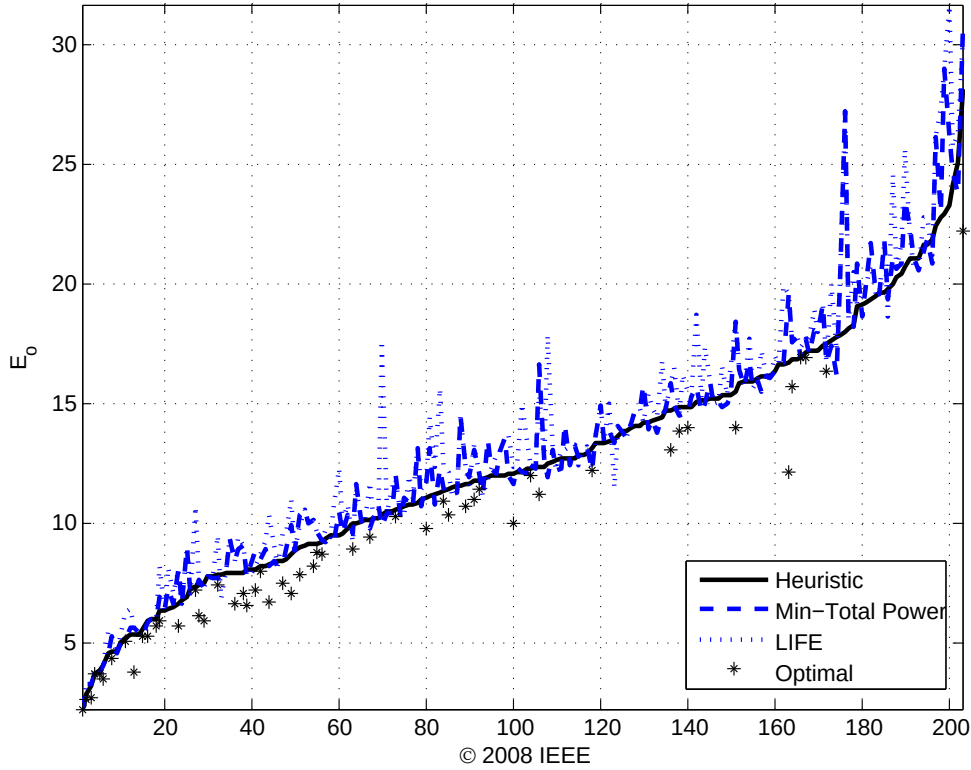


Figure 4.4: Performance Comparison for 10 Node Networks ©2008 IEEE. Reprinted, with permission, from Friend *et al.*, “Environmentally-friendly secondary network topology control for minimizing outage potential,” in *IEEE DySPAN*, (Chicago, IL), pp. 178–185, October 2008.

of the heuristic against the $E_o(\bar{P})$ for the min-total power topology and for the topology obtained from the LIFE algorithm. The asterisks are the optimal $E_o(\bar{P}^*)$ of a sampling of about a quarter of the randomly generated networks. In general, the performance of the heuristic is better than the two topology control algorithms. Note that the shape of the heuristic’s E_o plot appears smooth simply because the results were sorted using these values as the key.

The differences in performance seen in Figure 4.4 is quite small; however, the results from this particular set of simulations should not be interpreted as absolute for two main reasons. First, the outage potential map has a significant effect on the performance difference. Secondly, when there are only 10 nodes in the network, the degrees of freedom available for optimizing

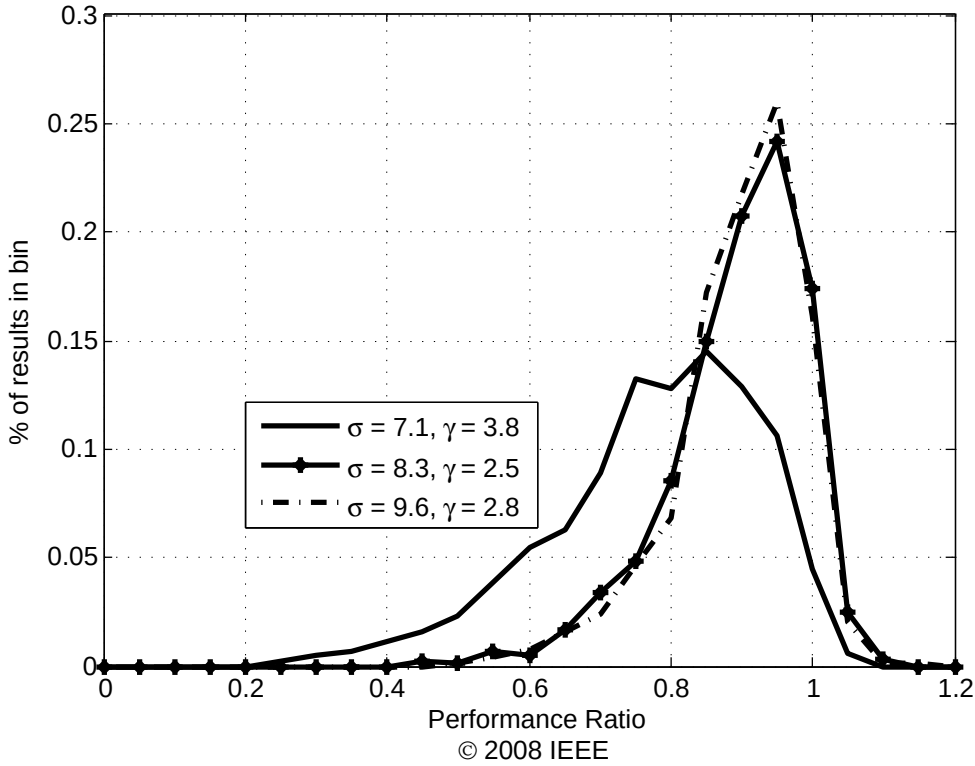


Figure 4.5: Histograms of the Ratio of Heuristic Performance vs. Min-total Power Topology Control ©2008 IEEE. Reprinted, with permission, from Friend *et al.*, “Environmentally-friendly secondary network topology control for minimizing outage potential,” in *IEEE DySPAN*, (Chicago, IL), pp. 178–185, October 2008.

the topology are limited. As the number of nodes increases, we begin to see the effect of the increase in degrees of freedom.

Both of these factors are highlighted in Figures 4.5 and 4.6, with Figure 4.5 comparing the heuristic against the min-total power topology and Figure 4.6 comparing the heuristic against the min-interference topology. For these simulations, we have increased the values of the random deltas in the outage potential map by a factor of 10, increased the number of deltas to 20 (about 1% of the tiles), and increased the number of nodes to 20. All other simulation parameters are the same as for Figure 4.4. The results are plotted as histograms of the ratio of the (actual) E_o of our heuristic to the E_o of the comparison topology control algorithm. Each

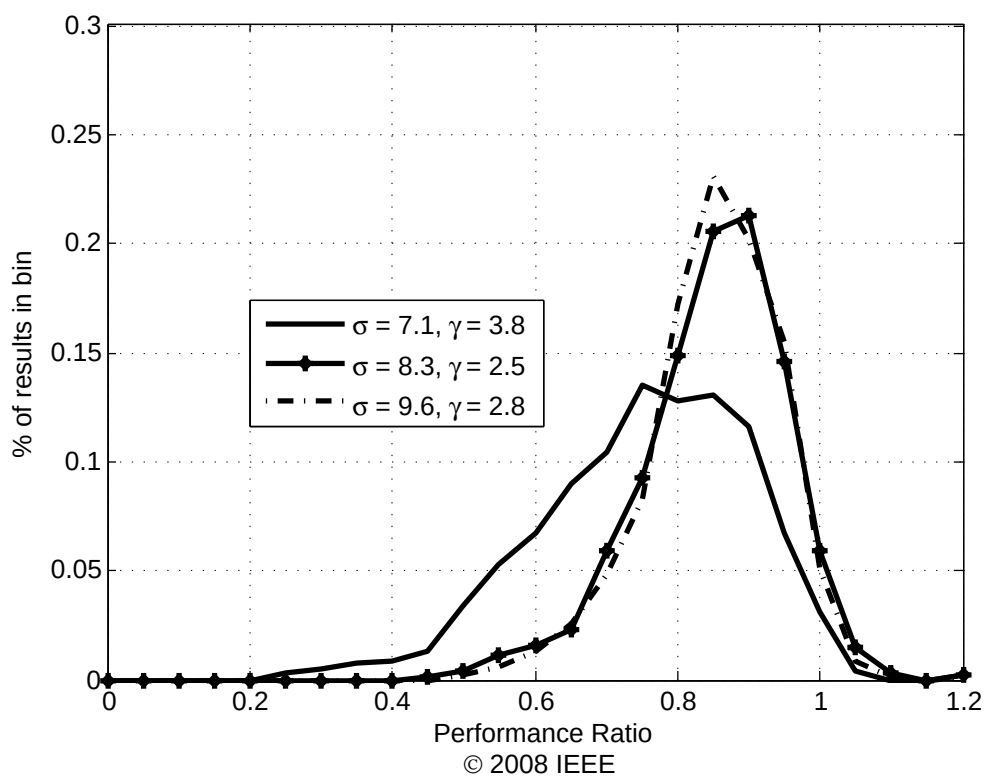


Figure 4.6: Histograms of the Ratio of Heuristic Performance vs. LIFE ©2008 IEEE. Reprinted, with permission, from Friend *et al.*, “Environmentally-friendly secondary network topology control for minimizing outage potential,” in *IEEE DySPAN*, (Chicago, IL), pp. 178–185, October 2008.

figure presents the results for three sets of lognormal shadowing parameters taken from [89]. Since it is impractical to perform a complete search to find $E_o(\bar{P}^*)$ (using Kirchhoff's matrix tree theorem, we find that just the number of spanning trees is on the order of 10^{16} for these networks), we are unable to compare against the optimal topology for these networks. In fact, just evaluating the performance of a single topology (i.e. computing $E_o(\bar{P})$) quickly becomes infeasible due to exponential growth in the number of MISs. Therefore, although the heuristic, min-total, and LIFE algorithms run quickly, exact computation of $E_o(\bar{P})$ for the thousands of topologies necessary to produce histograms limits the size of the network for which we can provide these results.

We have chosen to display the results in Figures 4.5 and 4.6 as histograms because of the random nature of the heuristic. Values less than 1 on the horizontal axes of Figures 4.5 and 4.6 indicate that the heuristic performs better than the comparison topology control algorithm. For example, a ratio of 0.8 means that the heuristic's performance was 20% better than the comparison algorithm. We see from the figures that our heuristic generally outperforms the other topology control algorithms, with the peak performance gain being 77% (a ratio of 0.23). Our heuristic outperforms the topology control algorithms between 91.1 and 98.6% of the time, and when the heuristic performs worse than a topology control algorithm, it is usually by less than 10%.

It is also worth mentioning that our heuristic may produce a topology with more links than a spanning tree, whereas both the min-total power and the LIFE topologies always produce spanning trees. In general, tree-based topologies increase the length of routes in the network and are less robust due to the fact that loss of a single link results in a disconnected network. From 13 to 16% of the topologies found by our heuristic for the simulations in Figures 4.5 and 4.6 had from 1 to 3 more links than a spanning tree.

4.6.2 Multi-channel Heuristic Performance

In Section 4.5, we developed both a multi-channel topology control objective and a heuristic for solving it. The heuristic is not optimal, but its performance in comparison to that of a long-running GA is excellent. Since obtaining optimal solutions to **MinMaxEo** appears to be at least as difficult as obtaining optimal solutions to **MinMaxP**, we again develop a long-running GA to provide comparison solutions to evaluate the topology control heuristics. After detailing the GA, we provide our final results comparing the **MinMaxEo** heuristic, the **MinMaxP** heuristic, and a completely random multi-channel topology against the performance of the long-running GA.

4.6.2.1 Metaheuristic for Obtaining Comparison Solutions to MinMaxEo

The discussion in Section 4.5.1 regarding the difficulty with solving **MinMaxP** via complete search is directly applicable to solving **MinMaxEo** as well. Therefore, we face the same issue with comparing solutions obtained via the heuristic in Section 4.4.5 with optimal solutions. We again turn to a long-running genetic algorithm to provide comparison solutions.

We repeat much of the structure of the GA in Section 4.5.2 with modifications to crossover and mutation, and, of course, using the objective function given in (4.9). The change in crossover is in the sorting of links after combining links from both parents. Rather than use link transmit power as the sort key, we sort links in descending order by $E_o(l_{ij}, H)$ so that crossover attempts to discard links with higher $E_o(l_{ij}, H)$ first.

Mutation has been changed more extensively due to the complex relationship between link selection (and the resulting conflict graphs) and channel assignment. Whereas the effect of addition or removal of a link in **MinMaxP** is somewhat neutralized when the link's power is small compared to the total power of all other links in the channel, the same property does not generally hold for **MinMaxEo** because adding or removing a single link may have a dramatic effect on the set of MISs for a particular channel. Changing the channel assignment

of a link similarly impacts the set of MISs for the old and new channels.

We have found that when link and channel mutation occur completely randomly, the resulting solutions are generally much worse than the original. In fact, any time that link and channel mutation occur separately, the result is likely to be a much worse solution because the conflict graphs are sensitive to changes in either. Therefore, we have chosen to limit mutation to operate on only channel assignment. The concern that arises is whether there is sufficient diversity (i.e. exploration of diverse regions within the solution space) in the GA if we limit mutation to just channels. This concern is mitigated by the re-initialization procedure in which a completely new set of solutions is injected into the population after convergence has occurred.

There are two properties that we take advantage of in designing the channel mutation function. The first is that $E_o(\bar{P}, \bar{H})$ can only improve if links are either moved into or out of the channel $H_{max} = \arg \max_{\mathcal{H}} E_o(\bar{P}, H)$. The second property is that only links not assigned to H_{max} that satisfy $E_o(l_{ij}, H_{max}) < E_o(\bar{P}, H_{max})$ can potentially lower $E_o(\bar{P}, \bar{H})$. So, we first determine the set of links that are currently assigned to H_{max} and randomly reassign these links with probability α_{ch} to one of the other channels. Then we take the set of links not assigned to H_{max} that satisfy $E_o(l_{ij}, H_{max}) < E_o(\bar{P}, H_{max})$ and assign them to H_{max} with probability $\alpha_{ch}/(|\mathcal{H}| - 1)$. The scaling of α_{ch} by $(|\mathcal{H}| - 1)$ balances the movement of links in and out of H_{max} .

4.6.2.2 Performance of the Multi-channel Heuristic

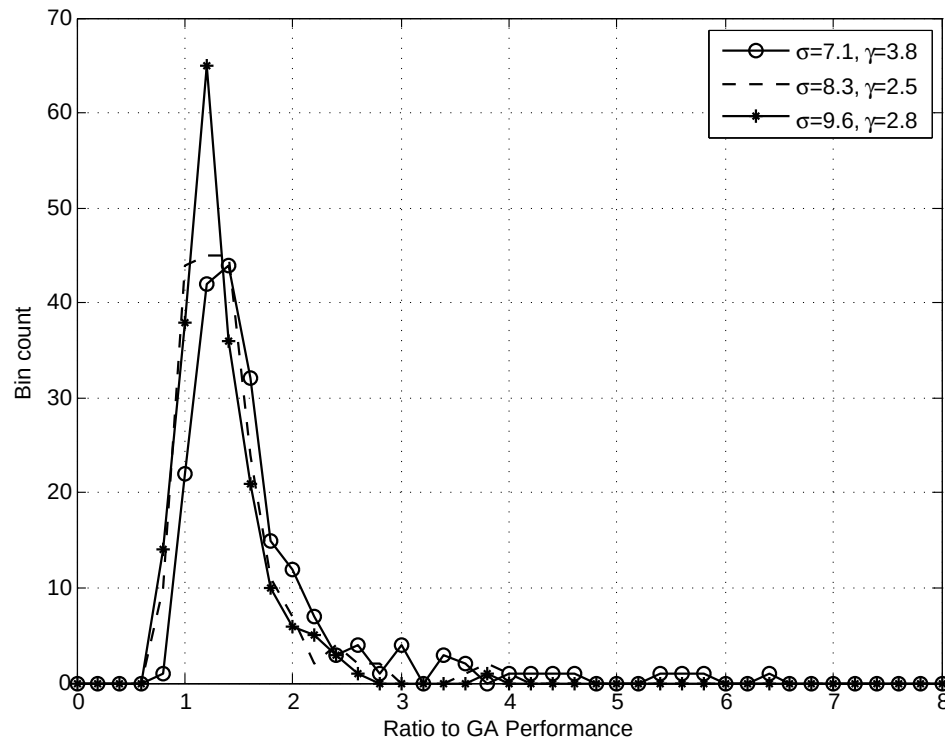
Having described the GA that we use for obtaining good solutions for **MinMaxEo** as well as the comparison multi-channel topology control algorithm, detailed in Section 4.5.1, we can compare these to our heuristic for solving **MinMaxEo**. The GA performance is intended to provide some idea of what an optimal solution may be, though we cannot claim that the GA actually finds the optimal solution. The **MinMaxP** heuristic is intended to show the difference in performance using a more conventional, power-focused multi-channel topology

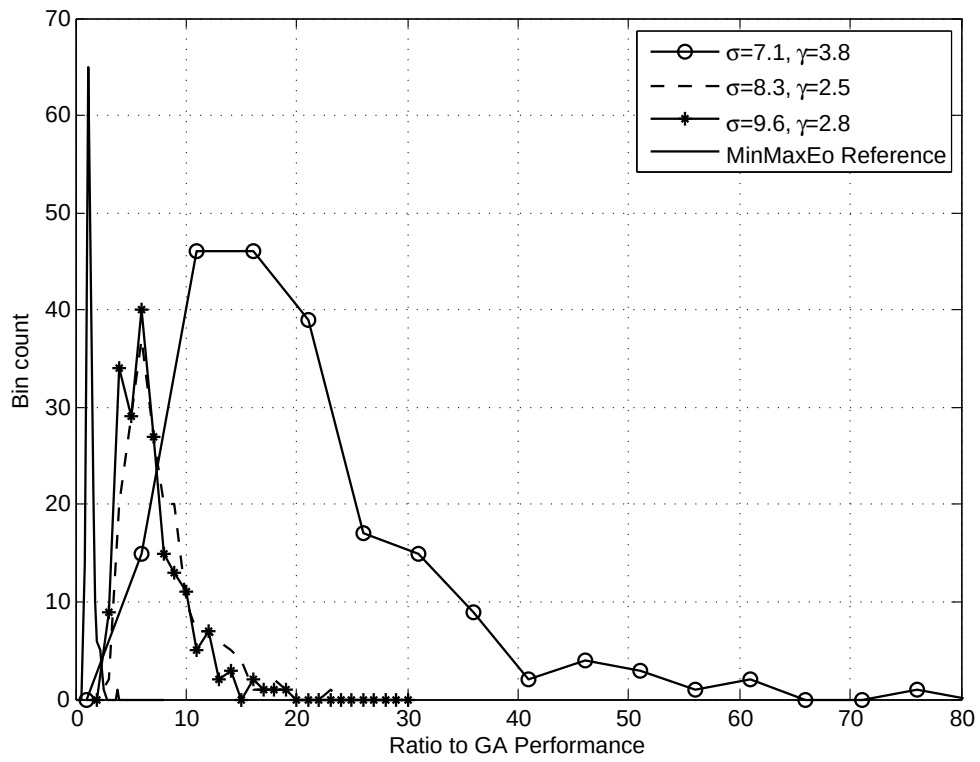
control algorithm. Additionally, we compare the performance of these algorithms against that of a randomly selected solution.

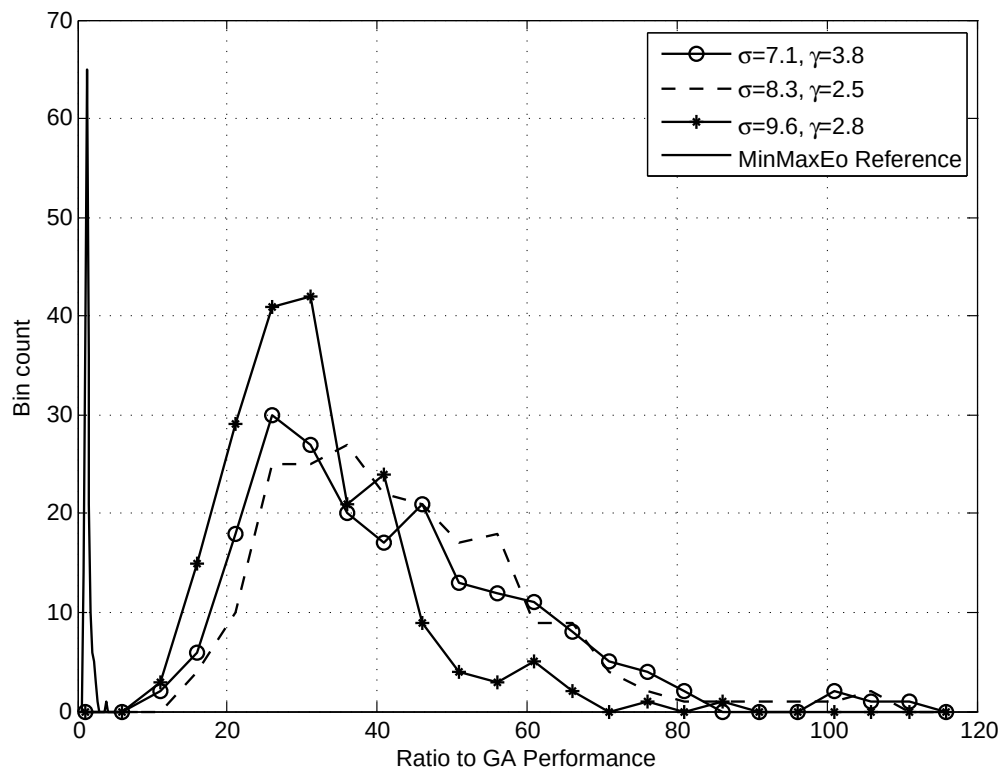
We generate networks by uniformly placing 25 nodes in a 1km^2 square. The ROI, $\mathcal{A}$, is taken to be the 4km^2 square centered on the square containing the network nodes. The ψ_i in $\mathcal{A}$ form a grid, with a spacing of 250 meters. Values of $w_i(H)$ are set to 0.01 except for 20 randomly located values per channel that are drawn randomly from the interval $[0, 1]$. For the path loss model parameters, we use $d_o = 100\text{m}$ with a reference loss of $\lambda_o = 71.9$ dB. We again choose the combinations $(\sigma=7.1, \gamma=3.8)$, $(\sigma=8.3, \gamma=2.5)$, and $(\sigma=9.6, \gamma=2.8)$ from [89]. The connectivity threshold is $\mu_c = -97$ dBm, and the detection threshold is $\mu_c = -110$ dBm. For computing outage probability, we assume an outage threshold of $\mu_{out} = -93$ dBm. The transmit power limits are $P_{min} = -60$ dBm and $P_{max} = 0$ dBm.

Fig. 4.7 shows histograms of the performance of the **MinMaxEo** heuristic relative to the lowest E_o found by the GA, for three different lognormal fading parameter sets in [89]. A ratio of less than 1 indicates that the heuristic outperformed the GA. The performance of the **MinMaxEo** heuristic is within a factor of 2 of the GA performance 89% of the time, with the average performance being 48% worse than the GA.

Fig. 4.8 and 4.9 show histograms of the performance ratio of the **MinMaxP** heuristic and randomly selected solutions, respectively. The results for the **MinMaxEo** heuristic with parameter set $(\sigma=9.6, \gamma=2.8)$ are also plotted in Fig. 4.8 and 4.9 to provide a reference point for how much better the **MinMaxEo** heuristic performs. Fig. 4.8 shows that the **MinMaxEo** heuristic is much better than the **MinMaxP** heuristic, with up to an order of magnitude difference. Fig. 4.9 illustrates an even larger difference when compared against randomly selected solutions.

Figure 4.7: Performance of the **MinMaxEo** Heuristic

Figure 4.8: Performance of the **MinMaxP** Heuristic on **MinMaxEo**

Figure 4.9: Performance of Random Solutions on **MinMaxEo**

4.7 How This Problem Solution is a Cognitive Network

For most of this chapter, we have described a problem and its solution without reference to cognitive networks. This provides some level of reassurance that we are able to present the problem and its solution in a standalone manner, first as a problem of substance and value and secondly as a solution at which we arrive using methods that are generally familiar to networking researchers. However, as intended, the problem solution satisfies the key characteristics of a cognitive network.

Referring back to the definition in Chapter 2, we see that the key characteristics of a cognitive network are its ability to sense its environment, learn, plan, decide, and act in order to achieve an end-to-end goal. Our solution to the **MinMaxEo** problem incorporates the ability to sense the environment by the assumption that nodes are equipped to perform multi-channel spectrum sensing. The spectrum sensing results are used to estimate primary user locations and update an outage potential map, which is equivalent to learning about primary user behavior with knowledge stored in the outage potential maps. This learning is directly incorporated into the network objective, or end-to-end goal, which is to minimize the maximum per-channel expected outage potential.

The objective is forward-looking, which makes it akin to planning, since the outage potential map is primarily used in a predictive sense to adapt the topology in order to avoid outages that may occur in the future. The multi-channel heuristic is used to decide on a topology that satisfies the objective to within a reasonable amount. Our heuristic provides an excellent example of exploiting problem-specific details as opposed to using a more generic method such as the GA that we use for obtaining comparison solutions. The heuristic provides solutions in orders of magnitude less time than is required by the GA; however, the heuristic is unlikely to perform well on a slightly different objective function.

Chapter 5

Minimum Expected Cost Routing in Markovian MANETs ¹

In Chapter 4, our focus was on applying the cognitive network concept to an application that is widely associated with cognitive radio and cognitive radio networks. It is important to show how cognitive networking may be applied to the problem that dominates cognitive radio network research, as this is likely the first application to come to mind. However, in focusing solely on its application to the currently prevailing problem, we are apt to lose sight of the broader uses for cognitive networks. In this vein, it is helpful to demonstrate the versatility of cognitive networks by selecting an application that differs substantially from DSA. With this in mind, this chapter focuses on the elusive problem of optimal routing in a MANET.

Routing in MANETs has been of considerable interest to networking researchers since before wireless networking began its transformation from predominantly point-to-multipoint configurations to mesh and ad hoc configurations. While the theoretical foundation for optimal routing in static networks is well grounded in dynamic programming, MANET research has not converged upon a single link-level model to enable analysis of optimality, with much

¹This chapter is based on the work in [94, 95].

research being performed without any link-level model at all. Instead, the focus has been on simulation, using mobility models at the physical level to drive changes in network structure. While such an approach is not unwarranted, demonstrating the optimality of routing protocols when using one of the common physical-level mobility models would appear to be intractable. Additionally, the assumption that a path must be instantaneously connected has carried over from the static network domain, though it is not a strict requirement for packet delivery, as evidenced by a delay-tolerant network (DTN) [96]. Since MANETs may exhibit frequent changes in link status, requiring paths to be static for the lifetime of a packet may lead to over-engineering of the network, if the problem is solvable at all.

In this chapter, we address the problem of optimal routing in a MANET by first establishing a link-level model in which the network is a Markovian sequence of random graphs. Our approach allows us to dismiss the requirement of instantaneous connectivity and provides the context for discussing a probabilistic measure of routing optimality – *minimum expected total cost*. We show how obtaining minimum expected total cost routing policies for such stochastic networks corresponds to solving a MDP. We show the existence of an optimal, stationary, deterministic policy with finite expected total cost and investigate structural properties of optimal solutions for certain classes of Markovian networks. We analyze the convergence behavior of the MDP’s value function for linear networks, providing exact solutions and bounds for certain special cases, and we prove a localization result for a more general class of Markovian networks. We then analyze the effect of delayed state information, showing that the delay results in an increase in the already exponential rate at which the value function converges.

Since the requirement for global instantaneous state information generally cannot be satisfied by real networks, the value function obtained from the optimal policy for the MDP serves as an upper bound on how well we can perform in a Markovian network. In actuality, nodes only have partial state information, meaning that the best they can do is solve a partially observable Markov decision problem (POMDP). We describe how the optimal routing problem in the realistic case corresponds to the description of a DEC-MDP given by

Bernstein *et al.* [97].

5.1 Related Work

A tremendous amount of research has been done in routing for MANETs and many surveys exist (see e.g. [98]). For most MANET routing protocols, a route is considered optimal if it is the shortest path between source and destination, given the current state of the network. The shortest path concept may refer to the least number of hops, the shortest spatial distance (as in position-based routing [99]), or to the path that minimizes some other combination of the individual costs of the links in the path. However, such an optimality criterion fails to accommodate networks in which there either may not be an instantaneously connected path to the destination or in which the initial path may become disconnected while the packet is en route.

Since link states are constantly changing in MANETs, evaluating the performance of MANET routing protocols has mostly been relegated to simulation. This is because routing protocol optimality depends on the way that links change with time, which requires a link-level model of the network dynamics. More recently, researchers in DTN have investigated routing optimality for DTNs by first establishing a link-level model for describing how link state changes with time. A general overview of various approaches is given by Zhang [100], with one of the most notable approaches being the deterministic evolving graphs framework of Ferreira [101].

Altman [102] has identified many problems in networking that have been cast as Markov decision problems. Most of these applications are to wired networks. More recently, Wang and Wang [103] proposed a method employing the MDP approach to routing in wireless sensor networks; however, the emphasis is on learning an unknown reward function rather than finding optimal routing policies in randomly varying networks. The DEC-MDP was originally defined by Bernstein *et al.* [97]. It is the subject of current research in artifi-

cial intelligence, with some applications being interruption management in human-computer interactions [104] and learning the meaning of messages exchanged by agents [105].

5.2 Markovian Network Processes

Rather than treating a MANET as quasi-static, we may model the progression of network states as a stationary, ergodic Markov process. We assume that time is discrete, and we represent the instantaneous connectivity of the network with an undirected graph, $\mathcal{G}_t = (\mathcal{N}, \mathcal{L}_t)$, which is a random variable at each point in time t . The complete network process, or random sequence of graphs, from $t = 1$ to ∞ , denoted by $\{\mathcal{G}_t\}$, is assumed to possess the Markov property, i.e.

$$P(\mathcal{G}_t | \mathcal{G}_{t-1}, \mathcal{G}_{t-2}, \dots) = P(\mathcal{G}_t | \mathcal{G}_{t-1}).$$

Each graph, $\mathcal{G}_t$, is a subgraph of the *ergodic communications graph* $\mathcal{G}_e = (\mathcal{N}, \mathcal{L}_e)$, which consists of all $|\mathcal{N}| = N$ nodes in the network and all $|\mathcal{L}_e| = K$ links that have non-zero probability of being in the up state at some point in time. The concept of an ergodic communications graph is similar to the long-time expected communication network described by Porfiri *et al.* [106].

Since the set of nodes, $\mathcal{N}$, is fixed for all $\mathcal{G}_t$, we consider only the variation in link states when defining the support set of $\{\mathcal{G}_t\}$. This support set is also referred to as the network states. The network states are vectors composed of individual link states

$$L = [l_1 \ l_2 \ \dots \ l_K] \tag{5.1}$$

where l_i is the binary state of link i .

5.3 Routing as a Markov Decision Problem

Under the model described in section 5.2, there may not be a connected path between the source and destination nodes in $\mathcal{G}_t$. Even if there is a connected path at time t , a link on this path may transition to the down state by the time the packet arrives at the link, forcing the packet to either wait or be forwarded along a different path. This means that if we were to choose a particular path for the packet, regardless of the present state of the network, the number of time steps it would take for the packet to reach the destination is a random variable. Thus, the use of minimum cost as the measure of optimality for routing in stochastic networks is no longer meaningful because the cost is also a random variable. A more fitting measure of routing optimality in stochastic networks is *minimum expected total cost*.

Using minimum expected total cost requires that we define some additional parameters and functions. We first define the set of system states, $s \in \mathcal{S}$, which combine the packet location with the network state. System state s is the two-tuple

$$s = \begin{cases} (n, L) & \text{for } n \neq n_d \\ (n) & \text{for } n = n_d \end{cases} \quad (5.2)$$

where L is the network state (i.e. the binary vector of link states), n is the node at which the packet is located when the system is in state s , and n_d is the destination node. Therefore, there are $M = (N - 1) * 2^K + 1$ system states. The destination node does not need to be associated with any link states because in this system state, the packet has been delivered and link states have become irrelevant.

The action sets are subsets of the set of all actions, $\mathcal{A}$, with each actions set $\mathcal{A}_s \subset \mathcal{A}$ containing the set of allowable actions given that the process is in state s . The allowable actions are to either forward the packet along one of the adjacent links that is up or to hold the packet for one time step. Thus, the action sets are state-dependent and do not

allow multiple copies of a packet to be forwarded. A decision rule, $d(s) = a$, is a function (potentially a random function) that maps states to actions. A policy, $\pi = \{d_1, d_2, d_3, \dots\}$, where d_i is the decision rule to be used at time step i , is a sequence of decision rules. For the special case when $d_1 = d_2 = d_3 = \dots$, the policy is called stationary and denoted by d^∞ . For a stationary policy, we can write the system state transition probability matrix, T_S , in the following manner:

$$T_S = \begin{bmatrix} p_{1|1}^d & p_{2|1}^d & \cdots & p_{M|1}^d \\ p_{1|2}^d & p_{2|2}^d & \cdots & p_{M|2}^d \\ \vdots & \vdots & \ddots & \vdots \\ p_{1|M-1}^d & p_{2|M-1}^d & \cdots & p_{M|M-1}^d \\ 0 & 0 & \cdots & 1 \end{bmatrix}, \quad (5.3)$$

where $p_{j|i}^d$ is the probability that the system will transition to state s_j given that it is currently in state s_i and action $d(s_i)$ is taken, and the finite set of system states is indexed by the set of integers from 1 to M , with s_M being the destination (terminal) state.

The cost function, $c(s, a)$, maps $s \in \mathcal{S}$ and $a \in \mathcal{A}_s$ to a non-negative real-valued cost. The cost may represent hops, time delay, bandwidth, energy, or some other type or combination of resources. Cost is a function of the chosen action as well as the system state because the action determines which link will be traversed. Therefore, different actions when in state s may incur different costs. As is the case with static networks, we assume that the cost depends only on the link over which the packet is forwarded or on the cost of holding the packet when this action is chosen. Therefore, the cost function only depends on the action taken, with the system state determining the allowable set of actions.

We now define two random processes: $\{X_t\}$ is the sequence of system states, and $\{Y_t\} = \{d_t(X_t)\}$ is the sequence of actions, which is a random process because it is a (deterministic or random) function of a random process. Because the standard MDP description assumes

that the objective is to maximize the expected reward, we define the reward function

$$r(s, a) = -c(s, a). \quad (5.4)$$

5.3.1 Selecting an Appropriate MDP Type

The MDP framework can take on a variety of forms. For example, the expected cost may be over a finite period of time (finite-horizon) or for all time (infinite-horizon). Finite-horizon MDPs are closely related to stochastic dynamic programming, which means that the multi-stage problem can be reduced to a sequence of simpler single-stage problems [107]. However, finite-horizon MDPs generally produce non-stationary policies. This is undesirable because it requires nodes to track time and change their decision rule with time. We selected an infinite-horizon MDP because it allows the use of stationary policies and because it matches the scenario where there is no fixed limit on the amount of time that the packet is allowed take in transit to the destination.

In addition to the infinite vs. finite horizon choice, an MDP may use a discounted or total reward model. In the discounted reward MDP, a discount factor, $0 < \lambda < 1$, is applied to ensure convergence of infinite sums. We have chosen to use the expected total reward criteria because we see no justification for discounting in the routing problem. The expected total reward of an infinite-horizon MDP (also referred to as the *value function*) is given by (Equation 5.1.1 in [107])

$$v^\pi(s) = \lim_{T \rightarrow \infty} E_s^\pi \left[\sum_{t=1}^T r(X_t, Y_t) \right], \quad (5.5)$$

where E_s^π indicates that the expectation is evaluated using the transition probabilities resulting from policy π and that the system state is initially s , i.e. $X_1 = s$. We refer to the optimal policy, i.e. the policy that achieves the maximum expected total reward for all states, as π^* .

5.3.2 Characteristics of an Optimal Policy

The limit in (5.5) is not guaranteed to exist for arbitrary π and $r(s, a)$. However, we note that our MDP satisfies the requirements for being a *negative* model, meaning that $r(s, a) \leq 0$ for all $a \in \mathcal{A}_s$ and $s \in \mathcal{S}$. The limit in (5.5) is guaranteed to converge for the negative model. Additionally, because $\mathcal{A}$ and $\mathcal{S}$ are finite, an optimal stationary and deterministic policy exists [107]. However, in order for the expected total reward criteria to be meaningful, we must show that it is finite for the optimal stationary deterministic policy. This is established by the following theorem:

Theorem 5.3.1. *For an infinite horizon MDP with system state transition matrix, $T_{\mathcal{S}}$, of the form of (5.3) and for which $|r(s, a)| < \infty$ for all $s \in \mathcal{S}, a \in \mathcal{A}$ and $r(s_M, a) = 0$, there exists a stationary policy π for which $v^\pi(s) > -\infty$ for all $s \in \mathcal{S}$.*

Proof. Since $\mathcal{G}_e$ is assumed to be connected, we know that there exists a path from any node to the destination node, n_d . Suppose we choose a particular path in $\mathcal{G}_e$ that connects the source node to the destination node. Then we then select the stationary policy, π' , in which a node holds the packet until the next link along the path is up. Because the network process is ergodic, any link in $\mathcal{G}_e$ will eventually be in the up state with probability 1. Therefore, the process will exit every $s \neq s_M$ with probability 1 and then never return, meaning that every $s \neq s_M$ is transient. Once the packet reaches the destination, or equivalently state s_M , the system will remain in state s_M with probability 1. This means that s_M is an absorbing state, and that π' results in an *absorbing Markov chain* [108]. We can write $T_{\mathcal{S}}$ as

$$T_{\mathcal{S}} = \begin{bmatrix} Q & R \\ \bar{0} & 1 \end{bmatrix}$$

where Q is an $(M - 1)$ -by- $(M - 1)$ matrix, R is an $(M - 1)$ -by-1 column vector, and $\bar{0}$ is a row vector of $M - 1$ zeros. The expected number of time steps to reach the absorbing state from each transient state is given by the column vector $\tau = (I - Q)^{-1}\bar{1}$ [108], where

I is the $(M - 1)$ -by- $(M - 1)$ identity matrix, $\bar{1}$ is an $(M - 1)$ -by-1 column vector of ones, and each element of τ is the expected time to absorption given a particular starting state. For any absorbing Markov chain, $(I - Q)^{-1}$ exists, and each element of τ is therefore finite. Since, by assumption, the reward function is finite and $r(s_M, d(s_M)) = 0$, $v^{\pi'}(s) > -\infty$ for all $s \in \mathcal{S}$. $\square$

5.3.3 Finding an Optimal Policy

Now that we have described our routing problem as an MDP, we are able to apply standard methods for solving total reward infinite-horizon MDPs. The typical methods include policy iteration, value iteration, and linear programming. However, because we are using the total reward form of MDP, policy iteration and linear programming are not feasible [107]. Additionally, value iteration is not guaranteed to converge for a negative total reward infinite-horizon MDP unless we satisfy the following conditions [107]:

1. There exists a π for which $v^\pi(s) > -\infty$ for all $s \in \mathcal{S}$.
2. $\mathcal{S}$ is finite.

It is clear that condition 2 is satisfied, and condition 1 follows from Theorem 5.3.1. An important property of value iteration is that it converges monotonically to the value function of the optimal policy.

The key step in value iteration is updating the value function. Using $v^n(s)$ to represent the value function of state s at the n -th step of the algorithm, we define the function

$$\delta^n(s, a) = r(s, a) + \sum_{s_j \in \mathcal{S}} P(s_j \mid s, a) v^n(s_j). \quad (5.6)$$

At each iteration, the value function is updated by

$$v^{n+1}(s) = \max_{a \in \mathcal{A}_s} \delta^n(s, a). \quad (5.7)$$

Each step of value iteration requires $O(|\mathcal{A}||\mathcal{S}|^2)$ operations. Since $|\mathcal{S}|$ is exponential in the number of links, the solution to our MDP requires exponential time to compute; therefore, we consider the exponential size of $\mathcal{S}$ to be the key issue in solving the MDP.

5.3.4 The Suboptimality of Shortest-Path Routing in Markovian Networks

There are two dimensions, time and space, in which the network process exists. Thus, there may be spatial or temporal correlation between individual links in the network process. For example, suppose the network contains a satellite node while the other nodes are terrestrial. A pair of co-moving co-located terrestrial nodes will experience high correlation in their links to the satellite node, although the links may still experience random outages. We consider this to be a spatial relationship because it relates the behavior of links between different node pairs.

As an example of temporal correlation, consider a link that is spatially independent of all other links but is Markovian in time, i.e. $P(l_i[t+1] \mid l_i[t], l_i[t-1], \dots) = P(l_i[t+1] \mid l_i[t])$. In this case, the current state of the link is correlated with its past and future states but not with the state of any other link. We refer to this link as being spatially independent and temporally dependent (SITD). Another special case is the spatially and temporally independent (SITI) link, which has the property $P(l_i[t+1] \mid l_i[t], l_i[t-1], \dots) = P(l_i[t+1])$.

Thus far, we have focused on how we may model a stochastically-varying network and the methodology for obtaining optimal routing policies for such networks. However, existing link state-based MANET routing protocols use the shortest-path optimality criteria. This implicitly assumes that the network is static over the lifetime of the packet. To demonstrate the effect of using shortest-path routing in a Markovian network, we consider the example network in Fig. 5.1. All links are static except for those labeled SITI, and all SITI links have $p_U = p_D = 0.5$. There are C SITI links along the lower path and kC , $k > 2$, static

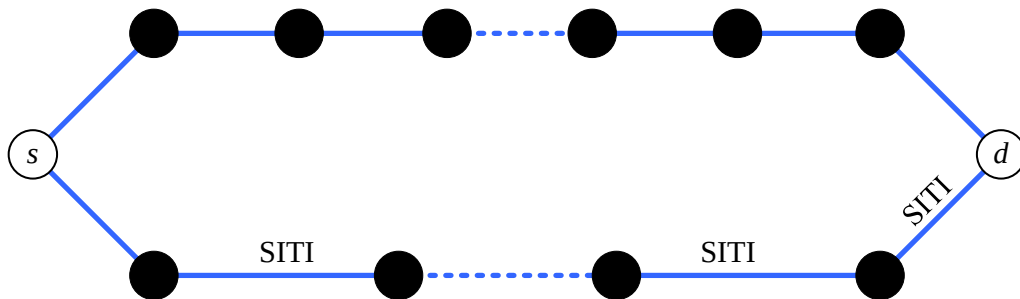


Figure 5.1: Example Markovian Network

links along the upper path.

Because the SITI links are only up with probability 0.5, the source node, labeled s , sees a connected lower path with probability 0.5^C . Using shortest-path routing, the source will choose the upper path with a probability approaching 1 for increasing C , with the packet requiring $kC + 1$ time slots to reach the destination. However, the expected number of time slots required to reach the destination along the lower path is $1 + 2C$. Since k may be arbitrarily large without affecting the probability that the source chooses the upper path, we see that shortest-path routing may result in the selection of an arbitrarily long route over one that has much shorter expected length.

5.4 Relating Distance and Link State Information

In this section, we take a closer look at the relationship between the distance between a link and a node making a decision and the variation in the value function over the possible states for the link. The motivation behind this is that maintaining link state information for all links in the network is costly. Therefore, we would prefer to establish some localization properties that limit the amount of link state information that each node must maintain.

5.4.1 Link Distance in SITI Networks

In general, it is necessary to include all links in the system state in order to find the optimal policy. This implies that the size of the MDP's state space grows exponentially in the number of links, and finding the optimal policy for large networks becomes impractical. However, the size of the state space can be reduced when there are one or more SITI links, for which we have the following theorem:

Theorem 5.4.1. *The state of SITI link l_i has no effect on $v^{\pi^*}(s)$, $s = (n, L)$, when l_i is not adjacent to n .*

Proof. We first define the sets $\mathcal{S}_{\mathbf{U}} = \{s \mid l_i = \mathbf{U}\}$ and $\mathcal{S}_{\mathbf{D}} = \{s \mid l_i = \mathbf{D}\}$. Let us choose an arbitrary pair of states, $s_{\mathbf{U}} \in \mathcal{S}_{\mathbf{U}}$ and $s_{\mathbf{D}} \in \mathcal{S}_{\mathbf{D}}$. We then note that $P(s_j \mid s_{\mathbf{U}}, a) = P(s_j \mid s_{\mathbf{D}}, a)$ because of the SITI property of l_i . This causes the summation in (5.6) to be the same for $s_{\mathbf{U}}$ and $s_{\mathbf{D}}$. We also note that $r(s_{\mathbf{U}}, a) = r(s_{\mathbf{D}}, a)$ because the action set is strictly dependent on links adjacent to n , which, by assumption, does not include l_i . Therefore, $\delta^n(s_{\mathbf{U}}, a) = \delta^n(s_{\mathbf{D}}, a)$. Additionally, the set of actions is the same for $s_{\mathbf{U}}$ and $s_{\mathbf{D}}$ because the actions depend solely on the states of links adjacent to n . Therefore, in Step 2 of value iteration, we always have $v^{n+1}(s_{\mathbf{U}}) = v^{n+1}(s_{\mathbf{D}})$. Since it is the same action that maximizes $v^{n+1}(s_{\mathbf{U}})$ and $v^{n+1}(s_{\mathbf{D}})$ and value iteration converges to the optimal solution, we conclude that the state of l_i does not affect the optimal policy. $\square$

The size of the MDP state space is generally $2^K(N-1)+1$. An important result of Theorem 5.4.1 is that the size of the MDP state space can be reduced by almost a factor of 2 for each SITI link. If, for example, there is a single SITI link, the size of the MDP state space can be reduced to $2^{K-1}(N-2) + 2^K + 1$. For the special case in which all links are SITI, the size of the state space is $\sum_{j \neq M} 2^{K_j} + 1$, where K_j is the number of links adjacent to node j .

5.4.2 Link Distance in SITD Networks

In Section 5.4.1, we showed that the influence of the state of SITI links is localized to the pair of nodes to which it is adjacent. Such localization has the benefit of reducing the size of the MDP state space. In this section, we focus on SITD links in an effort to determine if the system state space can be reduced when one or more links are classified as SITD.

To facilitate discussion of the effect of distance on optimal policy, we define the *minimum separation distance*, $D_{min}(n, l)$, which is the length (in hops) of the minimum length path among all paths in $\mathcal{G}_e$ originating at node n and terminating at one of the endpoints of link l . Intuitively, we would expect that the state of a particular link would have less impact on $v(s)$ as the minimum separation between the link and the node making the decision increases. This intuition is based on the behavior of stationary Markov processes. It is well known that the distribution of a Markov process approaches the stationary distribution as the time separation between the observation and the next state grows:

$$\lim_{k \rightarrow \infty} P(l_i[t+k] \mid l_i[t]) = P(l_i[t]). \quad (5.8)$$

5.4.2.1 Linear Networks

Our initial analysis is based on the linear class of networks that can be described by Fig. 5.2. For ease of exposition, we assume the latency (in time slots) reward function, which is

$$r(s, a) = \begin{cases} -1 & \text{for } s \neq s_M \\ 0 & \text{for } s = s_M \end{cases}$$

Given this reward function and the network in Fig. 5.2, it is easily seen that the optimal policy, π^* , is to forward the packet toward the destination whenever the adjacent link states permit such an action. Otherwise, the packet is held. We wish to find expressions for $v^{\pi^*}(s)$ for the states associated with the source node.

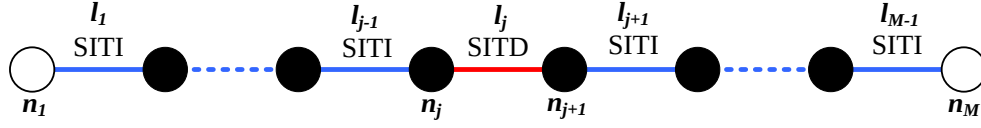


Figure 5.2: Linear Markovian Network

As was shown in Section 5.3.2, the limit in (5.5) is guaranteed to converge for our MDP, which allows us to write (5.5) as

$$v^{\pi^*}(s) = E_s^{\pi^*} \left[\sum_{t=1}^{\infty} r(X_t, Y_t) \right]. \quad (5.9)$$

When using the latency reward function, we can write (5.9) as

$$v^{\pi^*}(s) = - \sum_{\mathcal{X}} \mathbf{P}(\{X_t\} \mid s) w(\{X_t\}), \quad (5.10)$$

where $\{X_t\}$ is a random system state sequence terminating in the state s_M , $w(\{X_t\})$ is the number of time slots in $\{X_t\}$, and $\mathcal{X}$ is the set of all possible state sequences. The policy π^* is implicit in the probability distribution over $\{X_t\}$ in (5.10), as in the remainder of this chapter, except where it is necessary to be explicit.

Static Links A special case of Fig. 5.2 is when all link preceding n_j are static (i.e. up with probability 1). Let us refer to the probability that SITI link l_i is down as p_i , and, for the SITD link, $p = \mathbf{P}(l_j[t+1] = \mathbf{D} \mid l_j[t] = \mathbf{D})$ and $q = \mathbf{P}(l_j[t+1] = \mathbf{U} \mid l_j[t] = \mathbf{U})$. For this scenario, we have the following theorem.

Theorem 5.4.2. *When links l_1 through l_{j-1} are static, the minimum expected reward for any initial system state in which l_j is down ($s_{\mathbf{D}}$) is given by*

$$v_{stat}^{\pi^*}(s_{\mathbf{D}}) = - \left[p_{\mathbf{U}|\mathbf{D}}^{j-1} + \frac{2-p}{1-p} p_{\mathbf{D}|\mathbf{D}}^{j-1} \right] - (j-1) - \sum_{i=j+1}^{M-1} \frac{1}{1-p_i}, \quad (5.11)$$

where

$$p_{\mathbf{U}|\mathbf{D}}^k + \frac{2-p}{1-p} p_{\mathbf{D}|\mathbf{D}}^k = \frac{3-3p-2q+pq+p^2+(1-p)(p+q-1)^k}{(1-p)(2-p-q)}. \quad (5.12)$$

The minimum expected reward for any initial system state in which l_j is up ($s_{\mathbf{U}}$) is given by

$$v_{stat}^{\pi^*}(s_{\mathbf{U}}) = - \left[p_{\mathbf{U}|\mathbf{U}}^{j-1} + \frac{2-p}{1-p} p_{\mathbf{D}|\mathbf{U}}^{j-1} \right] - (j-1) - \sum_{i=j+1}^{M-1} \frac{1}{1-p_i}. \quad (5.13)$$

where

$$p_{\mathbf{U}|\mathbf{U}}^k + \frac{2-p}{1-p} p_{\mathbf{D}|\mathbf{U}}^k = \frac{3-3p-2q+pq+p^2-(1+q)(p+q-1)^k}{(1-p)(2-p-q)}. \quad (5.14)$$

Proof. See Appendix B. □

Using the facts that $j = D_{min}(n_1, l_j) + 1$ and that $|p+q-1| < 1$ for a SITD link, it is clear that (5.11) and (5.13) converge to the same result as $D_{min}(n_1, l_j) \rightarrow \infty$.

Bounds on $v^{\pi^*}(s)$ Referring to Fig. 5.2, we may break $\{X_t\}$ into the sub-sequences $\{X_t\}^-$, $\{X_t\}^j$, and $\{X_t\}^+$. $\{X_t\}^-$ is the state sequence prior to reaching n_j ; $\{X_t\}^j$ is the state sequence while the packet is at n_j and being forwarded over the SITD link to n_{j+1} ; and $\{X_t\}^+$ is the state sequence from n_{j+1} to the destination. Furthermore, since π^* is deterministic and implicit in our probability distributions and the initial system state is given, the sequence of nodes at which the packet is located is uniquely determined by the sequence of network-wide link states, denoted by $\{L_t\}$. We may also break $\{L_t\}$ into the sub-sequences $\{L_t\}^-$, $\{L_t\}^j$, and $\{L_t\}^+$, with the analogous definitions applied.

When links l_1 through l_{j-1} have the same $p_i = \alpha$, the distribution of $w(\{X_t\}^-)$ is negative binomial, so

$$\mathbf{P}(w(\{X_t\}^-) = k) = \binom{k-1}{j-2} (1-\alpha)^{j-1} \alpha^{(k-j+1)}. \quad (5.15)$$

However, when the p_i are not equal, the distribution is the sum of geometric random vari-

ables with different parameters, which, to our knowledge, has no closed-form probability distribution. Therefore, it would be useful to have upper and lower bounds on $v^{\pi^*}(s)$.

Theorem 5.4.3. *Assuming that l_1 is up in the initial system state and that $p+q \geq 1$, $v^{\pi^*}(s)$ is lower bounded by*

$$v_{lb,\geq 1}^{\pi^*}(s) = - \left[p_{\mathbf{U}|\mathbf{D}}^{j-1} + \frac{2-p}{1-p} p_{\mathbf{D}|\mathbf{D}}^{j-1} \right] - 1 - \sum_{i=2}^{j-1} \frac{1}{1-p_i} - \sum_{i=j+1}^{M-1} \frac{1}{1-p_i}, \quad (5.16)$$

and upper bounded by

$$v_{ub,\geq 1}^{\pi^*}(s) = - \left[p_{\mathbf{U}|\mathbf{U}}^{j-1} + \frac{2-p}{1-p} p_{\mathbf{D}|\mathbf{U}}^{j-1} \right] - 1 - \sum_{i=2}^{j-1} \frac{1}{1-p_i} - \sum_{i=j+1}^{M-1} \frac{1}{1-p_i}. \quad (5.17)$$

Assuming that l_1 is up in the initial system state and that $p+q < 1$, $v^{\pi^*}(s)$ is lower bounded by

$$v_{lb,< 1}^{\pi^*}(s) = - \left[p_{\mathbf{U}|\mathbf{D}}^{\text{even}(j,j-1)} + \frac{2-p}{1-p} p_{\mathbf{D}|\mathbf{D}}^{\text{even}(j,j-1)} \right] - 1 - \sum_{i=2}^{j-1} \frac{1}{1-p_i} - \sum_{i=j+1}^{M-1} \frac{1}{1-p_i}, \quad (5.18)$$

and upper bounded by

$$v_{ub,< 1}^{\pi^*}(s) = - \left[p_{\mathbf{U}|\mathbf{U}}^{\text{even}(j,j-1)} + \frac{2-p}{1-p} p_{\mathbf{D}|\mathbf{U}}^{\text{even}(j,j-1)} \right] - 1 - \sum_{i=2}^{j-1} \frac{1}{1-p_i} - \sum_{i=j+1}^{M-1} \frac{1}{1-p_i}, \quad (5.19)$$

where $\text{even}(j, j-1)$ is the function that selects the even value from the pair of integers j and $j-1$.

Proof. See Appendix C. □

Again we see that, as $D_{\min}(n_1, l_j) \rightarrow \infty$, (5.16)-(5.19) converge to the same result. Due to the similarity between the bounds for $p+q \geq 1$ and $p+q < 1$, we restrict further discussions to the former. The convergence of (5.16)-(5.19) to the same result indicates that, given some

$\epsilon > 0$ and initial system states s_1 and s_2 , there is some value of j for which

$$\frac{|v^{\pi^*}(s_1) - v^{\pi^*}(s_2)|}{\min(|v^{\pi^*}(s_1)|, |v^{\pi^*}(s_2)|)} \leq \frac{v_{ub, \geq 1}^{\pi^*}(s_1) - v_{lb, \geq 1}^{\pi^*}(s_1)}{|v_{ub, \geq 1}^{\pi^*}(s_1)|} < \epsilon, \quad (5.20)$$

where we have arbitrarily chosen to use s_1 in the middle term. The same inequality holds if we choose s_2 . Now, if we assume that s_1 and s_2 represent system states that only differ in the state of link l_j , then we can interpret (5.20) as bounding the maximum percent error that may occur when the state of l_j is unknown. In other words, the error due to having no knowledge of the state of l_j vanishes as the minimum distance from the decision node, n_1 , to l_j increases. Intuitively, this indicates that the value of state information about l_j decreases for links further from the node making the routing decision.

5.4.2.2 Arbitrary Networks

In Section 5.4.2.1, we were able to obtain bounds on the value function that show that as $D_{min}(n_1, l_j) \rightarrow \infty$, the state of the SITD link has less effect on the value function. Of course, in the linear network of Fig. 5.2, it makes no difference what the state of l_j is when deciding on the action because the optimal action is always to forward the packet toward the destination. This is a byproduct of having a limited set of actions at any particular node when the network is linear. Therefore, we may ask whether the same property, namely that $v_{stat}^{\pi^*}(s_{\mathbf{D}})$ and $v_{stat}^{\pi^*}(s_{\mathbf{U}})$ converge to the same value as $D_{min}(n_1, l_j) \rightarrow \infty$, holds for a broader class of networks.

Let us consider the class of networks in which all links are SITD, possibly with unique transition probabilities for each link. Again, we refer to the optimal policy as π^* , which, as explained in Section 5.3.2, may be assumed to be deterministic and stationary. Since the policy is stationary, the system state sequence $\{X_t\}$ is a Markov process.

Again, knowledge of the (deterministic) policy and the link state sequence $\{L_t\}$ is sufficient to uniquely determine the sequence $\{X_t\}$; therefore, we use only $\{L_t\}$ in our analysis. When

the optimal policy is unknown, the optimum value function can be written as

$$v^{\pi^*}(s) = \max_{\pi} \left[- \sum_{\mathcal{L}} P^{\pi}(\{L_t\} \mid L_1) w(\{L_t\}) \right], \quad (5.21)$$

where $P^{\pi}(\cdot)$ is the probability distribution of link state sequences under the policy π . For this arbitrary SITD network, we have the following theorem.

Theorem 5.4.4. *For a network consisting of SITD links and for initial states $s_{\mathbf{D}} = (n, L_{\mathbf{D}})$ and $s_{\mathbf{U}} = (n, L_{\mathbf{U}})$ that only differ in that the state of l_j is initially down in $s_{\mathbf{D}}$ and up in $s_{\mathbf{U}}$, $|v^{\pi^*}(s_{\mathbf{D}}) - v^{\pi^*}(s_{\mathbf{U}})| \rightarrow 0$ as $D_{\min}(n, l_j) \rightarrow \infty$.*

Proof. See Appendix D. □

5.4.3 Delayed State Information

In formulating the minimum expected cost routing problem as a MDP, we must assume that state information for all pertinent links is available instantaneously. In reality, link state information is only available instantaneously for links adjacent to the node holding the packet. State information for links that are not adjacent must be transferred from an observing node to other nodes in the network that are incapable of direct observation. In this case, the value of $D_{\min}$ again plays an important role. For the derivations in Sections 5.4.2.1 and 5.4.2.2, we have shown that for a given set of link transition probabilities, increasing $D_{\min}$ results in a vanishingly small difference in $s_{\mathbf{D}}$ and $s_{\mathbf{U}}$. The rate at which $s_{\mathbf{D}}$ and $s_{\mathbf{U}}$ converge is exponential since it is a function of the k -step link transition probabilities, shown in (B.14).

When we incorporate a delay in the availability of link state information, the optimal routing problem is more aptly described as a POMDP. In the POMDP, our knowledge of the current system state is described by a probability distribution over all possible system states. However, the problem can still be represented in a form similar to (5.21), except that we

assume that the state observation is delayed by some number of time slots. The result is

$$v^{\pi^*}(s) = \max_{\pi} \left[- \sum_{\mathcal{L}} P^{\pi}(\{L_t\} \mid L_{-K}) w(\{L_t\}) \right], \quad (5.22)$$

where L_{-K} indicates a delay of K time slots in the initial state observation. We may follow the steps from (D.1) to (D.6), replacing L_1 with L_{-K} , after which we find that $v^{\pi^*}(s_{\mathbf{D}})$ and $v^{\pi^*}(s_{\mathbf{U}})$ converge at a rate that is exponential in $D_{\min} + K$.

We see the same behavior in the results for the linear network in Section 5.4.2.1. In this case, we can analyze the effect on a particular link. Since the delay in obtaining link state information on the SITD link l_j must be at least $D_{\min}(n_1, l_j)$, we may follow the procedure in Appendix B through, replacing $l_j[1]$ with $l_j[-j + 1]$, after which we find that the upper and lower bounds in Theorem 5.4.3 are exponential in $2(j - 1)$ as opposed to just $(j - 1)$. Thus, when we consider the minimum delay in obtaining link state information, we see a dramatic shortening of the distance at which the difference in $v^{\pi^*}(s_{\mathbf{D}})$ and $v^{\pi^*}(s_{\mathbf{U}})$ becomes inconsequential.

5.5 The DEC-MDP Formulation

In general SITD networks, not every link would experience the same amount of delay. In fact, in the most realistic description, the per-link state information would be delayed by a random amount. Such a characterization may be intractable. However, recent research in decentralized MDPs has led to a type of POMDP called the DEC-MDP. The DEC-MDP is a special case of a POMDP in which the observations of the agents (i.e. nodes) collectively determine the state of the system, although no single agent has the complete set of observations [97].

While the general DEC-MDP is NEXP-Hard (NEXP is the class of decision problems that can be solved in no better than exponential time by a non-deterministic Turing machine) [97],

there are special cases in which polynomial time algorithms exist, such as the one formulated by Beynier and Mouaddib [109]. A key reason why the DEC-MDP in [109] is polynomial-time solvable is that there are precedence constraints on the way that agents make decisions. Precedence constraints also exist in the Markovian routing DEC-MDP since only the node with the packet makes the forwarding decision. Also, our DEC-MDP is clearly polynomial-time solvable when the network is composed of only SITI links. This is due to Theorem 5.4.1, which shows that even though the system state is not completely observed, the optimal policy can be executed based only on local link observations. Therefore, we remain hopeful that polynomial-time algorithms exist for either approximate or exact solutions to the MANET routing DEC-MDP.

5.6 The Markov Routing DEC-MDP as a Cognitive Network

We again refer to Chapter 2 and the key characteristics of a cognitive network, which are its ability to sense its environment, learn, plan, decide, and act in order to achieve an end-to-end goal. The agent-based framework of the DEC-MDP immediately recalls the presentation in Chapter 3 of a cognitive network as a MAS. The agents sense the environment by observing the state of network links. Their actions come from the set, $\mathcal{A}$, defined in Section 5.3. Decisions are made according to the optimal routing policy, which is forward-looking, similar to the case in Chapter 4, in that the objective is to minimize an expected future cost based on the available current state information. The objective involves the actions of all intermediate nodes/agents, while the expected cost covers the lifetime of the packet in the network, making the objective truly end-to-end.

Learning is not explicitly treated in this chapter, but becomes apparent when we consider that the transition probabilities of the Markov process are generally not available a priori. This necessitates a technique for estimating, or learning, the transition probabilities. It has

been shown that reinforcement learning is able to converge to the optimal policy when the transition probabilities of the underlying Markov process are unknown [110]. Furthermore, Q-learning, which we discussed in Section 3.2.7, has been applied to solving POMDPs [111]. This relationship between reinforcement learning and MDPs satisfies the learning requirement and completes our view of the cognitive network as a fitting solution to this MANET routing problem.

Chapter 6

Conclusions, Contributions, and Future Work

6.1 Conclusions

This dissertation has sought to solidify the cognitive network concept, while simultaneously addressing two important open problems. We initially focused on answering fundamental questions such as “What is a cognitive network and how does it compare to related concepts?” and “What methods might one use for learning and reasoning in cognitive networks?” The answers to these questions could be debated and expounded upon for years, but one is not likely to reach concrete footing until applications of cognitive networks are considered. Thorough investigation of applications has the tendency to refine what is initially conceptual. Therefore, the bulk of this dissertation has sought to tie applications to the high-level concept.

In Chapter 4, we presented a multichannel topology control problem for DSA. This extended the focus of DSA research to the post-detection stage, i.e. the time period after the network nodes have decided that some group of channels is available for use. It also expanded

traditional topology control into the exogenous objective function realm. As the objective function in this problem is quite difficult to evaluate, and even more difficult to solve optimally, we developed a suboptimal heuristic algorithm. We compared the performance of this heuristic against a heuristic for a power-based multichannel topology control problem and against solutions obtained via a long-running genetic algorithm. We showed that our heuristic is a good compromise of performance versus solution quality. Finally, we related this problem solution back to cognitive networks.

While DSA is a fitting application for cognitive networks, focusing on a single application may lead some to the conclusion that they are synonymous. Therefore, Chapter 5 presents a different application for cognitive networks, that of MANET routing. We selected this problem because, despite an overabundance of MANET routing protocols, the question of whether a MANET routing protocol is optimal is still open, with few researchers even asking the question. We place the question on solid analytical ground by first modeling the network at the link-level using a Markovian random graph sequence. For this model, the objective is to minimize the expected cost along the random path to the destination, and the optimal routing policy is the solution to a MDP. We showed that there always exists a stationary, deterministic, finite expected cost optimal policy and performed analysis on several special cases. The analysis revealed analytical evidence justifying the use of local information, where the locality of information is a function of a link's transition probabilities. We then extended the MDP formulation to the more realistic case of the DEC-MDP, in which agents act in a decentralized manner with partial information. The DEC-MDP has recently attracted the attention of the AI as a framework for decision-making in MAS. This tie back to MAS brings us full-circle to Chapter 3, where we initially frame a cognitive network as a MAS.

6.2 Contributions of this Dissertation

Here, we identify the primary contributions of this dissertation. For ease of review, we provide this information in list form, organized into three categories: foundational work, the DSA application, and the MANET routing application. For information regarding the publications resulting from these contributions, see the footnotes at the beginnings of Chapters 2 through 5.

6.2.1 Foundations

While background information is usually pulled from existing literature, work on cognitive networks was in such an early stage of maturity at the time that our research began that the work in Chapters 2 and 3 was not available in the existing literature. The foundational contributions to cognitive networking are:

- We provided further formalization of the cognitive network concept.
- We conducted the first effort to identify suitable approaches for learning and reasoning in cognitive networks.
- We formulated a cognitive node architecture.

6.2.2 The DSA Application

In Chapter 4, we introduced a multichannel topology control problem for DSA that is new to the DSA and topology control literature. The multichannel nature of the work places it within a small collection of topology control research that addresses power and channel adaptation simultaneously. The contributions are:

- We developed the first topology control model that combines a stochastic path loss

model, direct incorporation of a MAC protocol in the objective function, and joint consideration of power and channel selection.

- We developed a heuristic for the multichannel topology control problem **MinMaxEo**. In particular, the randomized algorithm in Section 4.4.3 for estimating the per-MIS E_o is new and is crucial to developing a polynomial-time heuristic.
- We formulated a novel power-based multichannel topology control problem, **MinMaxP**, and proved that it is NP-hard.
- We developed a heuristic for **MinMaxP**, which exhibits excellent performance.
- We developed two genetic algorithms that provide performance comparisons for the heuristics.

6.2.3 The MANET Routing Application

Optimal routing in static networks is a subject that was resolved long ago. However, optimal routing in MANETs has been an elusive subject, which has likely been a major factor in the proliferation of MANET routing protocols. Because the research community has been unable to specify the best possible performance of a routing protocol in a MANET, it has relied largely on simulation for comparison, which has resulted in a great degree of confusion. Our contributions to MANET routing are:

- We analytically specify an optimal routing policy for the Markovian random graph model of a MANET.
- We prove that an optimal, stationary, finite-reward policy always exists, which alleviates concerns about the possibility that a network may not have a finite expected cost routing policy.

- We prove several theorems for special cases of Markovian networks related to the locality of information, which is crucial for reducing the complexity of the MDPs as well as dealing with incomplete state information.
- We re-formulate the problem as a DEC-MDP, reflecting realistic constraints on the availability of state information.

6.3 Future Work

As with any research, the process of answering a few questions generates dozens of new questions. This section mentions some of the questions that have arisen in the course of this work, which we feel are deserving of further research.

6.3.1 Directions for MinMaxEo Topology Control

We see tremendous opportunity for developing methods for estimating the outage potential map. The method that we have presented herein assumes a simple underlying statistical model for primary user behavior. Much more can be done to develop methods that more fully reflect primary user patterns. However, this development is in need of real-world data sets for the behavior of primary users as we believe that most existing mobility models do not reflect the behavior of rational primary users.

We have used the maximum likelihood position estimate in a binary sense in that we choose a single tile for the location of the detected primary user. However, it may be more accurate to update each tile's weight using the likelihood itself. The likelihood for a tile would be the area integral of the conditional probability over all points within that tile. It may also be feasible to use time difference of arrival (TDOA), perhaps in conjunction with the maximum likelihood RSS method. Additionally, questions remain as to how to select the ROI and the tile sizes/shapes within the ROI.

The heuristic that we have presented for **MinMaxEo** may be improved by more accurate estimates for $r(l_{ij}, \mathcal{M}_{\bar{P}, H})$ and $m(\mathcal{M}_{\bar{P}, H})$. The randomized algorithm in Section 4.4.3 for sampling the MISs exhibits a bias that varies with the conflict graph. An unbiased sampling of MISs, if such a sampling is possible in polynomial time, would improve both $\hat{r}(l_{ij}, \mathcal{M}_{\bar{P}, H})$ and $\hat{m}(\mathcal{M}_{\bar{P}, H})$. Additional improvement may be possible by jointly modifying power and channel selections rather than proceeding sequentially.

Although it is not the focus of this work, the **MinMaxP** multi-channel topology control problem is interesting in and of itself. We believe that it is a natural extension from the single-channel topology control realm to a multi-channel network. Such an application may focus on interference within the network, rather than interference to an external entity like primary users. The problem formulation is amenable to analysis, and it may be possible to formulate an approximation algorithm for **MinMaxP** by modifying an existing approximation algorithm for the minimum makespan problem.

6.3.2 MANET Routing Future Work

The MANET routing problem formulation assumes that a single packet is to be routed to a single destination. This is the standard approach in unicast routing in the Internet. The problem appears to be easily extensible to the case in which multiple packets are en route to multiple destinations simultaneously. This is akin to QoS routing since the objective covers multiple source-destination pairs simultaneously. Another extension would be to expand the action space to allow forwarding of multiple copies of a packet. However, this extension is problematic in that the cost function must be more complex than just the immediate link cost (i.e. a single hop, link latency, etc.); otherwise, the optimal solution will always be to forward the packet along all possible paths.

Another interesting expansion of the action set is to allow network coding. Traditional routing, as well as multipath routing, can be seen as a special case of network coding. However, in true network coding, packets are combined by intermediate nodes, usually using

finite field arithmetic, and then reconstructed at the destination once a sufficient number of packets have been received. Network coding allows for the introduction of packet redundancy, which is especially useful in wireless networks because of the high probability of packet loss.

Finally, the DEC-MDP formulation has many unanswered questions. Foremost among them are whether the problem is NEXP-hard and whether reasonable performance can be obtained with polynomial time heuristics or approximation algorithms. The use of reinforcement learning for solving a DEC-MDP is also an enticing topic. Our major concern here is that the DEC-MDP is relatively new, which may limit the methods and analysis that are available.

Appendix A

Proof of Lemma 4.5.1

Proof. Assume an instance of **MinMake** with a set of jobs $\{\tau_1, \tau_2, \dots, \tau_k\}$ and a set of processors $\{M_1, M_2, \dots, M_m\}$, $m \geq 2$. For k even, no reduction is necessary. Suppose k is odd and that an assignment of jobs to processors is denoted by S , with $\mathcal{S}^*$ being the set of assignments that yields the minimum makespan. Furthermore, for assignment S , let the makespan be denoted by $M_{max}(S)$, with $M_{min}(S)$ representing the total processing time of the processor that finishes first under assignment S . For any pair of assignments $S_1, S_2 \in \mathcal{S}^*$, it must be true that $M_{max}(S_1) = M_{max}(S_2) = M_{max}^*$.

We reduce the odd- k instance of **MinMake** to an even- k instance by choosing an additional job, $k + 1$, with completion time that satisfies

$$\tau_{k+1} < \left[\min_{S \notin \mathcal{S}^*} M_{max}(S) \right] - M_{max}^*, \quad (\text{A.1})$$

and, when

$$M_{max}^* > \max_{S \in \mathcal{S}^*} M_{min}(S),$$

we impose the additional constraint that

$$\tau_{k+1} < \min_{S \in \mathcal{S}^*} [M_{max}^* - M_{min}(S)]. \quad (\text{A.2})$$

The new set of possible assignments is the cross product of the possible assignments using k jobs with the m possible assignments of the new job $k + 1$. We can find the optimal solution for this new set of jobs by first fixing S (an assignment of the original k jobs) and finding the minimum makespan that can be achieved by assigning job $k + 1$ to one of the m processors. We then repeat this for every S and take the combination of S and the assignment of job $k + 1$ that result in the minimum makespan.

Consider first the set of assignments $S \notin \mathcal{S}^*$. We first note that adding a new job to these assignments can only increase the makespan, so that an optimal choice is always to assign the job to the processor that would finish first. This processor now has processing time $M_{\min}(S) + \tau_{k+1}$. By our selection of τ_{k+1} in (A.1), we can guarantee that

$$M_{\min}(S) + \tau_{k+1} < \left[\min_{S \notin \mathcal{S}^*} M_{\max}(S) \right] - M_{\max}^* + M_{\min}(S).$$

A direct result of the fact that $M_{\max}^* < M_{\max}(S)$ for $S \notin \mathcal{S}^*$ is that $M_{\max}^* > M_{\min}(S)$ because $\sum_i \tau_i$ must be conserved for all schedules. This allows us to say that

$$\left[\min_{S \notin \mathcal{S}^*} M_{\max}(S) \right] - M_{\max}^* + M_{\min}(S) < M_{\max}(S),$$

and therefore

$$M_{\min}(S) + \tau_{k+1} < M_{\max}(S).$$

This implies that for all $S \notin \mathcal{S}^*$, the optimum assignment of job $k + 1$ results in no change to the makespan.

Now we consider the change in the makespan for optimal assignments. The first case is when the optimal assignments satisfy the condition

$$M_{\max}^* > \max_{S \in \mathcal{S}^*} M_{\min}(S).$$

Again, the optimum assignment of job $k+1$ is to the processor with minimum total processing

time, in which case the constraint in (A.2) ensures that

$$M_{\min}(S) + \tau_{k+1} < M_{\max}^*,$$

for all $S \in \mathcal{S}^*$. Therefore, all $S \in \mathcal{S}^*$ remain optimal after the addition of the new job.

The second case occurs when

$$M_{\max}^* = \max_{S \in \mathcal{S}^*} M_{\min}(S).$$

In this case, all processors in all optimal assignments have exactly the same total processing time. This occurs because $M_{\max}^* = 1/m \sum_i \tau_i$. By conservation of total job completion time, $1/m \sum_i \tau_i$ is the minimum possible makespan for any instance of **MinMake**, and every processor must have the same total processing time in order to achieve a makespan of $1/m \sum_i \tau_i$. Therefore, after adding τ_{k+1} to an arbitrary processor in each optimal assignment, every $S \in \mathcal{S}^*$ will now have a makespan of $M_{\max}^* + \tau_{k+1}$. All of these assignments will still be optimal because (A.1) guarantees that $M_{\max}^* + \tau_{k+1} < [\min_{S \notin \mathcal{S}^*} M_{\max}(S)]$.

Therefore, given the optimum assignment for the odd- k instance, we can determine in linear time the optimal assignment for the even- k instance of **MinMake**. Conversely, knowing the optimal solution to the even- k instance immediately tells us the solution to the odd- k instance. □

Appendix B

Proof of Theorem 5.4.2

Proof. After breaking $\{L_t\}$ into the sub-sequences $\{L_t\}^-$, $\{L_t\}^j$, and $\{L_t\}^+$ as described in Section 5.4.2.1, we rewrite (5.10) as

$$v^{\pi^*}(s) = - \sum_{\mathcal{L}} P(\{L_t\} \mid L_1) \cdot [w(\{L_t\}^-) + w(\{L_t\}^j) + w(\{L_t\}^+)], \quad (\text{B.1})$$

where $\mathcal{L}$ is the set of possible sequences for $\{L_t\}$ and L_1 is the initial state of the links. This same procedure of subdividing the state sequence may be repeated on $\{L_t\}^-$ and $\{L_t\}^+$ until we have a set of subsequences $\{L_t\}^i$, each capturing the interval from when the packet arrives at node n_i through the transition across a SITI link. This procedure leads to the following equations:

$$\begin{aligned} \sum_{\mathcal{L}} P(\{L_t\} \mid L_1) w(\{L_t\}^-) = \\ \sum_{i=1}^{j-1} \sum_{\mathcal{L}} P(\{L_t\} \mid L_1) w(\{L_t\}^i) \end{aligned} \quad (\text{B.2})$$

and

$$\begin{aligned} \sum_{\mathcal{L}} \mathbb{P}(\{L_t\} \mid L_1) w(\{L_t\}^+) = \\ \sum_{i=j+1}^{M-1} \sum_{\mathcal{L}} \mathbb{P}(\{L_t\} \mid L_1) w(\{X_t\}^i). \end{aligned} \quad (\text{B.3})$$

Since all links are spatially independent, we may write

$$\mathbb{P}(\{L_t\} \mid L_1) = \prod_{i=1}^{M-1} \mathbb{P}(l_i[2], l_i[3], \dots, l_i[k_3] \mid l_i[1]), \quad (\text{B.4})$$

where $l_i[t]$ is the state of link i in time slot t and k_3 is the number of time slots in the sequence.

Using (B.4) in (B.3) and (B.2) and expanding the link state sequences to be on a per-link basis, we have

$$\begin{aligned} \sum_{\mathcal{L}} \mathbb{P}(\{L_t\} \mid L_1) w(\{L_t\}^i) = \\ \sum_{\mathcal{L}_1} \dots \sum_{\mathcal{L}_{M-1}} \prod_{i=1}^{M-1} \mathbb{P}(l_i[2], \dots, l_i[k_3] \mid l_i[1]) w(\{L_t\}^i), \end{aligned} \quad (\text{B.5})$$

where $\mathcal{L}_i$ is the set of all possible sequences of the state of l_i . Since $w(\{L_t\}^i)$ is constant for all links other than l_i and using the fact that l_i is a SITI link, (B.5) reduces to

$$\begin{aligned} \sum_{\mathcal{L}} \mathbb{P}(\{L_t\} \mid L_1) w(\{L_t\}^i) = \\ \sum_{\mathcal{L}_i} \mathbb{P}(l_i[2], \dots, l_i[k_3]) w(\{L_t\}^i). \end{aligned} \quad (\text{B.6})$$

The result in (B.6) is the expected number of time slots until the SITI link is up, which is simply the expected value of a geometric random variable. Therefore, substituting (B.6)

into (B.2) and (B.3) and then using this result in (B.1), we obtain

$$\begin{aligned}
v^{\pi^*}(s) = & - \sum_{\mathcal{L}} P(\{L_t\} \mid L_1) w(\{L_t\}^j) - \\
& \sum_{\mathcal{L}^1} P(l_1[2], \dots, l_1[k_3] \mid l_1[1]) w(\{L_t\}^1) - \\
& \sum_{i=2}^{j-1} \frac{1}{1-p_i} - \sum_{i=j+1}^{M-1} \frac{1}{1-p_i},
\end{aligned} \tag{B.7}$$

where p_i denotes the probability that SITI link l_i is down. When the initial system state is such that the packet is at n_1 and l_1 is up,

$$\sum_{\mathcal{L}^1} P(l_1[2], \dots, l_1[k_3] \mid l_1[1]) w(\{L_t\}^1) = 1, \tag{B.8}$$

and when the initial system state is such that the packet is at n_1 and l_1 is down,

$$\sum_{\mathcal{L}^1} P(l_1[2], \dots, l_1[k_3] \mid l_1[1]) w(\{L_t\}^1) = 1 + \frac{1}{1-p_1}. \tag{B.9}$$

The only term in (B.7) for which we do not yet have a direct solution is $\sum_{\mathcal{L}} P(\{L_t\} \mid L_1) w(\{L_t\}^j)$. We can apply the same procedure used in (B.4) and (B.5), but with a slightly different result since l_i is SITD,

$$\begin{aligned}
\sum_{\mathcal{L}} P(\{L_t\} \mid L_1) w(\{L_t\}^j) = \\
\sum_{\mathcal{L}_j} P(l_j[2], \dots, l_j[k_3] \mid l_j[1]) w(\{L_t\}^j).
\end{aligned} \tag{B.10}$$

Now, since $w(\{L_t\}^j)$ depends only on a subsequence of $\{l_j[1], \dots, l_j[k_3]\}$, we break the sequence of states of l_j into the following three subsequences: $\{l_j[1], \dots, l_j[k_1]\}$ is the sequence of states prior to the packet arriving at n_j , $\{l_j[k_1 + 1], \dots, l_j[k_2]\}$ is the sequence while the packet is located at n_j and being transferred across l_j , and $\{l_j[k_2 + 1], \dots, l_j[k_3]\}$ is the sequence from the time that the packet reaches n_{j+1} until it arrives at the destina-

tion. We denote the corresponding sets for each sequence as $\mathcal{L}_j^-$, $\mathcal{L}_j^j$, and $\mathcal{L}_j^+$. Because $w(\{L_t\}^j) = k_2 - k_1$ only depends on $\mathcal{L}_j^j$, we are able to sum out the parts of the link state sequence corresponding to $\mathcal{L}_j^-$ and $\mathcal{L}_j^+$, leaving us with

$$\begin{aligned} & \sum_{\mathcal{L}_j} P(l_j[2], \dots, l_j[k_3] \mid l_j[1]) w(\{L_t\}^j) = \\ & \sum_{k_1=j-1}^{\infty} P(k_1) \sum_{\mathcal{L}_j^j} P(l_j[k_1+1], \dots, l_j[k_2] \mid l_j[1]) (k_2 - k_1), \end{aligned} \quad (\text{B.11})$$

where $P(k_1)$ is the probability distribution of the length of the sequence $\{l_j[1], \dots, l_j[k_1]\}$. At this point, we may apply the Markov property to l_j to see that

$$\begin{aligned} & P(l_j[k_1+1], \dots, l_j[k_2] \mid l_j[1]) = \\ & P(l_j[k_1+1] \mid l_j[1]) \cdots P(l_j[k_2] \mid l_j[k_2-1]). \end{aligned} \quad (\text{B.12})$$

We recognize $P(l_j[k_1+1] \mid l_j[1])$ as the (k_1) -step transition probability for l_j and that probabilities of the form $P(l_j[t+1] \mid l_j[t])$ are 1-step link state transition probabilities.

Given the 1-step transition probability matrix for link l_j ,

$$T_{l_j} = \begin{bmatrix} p & 1-q \\ 1-p & q \end{bmatrix}, \quad (\text{B.13})$$

where $p = P(l_j[t+1] = \mathbf{D} \mid l_j[t] = \mathbf{D})$ and $q = P(l_j[t+1] = \mathbf{U} \mid l_j[t] = \mathbf{U})$, the k -step transition probabilities are given by the matrix

$$T_{l_j}^k = \begin{bmatrix} \frac{1-q+(1-p)(p+q-1)^k}{2-p-q} & \frac{1-q-(1-q)(p+q-1)^k}{2-p-q} \\ \frac{1-p-(1-p)(p+q-1)^k}{2-p-q} & \frac{1-p+(1-q)(p+q-1)^k}{2-p-q} \end{bmatrix}. \quad (\text{B.14})$$

As there is only one SITD link, we represent the k -step transition probabilities in the more compact form $p_{\mathbf{D}|\mathbf{D}}^k$, $p_{\mathbf{U}|\mathbf{D}}^k$, etc, recognizing that they can be directly computed from (B.14).

We now note that the policy restricts $\mathcal{L}_j^j$ to sequences consisting of 0 or more down states followed by a single up state. Therefore, assuming that $l_j[1] = \mathbf{D}$, we may write

$$\begin{aligned} \sum_{\mathcal{L}_j^j} \mathbb{P}(l_j[k_1 + 1] \cdots l_j[k_2] \mid l_j[1] = \mathbf{D}) (k_2 - k_1) = \\ p_{\mathbf{U}|\mathbf{D}}^{k_1} + p_{\mathbf{D}|\mathbf{D}}^{k_1} \sum_{t=1}^{\infty} (t+1) p^{t-1} (1-p). \end{aligned} \quad (\text{B.15})$$

The same holds when l_j is initially up, with $p_{\mathbf{U}|\mathbf{D}}^{k_1}$ being replaced by $p_{\mathbf{U}|\mathbf{U}}^{k_1}$ and $p_{\mathbf{D}|\mathbf{D}}^{k_1}$ replaced by $p_{\mathbf{D}|\mathbf{U}}^{k_1}$.

After performing some additional algebra on (B.15) and substituting it into (B.11), we can write (B.10) as

$$\begin{aligned} \sum_{\mathcal{L}} \mathbb{P}(\{L_t\} \mid L_1) w(\{L_t\}^j) = \\ \sum_{k_1=j-1}^{\infty} \mathbb{P}(k_1) \left[p_{\mathbf{U}|\mathbf{D}}^{k_1} + \frac{2-p}{1-p} p_{\mathbf{D}|\mathbf{D}}^{k_1} \right], \end{aligned} \quad (\text{B.16})$$

and

$$\begin{aligned} \sum_{\mathcal{L}} \mathbb{P}(\{L_t\} \mid L_1) w(\{L_t\}^j) = \\ \sum_{k_1=j-1}^{\infty} \mathbb{P}(k_1) \left[p_{\mathbf{U}|\mathbf{U}}^{k_1} + \frac{2-p}{1-p} p_{\mathbf{D}|\mathbf{U}}^{k_1} \right]. \end{aligned} \quad (\text{B.17})$$

To compute (B.16) and (B.17) in terms of p , q , and k , we may use

$$\begin{aligned} p_{\mathbf{U}|\mathbf{D}}^k + \frac{2-p}{1-p} p_{\mathbf{D}|\mathbf{D}}^k = \\ \frac{3 - 3p - 2q + pq + p^2 + (1-p)(p+q-1)^k}{(1-p)(2-p-q)}, \end{aligned} \quad (\text{B.18})$$

and

$$p_{\mathbf{U}|\mathbf{U}}^k + \frac{2-p}{1-p} p_{\mathbf{D}|\mathbf{U}}^k = \frac{3-3p-2q+pq+p^2-(1+q)(p+q-1)^k}{(1-p)(2-p-q)}. \quad (\text{B.19})$$

When links l_1 through l_{j-1} are static, the delay from n_1 to n_j is fixed, which means that $P(k_1 = j-1) = 1$. Using (B.16) and (B.17), we can write (B.7) as

$$v_{stat}^{\pi^*}(s_{\mathbf{D}}) = - \left[p_{\mathbf{U}|\mathbf{D}}^{j-1} + \frac{2-p}{1-p} p_{\mathbf{D}|\mathbf{D}}^{j-1} \right] - (j-1) - \sum_{i=j+1}^{M-1} \frac{1}{1-p_i}, \quad (\text{B.20})$$

and

$$v_{stat}^{\pi^*}(s_{\mathbf{U}}) = - \left[p_{\mathbf{U}|\mathbf{U}}^{j-1} + \frac{2-p}{1-p} p_{\mathbf{D}|\mathbf{U}}^{j-1} \right] - (j-1) - \sum_{i=j+1}^{M-1} \frac{1}{1-p_i}. \quad (\text{B.21})$$

□

Appendix C

Proof of Theorem 5.4.3

Proof. When $p + q \geq 1$, we see that (B.18) is monotonically decreasing in k and that (B.19) is monotonically increasing in k . Therefore, we can upper bound $v^{\pi^*}(s)$ by using the fact that

$$p_{\mathbf{U}|\mathbf{D}}^k + \frac{2-p}{1-p} p_{\mathbf{D}|\mathbf{D}}^k \geq p_{\mathbf{U}|\mathbf{D}}^{k+r} + \frac{2-p}{1-p} p_{\mathbf{D}|\mathbf{D}}^{k+r}, \quad (\text{C.1})$$

and form a lower bound using

$$p_{\mathbf{U}|\mathbf{U}}^k + \frac{2-p}{1-p} p_{\mathbf{D}|\mathbf{U}}^k \leq p_{\mathbf{U}|\mathbf{U}}^{k+r} + \frac{2-p}{1-p} p_{\mathbf{D}|\mathbf{U}}^{k+r}, \quad (\text{C.2})$$

where, for both equations, $r \geq 0$.

When $p + q < 1$, (B.18) and (B.19) are not monotonic. Instead, they alternate above and below their asymptotic values depending on whether k is odd or even. However, $|(1-p)(p+q-1)^k|$ and $|(1+q)(p+q-1)^k|$ are monotonically decreasing. Therefore, if we define the function $\text{even}(k, k+1)$ as choosing the even value from the pair k and $k+1$, then, we can form an upper bound using the property

$$p_{\mathbf{U}|\mathbf{D}}^{\text{even}(k, k+1)} + \frac{2-p}{1-p} p_{\mathbf{D}|\mathbf{D}}^{\text{even}(k, k+1)} \geq p_{\mathbf{U}|\mathbf{D}}^{k+r} + \frac{2-p}{1-p} p_{\mathbf{D}|\mathbf{D}}^{k+r}, \quad (\text{C.3})$$

and the lower bound follows from the property

$$p_{\mathbf{U}|\mathbf{U}}^{\text{even}(k,k-1)} + \frac{2-p}{1-p} p_{\mathbf{D}|\mathbf{U}}^{\text{even}(k,k-1)} \leq p_{\mathbf{U}|\mathbf{U}}^{k+r} + \frac{2-p}{1-p} p_{\mathbf{D}|\mathbf{U}}^{k+r}, \quad (\text{C.4})$$

where, for both equations, $r \geq 0$. □

Appendix D

Proof of Theorem 5.4.4

Proof. Supposing that we are interested in the effect of SITD link l_j , we may divide $\mathcal{L}$ into a set of sequences that include at least one transition over l_j , denoted by $\mathcal{L}^{1+}$ and a set of sequences that include no transitions across l_j , denoted by $\mathcal{L}^0$. We can then write (5.21) as

$$v^{\pi^*}(s) = \max_{\pi} \left[- \sum_{\mathcal{L}^{1+}} P^{\pi}(\{L_t\} \mid L_1) w(\{L_t\}) - \sum_{\mathcal{L}^0} P^{\pi}(\{L_t\} \mid L_1) w(\{L_t\}) \right]. \quad (\text{D.1})$$

If $\sum_{\mathcal{L}^{1+}} P^{\pi^*}(\{L_t\} \mid L_1) = 0$, then the state of l_j clearly has no effect on $v^{\pi^*}(s)$ and $v^{\pi^*}(s_{\mathbf{U}}) - v^{\pi^*}(s_{\mathbf{D}}) = 0$. For the alternative case, we may assume that $\sum_{\mathcal{L}^{1+}} P^{\pi^*}(\{L_t\} \mid L_1) > 0$. This allows us to break $\{L_t\}$ into the subsequences $\{L_t\}^-$ and $\{L_t\}^{j+}$, where $\{L_t\}^-$ is the state sequence prior to the first time that the packet arrives at a node adjacent to l_j and $\{L_t\}^{j+}$ is the state sequence from the time that the packet first reaches a node adjacent to l_j until it reaches the destination. Note that whereas $\{L_t\}^-$ does not include any states in which the packet is at a node adjacent to l_j , $\{L_t\}^{j+}$ may include an arbitrary number of such occurrences.

Using the fact that $w(\{L_t\}) = w(\{L_t\}^-) + w(\{L_t\}^{j+})$, we can break the first term in (D.1)

into

$$\begin{aligned} \sum_{\mathcal{L}^{1+}} \mathbb{P}^\pi(\{L_t\} \mid L_1) w(\{L_t\}) = \\ \sum_{\mathcal{L}^{1+}} \mathbb{P}^\pi(\{L_t\} \mid L_1) w(\{L_t\}^-) + \\ \sum_{\mathcal{L}^{1+}} \mathbb{P}^\pi(\{L_t\} \mid L_1) w(\{L_t\}^{j+}) \Big]. \end{aligned} \quad (\text{D.2})$$

By assumption, the packet is never at a node connected to l_j during $\{L_t\}^-$, so that, by the SITD nature of all links,

$$\begin{aligned} \sum_{\mathcal{L}^{1+}} \mathbb{P}^\pi(\{L_t\} \mid L_{\mathbf{U}}) w(\{L_t\}^-) = \\ \sum_{\mathcal{L}^{1+}} \mathbb{P}^\pi(\{L_t\} \mid L_{\mathbf{D}}) w(\{L_t\}^-), \end{aligned} \quad (\text{D.3})$$

where $L_{\mathbf{U}}$ is an initial state with l_j up and $L_{\mathbf{D}}$ is an initial state that matches $L_{\mathbf{U}}$ except that l_j is down.

Addressing the second term in (D.2), we separate $\sum_{\mathcal{L}^{1+}}$ into sums over $\{L_t\}^-$ and $\{L_t\}^{j+}$, resulting in

$$\begin{aligned} \sum_{\mathcal{L}^{1+}} \mathbb{P}^\pi(\{L_t\} \mid L_1) w(\{L_t\}^{j+}) = \\ \sum_{\mathcal{L}^-} \sum_{\mathcal{L}^{j+}} \mathbb{P}^\pi(\{L_t\}^-, \{L_t\}^{j+} \mid L_1) w(\{L_t\}^{j+}), \end{aligned} \quad (\text{D.4})$$

where $\mathcal{L}^-$ and $\mathcal{L}^{j+}$ are, respectively, the sets of all possible sequences of $\{L_t\}^-$ and $\{L_t\}^{j+}$. We now list out the series of states as $\{L_t\}^- = \{L_1, L_2, \dots, L_{k_1}\}$ and $\{L_t\}^{j+} = \{L_{k_1+1}, L_{k_1+2}, \dots, L_{k_2}\}$.

Summing over $\mathcal{L}^-$, we obtain

$$\begin{aligned} \sum_{\mathcal{L}^-} \sum_{\mathcal{L}^{j+}} P^\pi (\{L_t\}^-, \{L_t\}^{j+} \mid L_1) w (\{L_t\}^{j+}) = \\ \sum_{k_1=D_{min}}^{\infty} P(k_1) \sum_{\mathcal{L}^{j+}} P(L_{k_1+1}, \dots, L_{k_2} \mid L_1) w (\{L_t\}^{j+}), \end{aligned} \quad (\text{D.5})$$

where $P(k_1)$ is the probability that $w(\{L_t\}^-) = k_1$. Since the link state sequence $\{L_{k_1+1}, L_{k_1+2}, \dots, L_{k_2}\}$ is a Markov process, we may use Bayes' rule and the Markov property to write

$$\begin{aligned} P^\pi (L_{k_1+1}, \dots, L_{k_2} \mid L_1) = \\ P(L_{k_1+1} \mid L_1) \dots P(L_{k_2} \mid L_{k_2-1}). \end{aligned} \quad (\text{D.6})$$

Using (D.2)-(D.6) in (D.1), we see that $v^{\pi^*}(s_{\mathbf{D}})$ and $v^{\pi^*}(s_{\mathbf{U}})$ differ only in the terms $P(L_{k_1+1} \mid L_{\mathbf{D}})$ and $P(L_{k_1+1} \mid L_{\mathbf{U}})$. Since $k_1 \geq D_{min}$, the Markov property allows us to say that as $D_{min} \rightarrow \infty$, $P(L_{k_1+1} \mid L_{\mathbf{D}})$ and $P(L_{k_1+1} \mid L_{\mathbf{U}})$ approach the same steady-state probabilities, and therefore,

$$|v^{\pi^*}(s_{\mathbf{D}}) - v^{\pi^*}(s_{\mathbf{U}})| \rightarrow 0.$$

□

Appendix E

Acronyms

API application programming interface

AI artificial intelligence

BN Bayesian network

CAP credit assignment problem

CBR case-based reasoning

CDPS cooperative distributed problem solving

COD configuration and observation database

CSL cognitive specification language

DAG directed acyclic graph

DCOP distributed constraint optimization problem

DEC-MDP decentralized partially observable Markov decision problem

DisCSP distributed constraint satisfaction problem

DSA dynamic spectrum access

DTN delay-tolerant network

EA evolutionary algorithms

GA genetic algorithm

iGA island genetic algorithm

IL independent learning

JAL joint action learning

JPD joint probability distribution

MAC medium access control

MANET mobile ad hoc network

MAS multi-agent system

MDP Markov decision problem

MIS maximal independent set

ML maximum likelihood

MOO multi-objective optimization

MSBN multiply sectioned Bayesian network

MST minimum spanning tree

PGA parallel genetic algorithm

POMDP partially observable Markov decision problem

PSS parallel scatter search

PTS parallel tabu search

PVD path-velocity decomposition

QoS quality of service

RCSS replicated combination scatter search

ROI region of influence

RSS received signal strength

RKRL Radio Knowledge Representation Language

RPSS replicated parallel scatter search

SAN software adaptable network

SDR software defined radio

SINR signal-to-interference-and-noise ratio

SNR signal-to-noise ratio

SPSS synchronous parallel scatter search

TDOA time difference of arrival

UDP/IP Universal Datagram Protocol/Internet Protocol

Bibliography

- [1] R. W. Thomas, L. A. DaSilva, and A. B. MacKenzie, “Cognitive networks,” in *IEEE DySPAN*, (Baltimore, Maryland), pp. 352–360, Nov. 2005.
- [2] J. Mitola, *Cognitive Radio: An Integrated Agent Architecture for Software Defined Radio*. PhD thesis, Royal Institute of Technology (KTH), 2000.
- [3] R. W. Thomas, D. H. Friend, L. A. DaSilva, and A. B. MacKenzie, “Cognitive networks: Adaptation and learning to achieve end-to-end performance objectives,” *IEEE Communications Magazine*, vol. 44, pp. 51–57, Dec. 2006.
- [4] V. Srivastava and M. Motani, “Cross-layer design: a survey and the road ahead,” *IEEE Communications Magazine*, vol. 43, no. 12, pp. 112–119, 2005.
- [5] V. Kawadia and P. R. Kumar, “A cautionary perspective on cross-layer design,” *IEEE Wireless Communications*, vol. 12, no. 1, pp. 3–11, 2005.
- [6] D. D. Clark, C. Partridge, J. C. Ramming, and J. T. Wroclawski, “A knowledge plane for the internet,” in *ACM SIGCOMM*, (New York, NY, USA), pp. 3–10, ACM Press, 2003.
- [7] D. Bourse, M. Muck, O. Simon, N. Alonistioti, K. Moessner, E. Nicollet, D. Bateman, E. Buracchini, G. Chengeleroyen, and P. Demestichas, “End-to-end reconfigurability (E2R II): Management and control of adaptive communication systems,” in *IST Summit 06*, (Mykonos, Greece), 2006.
- [8] P. Demestichas, V. Stavroulaki, D. Bosovic, A. Lee, and J. Strassner, “m@ANGEL: Autonomic management platform for seamless cognitive connectivity to the mobile internet,” *IEEE Communications Magazine*, vol. 44, pp. 118–127, June 2006.
- [9] P. D. Sutton, L. E. Doyle, and K. E. Nolan, “A reconfigurable platform for cognitive networks,” in *ICST CROWNCOM*, (Mykonos Island, Greece), June 2006.
- [10] P. Mähönen, M. Petrova, J. Riihijärvi, and M. Wellens, “Cognitive wireless networks: Your network just became a teenager,” in *IEEE INFOCOM*, (Barcelona, Spain), Apr. 2006.

- [11] J. Jin and K. Nahrstedt, “QoS specification languages for distributed multimedia applications: A survey and taxonomy,” *IEEE Multimedia*, vol. 11, no. 3, pp. 74–87, 2004.
- [12] M. A. L. Thathachar and P. S. Sastry, *Networks of Learning Automata*. Kluwer Academic Publishers, 2004.
- [13] D. H. Friend, R. W. Thomas, A. B. MacKenzie, and L. A. DaSilva, “Distributed learning and reasoning in cognitive networks: Methods and design decisions,” in *Cognitive Networks – Towards Self-Aware Networks* (Q. H. Mahmoud, ed.), pp. 223–246, John Wiley & Sons, 2007.
- [14] D. H. Friend, M. Y. ElNainay, Y. Shi, and A. B. MacKenzie, “Architecture and performance of an island genetic algorithm-based cognitive network,” in *IEEE CCNC*, (Las Vegas, Nevada, USA), January 2008.
- [15] T. G. Dietterich, “Learning and reasoning,” tech. rep., Oregon State University, 2004.
- [16] J. R. Anderson, D. Bothell, M. D. Byrne, S. Douglass, C. Lebiere, and Y. Qin, “An integrated theory of the mind,” *Psychological Review*, vol. 111, no. 4, pp. 1036–1060, 2004.
- [17] J. F. Lehman, J. Laird, and Rosenbloom, “A gentle introduction to SOAR, an architecture for human cognition: 2006 update,” tech. rep., University of Michigan, Ann Arbor, 2006.
- [18] P. Langley and D. Choi, “A unified cognitive architecture for physical agents,” in *National Conference on Artificial Intelligence*, (Boston, Massachusetts, USA), July 2006.
- [19] D. J. Bryant, “Modernizing our cognitive model,” in *Command and Control Research and Technology Symposium*, (Copenhagen, Denmark), September 2004.
- [20] N. R. Jennings, K. Sycara, and M. Woolridge, “A roadmap of agent research and development,” *Journal of Autonomous Agents and Multi-Agent Systems*, vol. 1, no. 1, pp. 7–38, 1998.
- [21] K. P. Sycara, “Multiagent systems,” *AI Magazine*, vol. 19, no. 2, pp. 79–92, 1998.
- [22] G. Weiss, ed., *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. Cambridge: MIT Press, 1999.
- [23] P. J. Modi, *Distributed constraint optimization for multiagent systems*. PhD thesis, University of Southern California, 2003.

- [24] P. J. Modi and S. M. Ali, “Distributed constraint reasoning under unreliable communication,” in *International Joint Conference on Autonomous Agents and Multiagent Systems*, (Melbourne, Australia), July 2003.
- [25] E. Bowring, M. Tambe, and M. Yokoo, “Multiply-constrained dcop for distributed planning and scheduling,” in *AAAI Spring Symposium*, vol. SS-06-04, (Stanford, CA, United States), pp. 25 – 32, 2006.
- [26] Y. Xiang, *Probabilistic Reasoning in Multiagent Systems*. Cambridge: Cambridge University Press, 2002.
- [27] C. Barrett, A. Marathe, M. V. Marathe, and M. Drozda, “Characterizing the interaction between routing and mac protocols in ad-hoc networks,” in *ACM MobiHoc*, (New York, NY, USA), pp. 92–103, ACM Press, 2002.
- [28] W. Watthayu and Y. Peng, “A bayesian network based framework for multi-criteria decision making,” in *International Conference on Multiple Criteria Decision Analysis*, (Whistler, B.C., Canada), August 2004.
- [29] E. Alba, ed., *Parallel Metaheuristics: A New Class of Algorithms*. John Wiley & Sons, 2005.
- [30] S.-C. Lin, W. F. Punch, and E. D. Goodman, “Coarse-grain parallel genetic algorithms: Categorization and new approach,” in *IEEE Symposium on Parallel and Distributed Processing*, (Dallas, Texas, USA), October 1994.
- [31] F. Garcia-Lopez, B. Melian-Batista, J. A. Moreno-Perez, and J. M. Moreno-Vega, “Parallelization of the scatter search for the p-median problem,” *Parallel Computing*, vol. 29, no. 5, pp. 575–589, 2003.
- [32] F. Glover, M. Laguna, and R. Marti, “Scatter search and path relinking: Advances and applications,” in *Handbook of Metaheuristics* (F. Glover and G. A. Kochenberger, eds.), pp. 1–36, Kluwer, 2003.
- [33] M. Gendreau, “An introduction to tabu search,” in *Handbook of Metaheuristics* (F. Glover and G. A. Kochenberger, eds.), Kluwer, 2003.
- [34] T. Crainic, M. Toulouse, and M. Gendreau, “Toward a taxonomy of parallel tabu search heuristics,” *INFORMS Journal on Computing*, vol. 9, no. 1, pp. 61 – 72, 1997.
- [35] E. L. Lawler, J. K. Lenstra, A. H. Rinnooy Kan, and D. B. Shmoys, “Sequencing and scheduling: Algorithms and complexity,” in *Logistics of production and inventory* (S. C. Graves, A. H. G. Rinnooy Kan, and P. H. Zipkin, eds.), vol. 4 of *Handbooks in Operations Research and Management Science*, pp. 445–522, Elsevier, 1993.

- [36] D. B. Shmoys, C. Stein, and J. Wein, “Improved approximation algorithms for shop scheduling problems,” in *ACM-SIAM SODA*, (San Francisco, CA, USA), pp. 148–157, 1991.
- [37] J. Canny and J. Reif, “New lower bound techniques for robot motion planning problems,” in *IEEE Symposium on Foundations of Computer Science*, (Los Angeles, CA, USA), pp. 49–60, October 1987.
- [38] K. Kant and S. Zucker, “Planning collision-free trajectories in time-varying environments: a two-level hierarchy,” in *IEEE International Conference on Robotics and Automation*, vol. 3, (Philadelphia, PA, USA), pp. 1644–1649, Apr 1988.
- [39] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*. New York, NY, USA: W. H. Freeman, 1979.
- [40] S. Chen and K. Nahrstedt, “Distributed quality-of-service routing in ad hoc networks,” *IEEE JSAC*, vol. 17, pp. 1488–1505, Aug 1999.
- [41] E. Bowring, M. Tambe, and M. Yokoo, “Distributed multi-criteria coordination in multi-agent systems,” in *Proceedings of Workshop on Declarative Agent Languages and Technologies*, (Utrecht, Netherlands), July 25–29 2005.
- [42] H. C. Lau and H. Wang, “A multi-agent approach for solving optimization problems involving expensive resources,” in *ACM Symposium on Applied Computing*, (Santa Fe, New Mexico, USA), March 2005.
- [43] A. Cardon, T. Galinho, and J.-P. Vacher, “Genetic algorithms using multi-objectives in a multi-agent system,” *Robotics and Autonomous Systems*, vol. 33, no. 3, pp. 179 – 90, 2000.
- [44] J. Cochran, S.-M. Horng, and J. Fowler, “A multi-population genetic algorithm to solve multi-objective scheduling problems for parallel machines,” *Computers & Operations Research*, vol. 30, no. 7, pp. 1087 – 102, 2003.
- [45] L. Panait and S. Luke, “Cooperative multi-agent learning: The state of the art,” *Autonomous Agents and Multi-Agent Systems*, vol. 11, no. 3, pp. 387–434, 2005.
- [46] G. Weiss, “Distributed reinforcement learning,” *Robotics and Autonomous Systems*, vol. 15, no. 1-2, pp. 135–142, 1995.
- [47] M. Tan, “Multi-agent reinforcement learning: Independent vs. cooperative agents,” in *Readings in Agents* (M. N. Huhns and M. P. Singh, eds.), pp. 487–494, San Francisco: Morgan Kaufmann, 1997.
- [48] C. Watkins and P. Dayan, “Technical note: Q-learning,” *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

- [49] C. Claus and C. Boutilier, “The dynamics of reinforcement learning in cooperative multiagent systems,” in *Conference on Artificial Intelligence*, (Madison, WI, USA), pp. 746 – 52, 1998.
- [50] M. Lauer and M. Riedmiller, “Reinforcement learning for stochastic cooperative multi-agent systems,” in *International Joint Conference on Autonomous Agents and Multi-agent Systems*, (New York City, USA), July 2004.
- [51] W. Buntine, “Theory refinement on bayesian networks,” in *Uncertainty in Artificial Intelligence*, (Los Angeles, California, USA), July 1991.
- [52] W. Lam and A. M. Segre, “A parallel learning algorithm for bayesian inference networks,” *IEEE Trans. on Knowledge and Data Engineering*, vol. 14, no. 1, pp. 93–105, 2002.
- [53] L. de Campos, J. Fernandez-Luna, J. Gamez, and J. Puerta, “Ant colony optimization for learning bayesian networks,” *International Journal of Approximate Reasoning*, vol. 31, no. 3, pp. 291 – 311, 2002.
- [54] L. de Campos and J. Puerta, “Stochastic local algorithms for learning belief networks: searching in the space of the orderings,” in *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, (Toulouse, France), pp. 228 – 39, 2001.
- [55] J. W. Myers, K. B. Laskey, and K. A. DeJong, “Learning bayesian networks from incomplete data using evolutionary algorithms,” in *Genetic and Evolutionary Computation Conference*, (Orlando, Florida, USA), July 1999.
- [56] J. Ocenasek and J. Schwarz, “The distributed bayesian optimization algorithm for combinatorial optimization,” in *Evolutionary Methods for Design, Optimisation and Control with Applications to Industrial Problems*, (Athens, Greece), September 2001.
- [57] A. Aamodt and E. Plaza, “Case-based reasoning: foundational issues, methodological variations, and system approaches,” *AI Communications*, vol. 7, no. 1, pp. 39 – 59, 1994.
- [58] E. Plaza and S. Ontañón, “Cooperative multiagent learning,” in *Adaptive Agents and Multi-Agent Systems: Adaptation and Multi-Agent Learning* (E. Alonso, D. Kudenko, and D. Kazakov, eds.), pp. 1–17, Berlin: Springer, 2003.
- [59] M. V. N. Prasad, “Distributed case-based learning,” in *International Conference on MultiAgent Systems*, (Boston, Massachusetts, USA), July 2000.
- [60] B. Le, T. W. Rondeau, and C. W. Bostian, “Cognitive radio realities,” *Wireless Communications and Mobile Computing*, vol. 7, no. 9, pp. 1037 – 1048, 2007.
- [61] A. Giridhar and P. R. Kumar, “Toward a theory of in-network computation in wireless sensor networks,” *IEEE Communications Magazine*, vol. 44, no. 4, pp. 98–107, 2006.

- [62] A. Scaglione and S. Servetto, “On the interdependence of routing and data compression in multi-hop sensor networks,” *Wireless Networks*, vol. 11, no. 1-2, pp. 149–160, 2005.
- [63] D. H. Friend and A. B. MacKenzie, “Environmentally-friendly secondary network topology control for minimizing outage potential,” in *IEEE DySPAN*, (Chicago, IL), pp. 178–185, October 2008.
- [64] D. H. Friend and A. B. MacKenzie, “Secondary network multichannel topology control for minimizing expected outage potential,” *IEEE Trans. on Mobile Computing*, 2009, submitted.
- [65] A. Ghasemi and E. S. Sousa, “Collaborative spectrum sensing for opportunistic access in fading environments,” in *IEEE DySPAN*, (Baltimore, MD, USA), pp. 131 – 136, November 2005.
- [66] G. Ganesan and Y. Li, “Cooperative spectrum sensing in cognitive radio networks,” in *IEEE DySPAN*, (Baltimore, Maryland, USA), pp. 137–143, Nov. 2005.
- [67] M. McHenry, E. Livsics, T. Nguyen, and N. Majumdar, “XG dynamic spectrum access field test results,” *IEEE Communications Magazine*, vol. 45, pp. 51–57, June 2007.
- [68] A. Annamalai, C. Tellambura, and V. Bhargava, “Simple and accurate methods for outage analysis in cellular mobile radio systems-a unified approach,” *IEEE Trans. on Communications*, vol. 49, pp. 303 – 16, February 2001.
- [69] C. C. Chan and S. Hanly, “Calculating the outage probability in a CDMA network with spatial Poisson traffic,” *IEEE Trans. on Vehicular Technology*, vol. 50, pp. 183 – 204, January 2001.
- [70] R. Menon, R. M. Buehrer, and J. H. Reed, “Outage probability based comparison of underlay and overlay spectrum sharing techniques,” in *IEEE DySPAN*, (Baltimore, MD, USA), pp. 101–109, November 2005.
- [71] R. Ramanathan and R. Rosales-Hain, “Topology control of multihop wireless network using transmit power adjustment,” in *IEEE Infocom*, (Tel Aviv, Israel), pp. 404–413, March 2000.
- [72] M. Marina and S. Das, “A topology control approach for utilizing multiple channels in multi-radio wireless mesh networks,” in *IEEE BroadNets*, (Boston, MA), pp. 381–390, October 2005.
- [73] H. Zhu, K. Lu, and M. Li, “Distributed topology control in multi-channel multi-radio mesh networks,” in *IEEE ICC*, (Beijing, China), pp. 2958–62, July 2008.
- [74] A. Naveed, S. S. Kanhere, and S. K. Jha, “Topology control and channel assignment in multi-radio multi-channel wireless mesh networks,” in *IEEE MASS*, (Pisa, Italy), pp. 1–9, October 2007.

- [75] R. W. Thomas, R. S. Komali, A. B. MacKenzie, and L. A. DaSilva, "Joint power and channel minimization in topology control: A cognitive network approach," in *IEEE ICC*, (Glasgow, Scotland), pp. 6538–6543, June 2007.
- [76] F. F. Digham, "Joint power and channel allocation for cognitive radios," in *IEEE WCNC*, (Las Vegas, NV), pp. 882–7, April 2008.
- [77] T. S. Rappaport, *Wireless Communications: Principles and Practice*. Upper Saddle River, NJ: Prentice Hall PTR, 1996.
- [78] N. Beaulieu and Q. Xie, "An optimal lognormal approximation to lognormal sum distributions," *IEEE Trans. on Vehicular Technology*, vol. 53, pp. 479 – 89, March 2004.
- [79] K.C.Zangi and R. Koilpillai, "Efficient filterbank channelizers for software radio receivers," in *IEEE ICC*, vol. 3, (Atlanta, GA), pp. 1566–1570, June 1998.
- [80] M. Kiessling and S. Mujtaba, "A software radio architecture for multi-channel digital upconversion and downconversion using generalized polyphase filterbanks with frequency offset correction," in *IEEE PIMRC*, vol. 1, (Lisboa, Portugal), pp. 105–109, September 2002.
- [81] L. Pucker, "Channelization techniques for software defined radio," in *SDR Technical Conference*, (Orlando, FL), November 2003.
- [82] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu, "Impact of interference on multi-hop wireless network performance," *Wireless Networks*, vol. 11, pp. 471–487, July 2005.
- [83] M. A. McHenry and A. E. Leu, "Method and system for determining spectrum availability within a network." USPTO Application No. 20070263566, 2007.
- [84] Y. Zhao, L. Morales, J. Gaeddert, K. K. Bae, J.-S. Um, and J. H. Reed, "Applying radio environment maps to cognitive wireless regional area networks," in *IEEE DySPAN*, (Dublin, Ireland), pp. 115–118, April 2007.
- [85] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, pp. 46–57, June 1989.
- [86] D. Pagac, E. M. Nebot, and H. Durrant-Whyte, "An evidential approach to map-building for autonomous vehicles," *IEEE Trans. on Robotics and Automation*, vol. 14, pp. 623–629, August 1998.
- [87] N. Patwari, R. J. O'Dea, and Y. Wang, "Relative location in wireless networks," in *IEEE VTC*, vol. 2, (Rhodes, Greece), pp. 1149–1153, May 2001.

- [88] E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan, “Generating all maximal independent sets: NP-hardness and polynomial-time algorithms,” *SIAM Journal on Computing*, vol. 9, pp. 558–565, August 1980.
- [89] S. Y. Seidel, T. S. Rappaport, S. Jain, M. L. Lord, and R. Singh, “Path loss, scattering, and multipath delay statistics in four European cities for digital cellular and microcellular radiotelephone,” *IEEE Trans. on Vehicular Technology*, vol. 40, no. 4, pp. 721 – 730, 1991.
- [90] M.-J. Jou and G. J. Chang, “The number of maximum independent sets in graphs,” *Taiwanese Journal of Mathematics*, vol. 4, pp. 685–695, December 2000.
- [91] E. L. Lloyd, R. Liu, M. Marathe, R. Ramanathan, and S. S. Ravi, “Algorithmic aspects of topology control problems for ad hoc networks,” *Mobile Networks and Applications*, vol. 10, no. 1, pp. 19–34, 2005.
- [92] R. L. Graham, “Bounds on multiprocessing time anomalies,” *SIAM Journal on Applied Mathematics*, vol. 17, pp. 416–429, March 1969.
- [93] M. Burkhart, P. von Rickenbach, R. Wattenhofer, and A. Zollinger, “Does topology control reduce interference?,” in *ACM MobiHoc*, (Roppongi, Japan), pp. 9–19, May 2004.
- [94] D. H. Friend and A. B. MacKenzie, “Minimum expected cost routing in Markovian MANETs,” *IEEE Trans. on Wireless Comm.*, 2009. in submission.
- [95] D. H. Friend, A. B. MacKenzie, and A. P. Worthen, “Minimum expected cost routing in markovian MANETs,” in *IEEE Sensor, Mesh and Ad Hoc Communications and Networks*, (Rome, Italy), June 2009. submitted.
- [96] S. Jain, K. Fall, and R. Patra, “Routing in a delay tolerant network,” *ACM SIGCOMM*, vol. 34, pp. 145–158, October 2004.
- [97] D. Bernstein, R. Givan, N. Immerman, and S. Zilberstein, “The complexity of decentralized control of Markov decision processes,” *Mathematics of Operations Research*, vol. 27, no. 4, pp. 819–840, 2003.
- [98] M. Abolhasan, T. Wysocki, and E. Dutkiewicz, “A review of routing protocols for mobile ad hoc networks,” *Ad Hoc Networks*, vol. 2, pp. 1–22, January 2004.
- [99] M. Mauve and J. Widmer, “A survey on position-based routing in mobile ad hoc networks,” *IEEE Network*, vol. 15, no. 6, pp. 30–39, 2001.
- [100] Z. Zhang, “Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: Overview and challenges,” *IEEE Communications Surveys*, vol. 8, no. 1, pp. 24–37, 2006.

- [101] A. Ferreira, “Building a reference combinatorial model for MANETs,” *IEEE Network*, vol. 18, no. 5, pp. 24–29, 2004.
- [102] E. Altman, “Applications of Markov decision processes in communication networks,” in *Handbook of Markov Decision Processes: Methods and Applications* (E. Feinberg and A. Shwartz, eds.), pp. 489–536, Springer, 2002.
- [103] P. Wang and T. Wang, “Adaptive routing for sensor networks using reinforcement learning,” in *IEEE International Conference on Computer and Information Technology*, pp. 219–224, September 2006.
- [104] E. Kamar and B. Grosz, “Applying MDP approaches for estimating outcome of interaction in collaborative human-computer settings,” in *Workshop on Multi-Agent Sequential Decision Making in Uncertain Domains*, (Honolulu, HI), pp. 25–32, May 2007.
- [105] C. Goldman, M. Allen, and S. Zilberstein, “Decentralized language learning through acting,” in *Autonomous Agents and Multiagent Systems*, (New York, NY), pp. 1006–1013, July 2004.
- [106] M. Porfiri, D. J. Stilwell, E. M. Bollt, and J. D. Skufca, “Random talk: Random walk and synchronizability in a moving neighborhood network,” *Physica D*, vol. 224, pp. 102–113, December 2006.
- [107] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley-Interscience, 1994.
- [108] J. G. Kemeny and J. L. Snell, *Finite Markov Chains*. D. Van Nostrand Company, 1960.
- [109] A. Beynier and A. Mouaddib, “A polynomial algorithm for decentralized Markov decision processes with temporal constraints,” in *Autonomous Agents and Multiagent Systems*, (Netherlands), pp. 963–969, July 2005.
- [110] J. Tsitsiklis, “Asynchronous stochastic approximation and Q-learning,” *Machine Learning*, vol. 16, no. 3, pp. 185–202, 1994.
- [111] T. Jaakkola, S. Singh, and M. Jordan, “Reinforcement learning algorithm for partially observable Markov decision problems,” in *Advances in Neural Information Processing Systems 7*, pp. 345–352, MIT Press, 1995.