Island Genetic Algorithm-based Cognitive Networks

Mustafa Y. El-Nainay

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. Hussien H. Aly
Dr. Charles W. Bostian
Dr. Luiz A. DaSilva
Dr. Mohamed N. El-Derini
Dr. Madhav V. Marathe
Dr. Scott F. Midkiff

July 1, 2009
Blacksburg, Virginia

Keywords: cognitive networks, dynamic spectrum access, distributed reasoning and learning, island genetic algorithm, channel allocation, power control, flow routing

Island Genetic Algorithm-based Cognitive Networks

Mustafa Y. El-Nainay

(ABSTRACT)

The heterogeneity and complexity of modern communication networks demands coupling network nodes with intelligence to perceive and adapt to different network conditions autonomously. Cognitive Networking is an emerging networking research area that aims to achieve this goal by applying distributed reasoning and learning across the protocol stack and throughout the network. Various cognitive node and cognitive network architectures with different levels of maturity have been proposed in the literature. All of them adopt the idea of coupling network devices with sensors to sense network conditions, artificial intelligence algorithms to solve problems, and a reconfigurable platform to apply solutions. However, little further research has investigated suitable reasoning and learning algorithms.

In this dissertation, we take cognitive network research a step further by investigating the reasoning component of cognitive networks. In a deviation from previous suggestions, we suggest the use of a single flexible distributed reasoning algorithm for cognitive networks. We first propose an architecture for a cognitive node in a cognitive network that is general enough to apply to future networking challenges. We then introduce and justify our choice of the island genetic algorithm (iGA) as the distributed reasoning algorithm.

Having introduced our cognitive node architecture, we then focus on the applicability of the island genetic algorithm as a single reasoning algorithm for cognitive networks. Our approach is to apply the island genetic algorithm to different single and cross layer communication and networking problems and to evaluate its performance through simulation. A proof of concept cognitive network is implemented to understand the implementation challenges and assess the island genetic algorithm performance in a real network environment. We apply the island genetic algorithm to three problems: channel allocation, joint power and channel allocation, and flow routing. The channel allocation problem is a major challenge for

dynamic spectrum access which, in turn, has been the focal application for cognitive radios and cognitive networks. The other problems are examples of hard cross layer problems.

We first apply the standard island genetic algorithm to a channel allocation problem formulated for the dynamic spectrum cognitive network environment. We also describe the details for implementing a cognitive network prototype using the universal software radio peripheral integrated with our extended implementation of the GNU radio software package and our island genetic algorithm implementation for the dynamic spectrum channel allocation problem. We then develop a localized variation of the island genetic algorithm, denoted LiGA, that allows the standard island genetic algorithm to scale and apply it to the joint power and channel allocation problem. In this context, we also investigate the importance of power control for cognitive networks and study the effect of non-cooperative behavior on the performance of the LiGA.

The localized variation of the island genetic algorithm, LiGA, is powerful in solving node-centric problems and problems that requires only limited knowledge about network status. However, not every communication and networking problems can be solved efficiently in localized fashion. Thus, we propose a generalized version of the LiGA, namely the K-hop island genetic algorithm, as our final distributed reasoning algorithm proposal for cognitive networks. The K-hop island genetic algorithm is a promising algorithm to solve a large class of communication and networking problems with controllable cooperation and migration scope that allows for a tradeoff between performance and cost. We apply it to a flow routing problem that includes both power control and channel allocation. For all problems simulation results are provided to quantify the performance of the island genetic algorithm variation. In most cases, simulation and experimental results reveal promising performance for the island genetic algorithm.

We conclude our work with a discussion of the shortcomings of island genetic algorithms without guidance from a learning mechanism and propose the incorporation of two learning processes into the cognitive node architecture to solve slow convergence and manual

configuration problems. We suggest the cultural algorithm framework and reinforcement learning techniques as candidate leaning techniques for implementing the learning processes. However, further investigation and implementation is left as future work.

This material is based upon work supported by the National Science Foundation under Grant No. 0519959. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. This work is also supported by the Egyptian Cultural Affairs and Missions Sector, the Alexandria University, and the VT–MENA program.

Acknowledgments

All praise be to Allah (God), who has guided me through this work. The past four years of my life were the toughest and except for Allah's help and protection with prayers for me from all my family members, I would have never made it to this moment. Through my past four years, I had to make many decisions that I knew will affect my future life and I was depending on Allah to guide me to the right choices. Although my family and I had passed through very hard times, I have always believed that Allah is guiding me to the best. Day after day, I discovered this clearly and cannot be happier with any other way except the way Allah guided me.

When I was in my early 20's, I was thinking that my parents' support is a natural duty and that they do what all other parents do. Hearing stories from friends and going through my own experience as a father, I have clearly discovered that my parents are different. They do their best and sacrifice their own to support me with anything I need because they want me to be happy even when this sometimes conflicts with their own happiness. I would like to express my deepest love and appreciation to my father and my mother. I pray to Allah to protect them and to help me be a reason for a smile and satisfaction on their faces and in their hearts with my success.

Sharing all hard times with me and giving up her own career for more than three years to support me and our son, I am sincerely thankful and grateful to my wife for being extremely patient and supportive to me during my Ph.D. work. I would like to tell her that I could not make it without your support and love. I would like to thank my son for bearing sometimes

weeks and months with a nervous indoor dad and no outdoor fun and for being understanding of the importance of my work and need for privacy. I am grateful to all my family for praying for me during my hard times especially my father-in-law, mother-in-law, brother, and sister.

The Ph.D., in my opinion, is mostly about choosing a good advisor. It has been my great fortune to work with Dr. Allen MacKenzie, not only because of his insightful ideas and endless support but also because of his unbelievable understanding of the different cultures and religions of his students. His confidence in my research abilities and my choices was greatly appreciated. His analytical view had and will have a continuous great impact on my way of thinking. I would like to express my especial gratitude to his support to his students' lives and families. If there is one portion to mention, it is his approval for me to take my vacation every year during the month of Ramadan to give me and my family the opportunity to spend this special month in my religion, Islam, with our families in Egypt although this period coincides with the beginning of the Fall semester in the past two years.

I am thankful to my committee members: Dr. Charles Bostian, for his invaluable comments and for leading the funding project with his wisdom and expertise. Project meetings were always a valuable source of knowledge about my colleagues' research work. I would like to thank Dr. Madhav Marathe, for his help and his productive feedback; Dr. Luiz DaSilva, for his inspiring and constructive criticisms; Dr. Scott Midkiff, for his expertise and support. I would especially like to thank Dr. Nazih El-Derini and Dr. Hussien Aly for helping me pass through this very tough period of my life and for being always ready to offer advice and support. I will never forget that Dr. Hussien was the one who encouraged me to travel abroad to pursue my Ph.D. and that Dr. Nazih is always like a father for me with his continuous sincere advice and support. I would like to thank my entire committee members for their flexibility when I was scheduling my preliminary exam and my final defense dates.

When I was about to finish my master back in 2005, Dr. Yasser Hanafy, the director of the VT-MENA program in Egypt, told me about the VT-MENA program. I immediately had a very good feeling toward the program especially with the residence flexibility that I could

use to accommodate my family needs. I would like to thank Dr. Yasser Hanafy and the Arab Academy for this great offer and for supporting me all the way to the degree's conclusion. I would also like to express my sincere gratitude to Dr. Sedki Riad who welcomed my family and me in Blacksburg and made us feel at home with his invaluable contributions to VT-MENA program and to the Muslim community in Blacksburg. I do not want to forget to thank Thierno Kane, Sherif Fadel, and all people who helped me settle during my first weeks in Blacksburg.

I would like to thank all my friends for their support and encouragements. I would like to thank Judy Hood for her continuous support to me and all my project colleagues. I would also like to thank all my colleagues for their contributions in my/our work. I would especially like to mention Dr. Daniel Friend, Amr Hilal, Feng (Andrew) Ge, and Yongshing (Sam) Shi for sacrificing their own time and effort helping me through parts of my Ph.D. work.

I would like to acknowledge the National Science Foundation, the Alexandria University, the Egyptian Cultural Affairs and Missions Sector, and the Arab Academy for supporting me financially throughout my Ph.D. work. From the Egyptian Cultural and Educational Bureau, I am especially thankful to Dr. Mohamed Hamza, the Cultural Attach, and Dr. Maha Kamel, the Deputy Director, for their friendly support and help.

Contents

List of Figures

List of Tables

List of Algorithms

Part I

Cognitive Networks: Introduction and Architecture

Chapter 1

Introduction

The heterogeneity and complexity of modern networks urge the need for coupling network nodes with intelligence to perceive and adapt to different network conditions. Cognitive Radios (CRs), with the ability to observe the surrounding network environment and reconfigure to adapt to network changes, are one of the most promising solutions. However, in order to achieve network-wide goals instead of node-centric goals, cognitive radios need to be organized together in a way that allows for cooperation and distributed reasoning and learning. A Cognitive Network (CN) allows for such cooperation by integrating distributed reasoning and learning across the protocol stack and throughout the network. In this dissertation, we propose an architecture for a cognitive node in a cognitive network. Furthermore, we investigate the applicability of island Genetic Algorithms (iGA) and our developed localized variations of the (iGA) as distributed reasoning algorithms in cognitive networks.

This chapter is organized as follows: We introduce an overview of our work presented in this dissertation in Section 1.1. Section 1.2 discusses the motivations behind our research work on cognitive networks. The scope of our work is presented in Section 1.3. Our research contributions are then summarized in Section 1.4. Finally, the dissertation outline is presented in Section 1.5.

1.1 Overview

Today's networks are characterized by rigidity towards environmental changes. Most are designed to work in specified conditions with limited adaptability, leaving the complexity of network configuration to humans. Spectrum access is a typical example where much of the spectrum is wasted either because of a fixed spectrum assignment policy or because of poor cooperation and adaptation of nodes operating in the unlicensed part [1]. For example, a WiFi access point may suffer from interference in one channel (usually channel 6, the default channel) while other channels have little or no interference. The user has to sense the performance degradation and manually reconfigure the access point to use another channel. On the other hand, the idea behind next generation networking is to use networks to transport all information and services. This requires networks to perform autonomously, understand the surrounding environment, and adapt to network changes.

Cognitive Radio (CR), with the ability to observe the surrounding network environment and reconfigure to adapt to network changes, is one of the most promising solutions. The core of cognitive radio as described by Mitola is the cognitive cycle, which consists of six processes: observe, orient, plan, decide, act, and learn [2]. These processes allow the cognitive radio to perceive user and application needs and network conditions, and manipulate the protocol stack to better satisfy the perceived needs. This results in better performance through better utilization of network resources. However, in order to achieve network-wide goals instead of node-centric goals, the cognitive radios need to be organized to allow for cooperation and distributed reasoning and learning.

The Cognitive Network (CN) introduced by Thomas *et al.* allows for network intelligence by integrating distributed reasoning and learning across the protocol stack and throughout the network [3]. Thus, CN can be seen as promising solution for future network challenges. Various cognitive node and cognitive network architectures with different levels of maturity have been proposed in the literature. All of them adopt the idea of coupling the network devices (often nodes with software defined radios) with sensors to sense network conditions,

artificial intelligence algorithms to solve problems, and a reconfigurable platform to apply solutions. However, most of the current work is dedicated to a single application (dynamic spectrum access), and most of the current work focuses on node-centric rather than network-wide solutions.

In this dissertation, we propose an architecture for cognitive network nodes and suggest the island Genetic Algorithm (iGA) as a distributed reasoning algorithm for this architecture. We develop localized variations of the iGA that allow for controllable cooperation range. We apply the iGA and its localized variations to various single and cross layer problems and study their performance through simulations and experiments. Problems with using distributed reasoning algorithms without learning mechanisms are identified and the incorporation of potential learning methods is suggested but left as future work.

1.2 Motivation and Objective

Modern networks' complexity along with the tremendous increase in wireless network usage demands a network that can perform autonomously with the minimum possible user intervention. The heterogeneity and dynamic environment of these networks motivates the development of network devices that can perceive and adapt to different network conditions. New regulatory trends, e.g. secondary easement rights, require better resource management algorithms. All these are challenges facing future networks.

Cognitive Networks (CNs) with their ability to work in a distributed and possibly cooperative manner to solve network-wide problems are a potential solution to the aforementioned challenges. CNs apply a cognition cycle that requires sensing the surrounding environment and adapting to network changes. New technologies like Software Defined Radio (SDR) provide the ability to sense and reconfigure, while artificial intelligence provides reasoning and learning algorithms to the CN.

This dissertation has two main objectives. The first objective is to study the challenges of

CNs and propose an architecture for a cognitive node in a cognitive network that can face current and future challenges. The other objective is to investigate the applicability of the island Genetic Algorithm (iGA) as a distributed CN reasoning algorithm. The iGA is a promising parallel version of the Genetic Algorithm (GA) that can be used for distributed reasoning in a CN [4]. We have also identified problems with using the standard iGA and developed flexible localized variations of it to better fit the characteristics and requirements of CNs.

1.3 Scope of Work

The Cognitive Network (CN) concept can be applied in various network contexts. Moreover, cognitive networking is a multidisciplinary field that requires knowledge in communications, networking, and artificial intelligence. We will focus our research and implementation on applying the cognitive network concept to ad hoc networks in dynamic spectrum environments. Targeting ad hoc networks implies that all network nodes have similar capabilities and functionalities within the CN. We do not assume the existence of any infrastructure, central servers, or nodes with administration responsibilities. In a dynamic spectrum environment, the set of available channels is location and time dependent. Each node may sense a different set of available channels because of geographic variations in primary user activity.

In this dissertation, we propose an architecture for a cognitive node in a CN that takes into account current and future challenges facing CNs and facilitates meeting CN expectations. After that, we focus on the reasoning process of the CN. In Chapter 3, we justify our choice of the island Genetic Algorithm (iGA) as the CN distributed reasoning algorithm. We then investigate the applicability of the iGA in CNs by applying different variations of the iGA to various communication and networking problems and evaluating their performance through simulations and experiments.

In all our work, we assume a cooperative network. Each node is willing to share information

with other network nodes and work toward network-wide objectives rather than selfish node objectives. In Chapter 6, we study the effect of non-cooperative behavior on CN performance in terms of channel and power allocation efficiency. Except for this simulation study, we leave further study of non-cooperative behavior to future work.

1.4 Summary of Contributions

The major contributions of this dissertation are:

1. We present a simple taxonomy of research coupling intelligence techniques with communication and networking and a survey of previous work on cognitive node and cognitive network architectures.

2. We propose an architecture for a cognitive node in a cognitive network that is general enough to solve future networking challenges. We propose the island genetic algorithm (iGA) as the distributed reasoning algorithm for the CN. To the best of our knowledge, this is the first work in the area of cognitive networks to focus on the application of the iGA as a single reasoning algorithm to tackle a wide variety of communication and networking problems.

3. We formulate the channel allocation, joint power and channel allocation, and flow routing problems in a way that is unique to dynamic spectrum cognitive network.

4. We investigate the applicability of iGA as the distributed reasoning algorithm for a CN by applying the iGA and our localized variations, Localized iGA and K-hop iGA, to the formulated problems, studying the performance through computer simulations, and comparing the performance to other applicable techniques.

5. We describe an implementation of a CN using commercial off-the-shelf (COTS) hardware and provide experimental results.

6. We describe possible improvements to the proposed techniques through the application of learning algorithms and present candidate techniques.

1.5 Dissertation Outline

The dissertation is structured in three main parts:

- Part I introduces our work and presents our cognitive node architecture. The background and related works on cognitive radio and cognitive network work are surveyed in Chapter 2. Our proposed cognitive node architecture is described in Chapter 3, highlighting the distributed reasoning process. We conclude this part by discussion for reasons behind our choice for the genetic algorithm (GA) in general and the island genetic algorithm (iGA) in particular as the basis for the distributed reasoning algorithm.

- Part II applies the standard iGA to the Dynamic Spectrum Access (DSA) channel allocation problem and studies the performance through simulations in Chapter 4. We chose this problem as our first cognitive network application because DSA is the focal application in most related work. Chapter 5 presents our proof of concept cognitive network implementation using commercial off-the-shelf (COTS) hardware and open source and our developed software components.

- Part III continues to show the applicability of the iGA by applying it to more complex cross layer communication and networking problems. Having identified the scalability problem of the standard iGA in Chapter 4, Chapter 6 introduces the Localized iGA (LiGA), a scalable variation of the iGA that is hence more promising as a distributed reasoning algorithm for CNs. The LiGA is then applied to the cross layer channel allocation and power control problem. The performance of the LiGA is evaluated through simulations. Chapter 7 then introduces a more flexible version of the LiGA, K-

hop iGA, which accommodates a large class of communication and networking problems with its adjustable cooperation range. The K-hop iGA is applied to the flow routing problem and simulation results compare its performance to theoretical bounds.

Finally, Chapter 8 provide general conclusions and future work directions.

Chapter 2

Background on Cognitive Radios and Cognitive Networks

This chapter, together with Sections 3.4 and 3.5, provides the necessary background for the terms, techniques, and technologies used in this dissertation. First, a simple taxonomy of current research work that couples intelligence techniques with communication and networking is presented in Section 2.1. Next, foundations and some potential applications for cognitive radio and cognitive network are introduced in Sections 2.2 and 2.3. A survey of related work in cognitive network research is presented in Section 2.3.2. Finally, Section 2.4 positions our work within the related work and describes the relations and differences between our work and previous work.

2.1 Taxonomy of Related Work

Current research coupling communication and networking with intelligence may be classified into three types: cognitive radios, cognitive radio networks, and cognitive networks. This taxonomy is based on which protocol layers are involved, the scope of the goals that drive

decision-making in the intelligent/cognitive processing, and the extent to which multiple nodes in the network are involved in intelligent/cognitive processing. This taxonomy is simple and more work is needed to create a more detailed one. This taxonomy also focuses on research evolving from the Software Defined Radio (SDR) technology and does not encompass all smart or intelligent network research.

Cognitive Radio (CR) research concentrates mainly on coupling the radio (physical layer and some medium access control (MAC) and/or link functionalities) with cognition within a single device. The general goal is to optimize the radio link for the current wireless environment. Performance is judged relative to individual radio goals and does not consider the goals of other radios. The cognitive process is implemented at each node and works independently of other nodes.

Cognitive Radio Networks (CRNs) use distributed and cooperative algorithms to achieve better performance and/or reduce node effort through communication among radio nodes. The focus remains on the physical and link layers, but radios now cooperate to share information. Cognition is still largely localized to the radio, though radios may share observations to improve the overall view of the network state.

Cognitive Networks (CNs) extend CR and CRN scope by integrating distributed reasoning and learning across the protocol stack and throughout the network. CNs organize network nodes to allow for cooperation and distributed reasoning and learning to achieve network-wide goals instead of node-centric goals. The cognitive process is performed collectively within the network. Our research falls under this category. Next sections introduce the foundations and some possible applications of CRs, CRNs, and CNs.

2.2 Cognitive Radio Foundations and Applications

In this section, a brief introduction of the cognitive radio and its history is presented followed by a discussion of its potential applications.

2.2.1 Software Defined Radio

A Software Defined Radio (SDR) is a radio that migrates hardware functionalities to the software domain. The goal is to produce a radio that can receive and transmit a new radio protocol just by running new software. This allows easy changes of the radio's fundamental characteristics including modulation type, operating frequency, etc.

The term "software defined radio" was coined by Joseph Mitola in 1991 [5]. However, software defined radios have their origins in the defense sector in both the U.S. and Europe since the late 1970s. One of the first software radio initiatives was a U.S. military project named SpeakEasy [6]. The primary goal of the SpeakEasy project was to use programmable processing to emulate more than 10 existing military radios, operating in frequency bands between 2 and 2000 MHz. Further, another design goal was to easily incorporate new coding and modulation standards in the future, so that military communications could keep pace with advances in coding and modulation techniques.

The idealized form of a SDR consists of an RF front end, analog-to-digital converter/ digital-to-analog converter, and a computer and/or a programmable hardware where digital signal processing that replaces several analog hardware stages is performed. Significant amounts of signal processing are then handed over to a general purpose processor or to a programmable hardware, rather than being done using special-purpose hardware. Such a design produces a radio that can receive and transmit a different form of radio protocol (sometimes referred to as a waveform) just by running different software. This comes with the cost of lower performance (at least until this moment) because of the performance difference between the dedicated special-purpose components used in conventional radios and the limited number/capabilities of shared general-purpose processors. This performance degradation is what directs most researchers to use field-programmable gate array (FPGA), or similar programmable hardware, in order to move some of the signal processing back to the hardware domain.

The flexibility of SDR motivates researchers to think about observing and adapting to the

surrounding environment conditions by controlling the radio parameters from the software domain introducing the cognitive radio concept.

2.2.2 Cognitive Radio

As defined in [7], a Cognitive Radio (CR) is an intelligent wireless communication system that is aware of its surrounding environment (i.e., outside world), learns from the environment and its previous actions, and adapts its internal states to statistical variations in the incoming radio frequency (RF) stimuli by making corresponding changes in certain operating parameters (e.g., transmit-power, carrier-frequency, and modulation strategy) in real-time. This definition reflects the focus of most cognitive radio research which concentrates on coupling the radio (physical layer and some medium access control (MAC) and/or link functionalities) with cognition within a single device. The goal is to optimize the radio for the current wireless environment. Performance is judged relative to individual radio goals and does not consider goals of other radios.

A more general idea is introduced by Mitola, who was the first to introduce the cognitive radio concept in [2]. Mitola couples the SDR with a cognitive cycle that consists of six phases as shown in Fig. 2.1. Changes in the environment and/or the performance of the radio act as stimuli that initiate a new cognitive cycle. The cognitive radio then orients itself by determining the priority associated with the cognitive cycle initiating stimulus. Depending on the priority of the stimulus, the cognitive radio may either acts immediately for high priority stimuli, decides then acts for medium priority stimuli, or plans first then decides and acts for non-emergency stimuli. Planning consists of plan generation and the "Decide" phase selects among candidate plans. Based on the observations and decisions, the cognitive radio then learns about the effectiveness of the previous decisions. These processes allow the cognitive radio to perceive user and application needs and network conditions and to manipulate the protocol stack accordingly to satisfy the perceived needs. This results in better performance through better utilization of network resources.

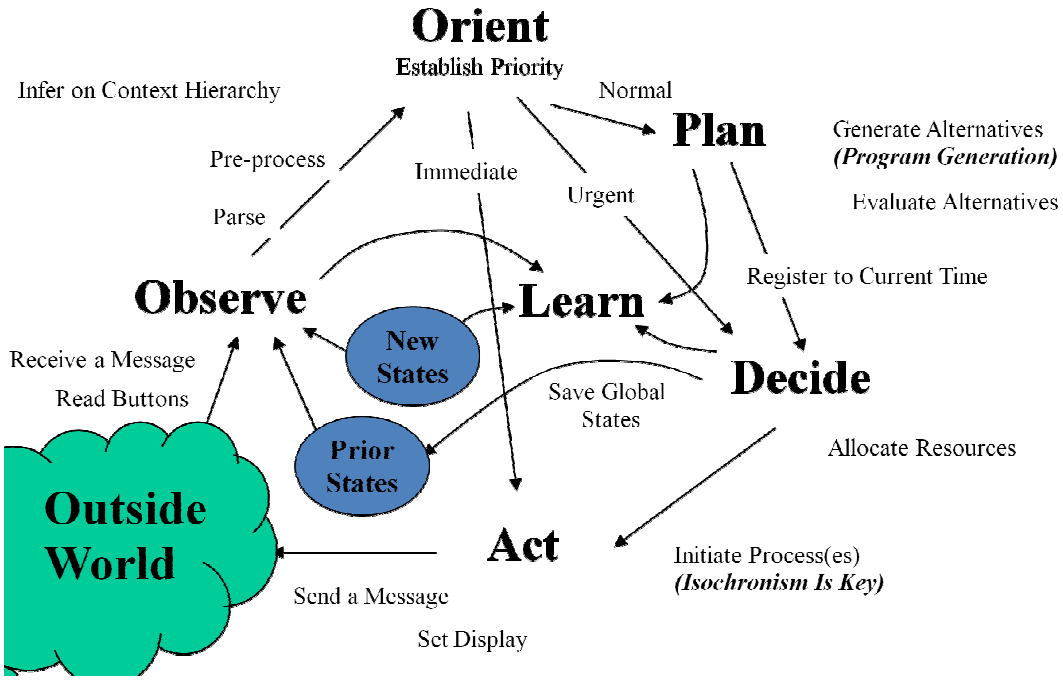

Figure 2.1: J. Mitola III, "Cognitive Cycle", Reprinted, with permission, from J. Mitola III, *Cognitive Radio: An Integrated Agent Architecture for Software Defined Radio*. PhD thesis, Royal Institute of Technology (KTH) Stockholm, Sweden, 2000.

The flexibility of the SDR and the intelligence of the cognitive cycle give the CR its promise as a solution to serious problems such as spectrum scarcity [8] and radio interoperability [9] [10].

2.2.3 Cognitive Radio Network

Cognitive radio nodes can group together to form a cognitive radio network. Cognitive Radio Networks (CRNs) use distributed and cooperative algorithms to achieve better performance and/or reduce node effort through communication among radio nodes. The focus remains on the physical and link layers, but radios now cooperate to share information. Cognition is still largely localized to the radio, though radios may share observations in order to improve the overall view of the network state.

One of the first applications for the CRNs was cooperative spectrum sensing [11] [12] [13]. Spectrum sensing is necessary for cognitive radio nodes to find and utilize available spectrum holes. The general goal of cooperative spectrum sensing is to reduce the overhead for each node to sense the whole available spectrum range by allowing CRs to share sensing information together. Cooperative spectrum sensing also improves sensing performance due to signal diversity and subsequently it improves the detection of primary (licensed) users who own the primary rights to use this spectrum. This way the cognitive radio nodes of the network can distribute the spectrum sensing task and then share the sensing information together to achieve better detection rate and reduce the sensing overhead.

2.2.4 Applications

Cognitive Radios (CRs) have many possible applications in different contexts. For example in cellular networks, CR can be used to build a universal cellular phone that can be used anywhere in the world independent of the deployed cellular technology. CRs can also be used to provide quality of service requirements efficiently to all network users. However, most of the research work is currently focusing on two problems. The first and most popular problem

is dynamic spectrum access for which CR is seen as the enabling technology. The second problem is the radio interoperability problem which has many impacts on first responders and military applications. The next two subsections provide some more detail about these two problems.

2.2.4.1 Dynamic Spectrum Access

The Dynamic Spectrum Access (DSA) paradigm has been suggested to solve the spectrum scarcity problem in unlicensed bands due to the tremendous increase in wireless network users and applications. Although most of the licensed spectrum is idle at any given time [1] [14], many wireless applications/users compete for a small portion of the spectrum, namely the unlicensed part. The underutilization of the licensed spectrum results from traditional spectrum management regulations that grant exclusive access over a frequency band to a licensed user (primary user) and prevent any other user (primary or secondary) from using this band even if it is idle most of the time. This exclusive access to licensed spectrum is being reconsidered by regulators around the world as they consider allowing opportunistic secondary access to the licensed spectrum provided that secondary users do not interfere with the access of primary users. Cognitive Radio (CR) is seen as an enabling technology for dynamic spectrum access with its ability to observe the surrounding network environment and reconfigure itself to adapt to network changes.

As described in [14], CR technology enable dynamic spectrum access by supporting the following functions:

- *Spectrum sensing*: CRs have to detect spectrum holes in their supported spectrum range and possibly share these spectrum holes with other users. Spectrum sensing can be performed independently at each node or in a distributed cooperative manner. Cooperative spectrum sensing improves the detection accuracy and reduces the detection time and the overhead of the spectrum sensing process. A spectrum hole here means a frequency band in which no primary users currently are operating. This implies that

the spectrum sensing must be augmented with a signal classification process to decide whether an existing signal is a primary user or a secondary user.

- *Spectrum management*: After sensing the spectrum for spectrum holes, cognitive radios should decide on the best frequency band to meet user communication Quality of Service (QoS) requirements.

- *Spectrum mobility*: In case a spectrum change is needed either to move to a better band or to evacuate the current band to primary users, CRs need to maintain seamless communication during the frequency change.

- *Spectrum sharing*: CRs also need to share the available spectrum holes fairly with other secondary users. This is a major challenge because it requires a degree of cooperation among all coexisting nodes.

DSA is by far the most explored application in CR research. However, CR technology has many other potential applications.

2.2.4.2 Radio Interoperability

In many emergency situations, first responders using radios operating in different frequency ranges and/or using different signaling need to cooperate together. Cognitive Radios (CRs) can solve this problem by bridging different radio standards together and providing techniques for interoperability. Instead of using several heavy radios, one CR can provide radio interoperability over a wide range of frequency. In [15], different possible CR governmental, military, public safety, and general commercial applications are discussed that all focus on solving the interoperability issues using CRs. In [9], a platform that combines a commercial off-the-shelf (COTS) software defined radio (SDR) and a cognitive engine is used to provide low-cost, multi-band, multi-mode operation for interoperability. Similar SDR approach in which the radio uses a small number of RF front ends, each with relatively wide band-

width and with tuning range sufficient to access most spectrum relevant to public safety communications is presented in [16].

Cognitive radios have many benefits and wide variety of applications. However, in order to achieve network-wide goals instead of node-centric goals, the cognitive radios need to be organized together in a way that allow for cooperation and distributed reasoning and learning. A cognitive network allows for such cooperation by integrating distributed reasoning and learning across the protocol stack and throughout the network. The next section presents the definition of a cognitive network and surveys related cognitive network research work.

2.3 Cognitive Network

Work on CRs and CRNs focuses mainly on physical and link layer node-centric goals, with possible cooperation among nodes in a CRN. However, the heterogeneity and complexity of modern communication networks demands a more thorough solution to achieve network-wide goals instead of node-centric goals. A Cognitive Network (CN), first defined by Thomas *et al.* in [3], appears as a promising candidate to solve modern communication networks' problems by integrating distributed reasoning and learning across the protocol stack and throughout the network.

Various cognitive node and cognitive network architectures have been proposed in the literature. All of them incorporate the network devices with performance monitors, intelligence, and a reconfigurable platform. This section presents the CN definition and surveys existing research work in the area of CNs. The next section positions our research work within the related work.

2.3.1 Definition

In [3], the idea of the SDR is generalized to form a cognitive network (CN). A definition of the CN is given that describes the main expectations of a CN:

> "A cognitive network has 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."

This definition shifts the node-centric goals of the CR to more general end-to-end goals that necessitate cooperation among network nodes. Moreover, it shifts the node intelligence of CR to network intelligence. The CN still applies a cognitive loop to perform reasoning and the CN learns from previous observations and decisions to make better faster future decisions in similar situations.

2.3.2 Related Work

Various cognitive node and cognitive network architectures are proposed in previous work. Most of them adopt the idea of coupling a flexible reconfigurable network stack with one or more cognitive algorithms.

In [17], Mähönen *et al.* discuss the architectural challenges associated with extending the cognitive radio to a cognitive network together with some possible solutions to each challenge. The Cognitive Resource Manager (CRM) is presented as a framework to perform network-wide optimizations for the communication stack as a whole, with global knowledge through information exchange among CRM instances on different nodes. This is an early work with few details about the modular architectural design of the CRM that is necessary to handle its various and complex functionalities.

In [3], Thomas *et al.* present a high level framework that illustrates the relationships between the functional units in a cognitive network. The authors propose cognitive network architecture of three main layers: User/Application requirements, cognitive process, and software adaptable network. The top-level component, user/application requirements, includes the end-to-end goals that drive the cognitive process. The cognitive process provides the node with the required intelligence. The software adaptable network is the reconfigurable radio component with the interfaces required to relay observations to the cognitive process and accept reconfiguration commands from the cognitive process.

In [18], Wu and Niemegeers propose a cognitive architecture for incorporating cognition into personal networks to provide personal services and network management. Besides being intelligent to manage itself, the proposed architecture is capable of learning the preferences of its owner, reasoning about what the owner intends to do, and acting proactively. The proposed architecture consists of three layers: the device layer, the cognitive layer, and the service layer. The device layer observes internal and external resources and provides this information to the cognitive layer. It executes actions decided by the cognitive layer. The service layer uses information from the cognitive layer to adapt the personal network to achieve its goals. The key component is the cognitive layer which in turn consists of five components: context cognition, personalization recognition, resource recognition, network recognition, and cognition management. As a whole, the cognitive layer is responsible for receiving observed resources' information from the device layer, learning, reasoning under uncertainty, and adapting the personal network to better serve the current situation of the user. Our work, in contrast, focuses on applying and improving distributed reasoning algorithms in cognitive networks.

The End-to-End Reconfigurability (E2R) project [19] aims at building a cognitive architecture that consists of reconfigurable elements and intelligent management functionality. In this article the management platform of the cognitive architecture is presented, namely m@ANGEL. m@ANGEL (management) entities are organized in a hierarchy of a two tiers. At the lower tier, each entity is device-specific that manages a specific reconfigurable element

while the upper tier entities control a network segment and the interface with the backbone network. Each m@ANGEL entity works autonomously or cooperatively with other entities and reconfigures the controlled element by changing parameter values, selecting new algorithms, or through the introduction of new executable code. Each m@ANGEL entity consists of several components that provide the means for monitoring current configurations; discovering alternate configurations; acquiring contextual information; describing user profiles and network operator goals and agreements; conducting resource and service brokerage; and negotiating, selecting, and implementing reconfigurations. Similarly, in [20], Sutton *et al.* divide a cognitive network node into two main components, a reconfigurable node and a cognitive engine, with the focus of the paper being the reconfigurable node. The reconfigurable node is a flexible platform that has the ability to observe network conditions and to reconfigure the entire protocol stack. The cognitive engine is the intelligent component that uses network-wide observations to decide on the best node configuration and to learn from past experiences. The reconfigurable node is designed using component-based design with interfaces to observe network conditions and notify the cognitive engine by either push or pull mechanisms. The reconfigurable node has interfaces that support different levels of reconfiguration, ranging from changing parameters within a component of a layer to the replacement of the entire stack.

In [21], Rondeau *et al.* provide a description of a cognitive engine that fits with the previous model. The cognitive engine uses a genetic algorithm called the Wireless System Genetic Algorithm (WSGA) to model the communication stack as an organism and optimize its performance through genetic and evolutionary processes. In the WSGA, each communication stack layer/sub-layer behavior is interpreted as a set of layer operation parameters defined by traits encapsulated in the genes of a chromosome. The WSGA analyzes the chromosome's fitness by considering a set of fitness functions defined by performance evaluations of the current communication stack. Each fitness function is weighted to represent the relative importance the user has associated with each objective. The stopping condition for deciding when an optimal or sufficient solution has been obtained is based on the user's QoS and

application requirements. A joint description of both reconfigurable node and cognitive engine of previous two teams is presented in [22], together with some initial experiments focusing on the radio level. Our work builds on top of the work presented in [20] by assuming the existence of such a reconfigurable platform and extends the work presented in [21] to the networking level by applying distributed reasoning techniques.

In [23], Raychaudhuri *et al.* present an architectural framework called CogNet for research into architectural tradeoffs and protocol design approaches for cognitive radio networks. CogNet aims at transforming a set of cognitive radios into a cognitive network and integrating cognitive radio networks into the global Internet. CogNet is designed to support a number of capabilities including fast spectrum scanning, fast physical-layer adaptation, dynamic spectrum coordination, fully programmable MAC layer, ad hoc cluster formation, and cross layer adaptation. This is done by utilizing a number of novel inter-module interfaces and protocols including a Global Control Plane (GCP) which is a cross layer network management overlay that monitors and adapts data communications, an API for physical layer adaptation on a per-packet basis, spectrum coordination protocols to facilitate dynamic spectrum sharing among radio nodes, a flexible MAC framework that permits dynamic selection of channel on a per-packet basis, and network layer protocols to support service discovery, naming, addressing and routing. In [24], Pawelczak *et al.* propose a network architecture for emergency networks using cognitive radios and list expected emergency services and system requirements. A similar work with more general objectives is presented in [25]. In this paper, Peddemors *et al.* present a conceptual view on the incorporation of cognitive processing capabilities in future generation computer systems. The paper points to the possible use of this system to improve the performance of data communication by adapting and reacting to network changes.

A broad review of research in cognitive radio, cognitive networks, and dynamic spectrum access is provided in [14] and more recent reviews are provided in [26] and [27]. In general, most of the previous works still in an early stage with the main focus still on radio level problems. Most of the current work is focused on dynamic spectrum access, and most of the

current work applies node-centric solutions rather than distributed network-wide solutions.

2.3.3 Reasoning and Learning

Two of the key distinguishing features of the CN are reasoning and learning. Besides having a reconfigurable node that is able to apply different possible solutions, ranging from a parameter change to a layer or the entire stack change, and having sensors that can observe the surrounding environment and picture its state, CN should also have the capabilities of reasoning to find good solutions and learning from past experience. The dynamic, heterogeneous, and distributed nature of computer networks imposes many challenges on the CN reasoning and learning processes. This also limits the number of possible techniques that are applicable to CNs. The current section defines our view of the reasoning and learning in CNs, discuses challenges for reasoning and learning techniques for CNs, and suggests reasoning and learning techniques that are most suitable to CN.

2.3.3.1 Reasoning

The word "reasoning" has many different definitions. All are related but have different scopes and some customized elements. In this dissertation, we consider reasoning to be the process of finding solutions for a networking problem based on the observed environment. These solutions can be translated later to a set of reconfiguration actions. The problem solving can be as simple as making an immediate decision using available historical knowledge or as complex as forming a new problem representation and applying an optimization technique to find a solution that is good enough based on the desired performance and available resources. From the Artificial Intelligence (AI) point of view, the previous definition encompasses different processes including: decision making, problem solving, planning, and reasoning.

The CN reasoning algorithm must deal with many challenges including: working with un-

certain or partial knowledge, dynamic real time situations, untrusted system components, cross layer interactions, and multiple (sometimes conflicting) objectives. It also needs to be flexible and customizable to a wide variety of communication and networking problems. With all these challenges and constraints, the complexity of the CN reasoning algorithm still needs to be minimized. In [4], Friend investigates some of the reasoning and learning techniques that appear to be most applicable to CNs. Friend suggests that among the most promising candidate techniques for reasoning and learning for CNs are metaheuristics. A metaheuristic is a high-level algorithmic framework or approach that can be specialized to solve optimization problems. Among the possible metaheuristic techniques, we have chosen to use a parallel genetic algorithm technique, namely the island Genetic Algorithm (iGA). Section 3.4 presents some candidate metaheuristics for the CN distributed reasoning algorithm and justifies our choice for the metaheuristics in general and the iGA in particular as the basis for the CN distributed reasoning algorithm.

2.3.3.2 Learning

Although cognitive networks (CNs) can be designed and implemented without a learning process, the complexity of the potential CN problems together with the real time requirements of most of communication and networking problems suggests the coupling of CN reasoning with learning. Learning from past experiences reduces the time required to find good solutions for new problems and helps in making an immediate decision in situations similar to past problems. In a real time environment, two levels of reasoning may be required: one for taking immediate decisions based on past experience and the other for searching for suitable solutions for newly encountered problems. Coupling learning with reasoning is needed to learn from past experience and maintain a knowledge base to help the reasoning algorithm make appropriate decisions based on the urgency of the encountered problem.

From the suggested leaning techniques in [4], we suggest the use of distributed reinforcement learning to learn successful island genetic algorithm (iGA) configurations in a knowledge base

that can be used later to decide for the best parameter settings for the iGA. Reinforcement learning is the technique of learning system behavior through trial-and-error interactions with a dynamic environment [28]. Given the existence of observation sensors in most suggested CN architectures, reinforcement learning is one of the best fitting techniques for learning in CNs. Moreover, we suggest the use of case-based reasoning/learning (CBR) to learn and maintain problem knowledge generated from iGA experience. This knowledge can be used to enhance the iGA global search ability and accelerate the search convergence. CBR can be defined as the process of solving new problems based on the solutions of similar past problems [29]. The use of CBR can help CNs make immediate decisions using existing solutions for similar cases in the case knowledge base. Due to the difficulty of CN reasoning alone, we leave CN learning to future work.

2.3.4 Applications

Cognitive network concepts can be applied to cognitive radio (CR) and cognitive radio network (CRN) applications. Having integrated the distributed reasoning and learning across the protocol stack and throughout the network, the CNs can be applied to more complex distributed networking problems. While CR research focuses on the dynamic spectrum access problem, CN research applies the CN principles to a variety of simple and complex cross layer problems.

In [30], Thomas applies the CN principles to multicast flow lifetime and topology control problems. For the first problem, the CN utilizes three reconfigurable elements: the radio transmission power, antenna directionality and element routing tables. A single objective optimization with the end-to-end goal of maximizing the flow lifetime constrained to nodes' available energy is used by the CN. The flow lifetime maximization is accomplished through a cognitive process that changes the states of these reconfigurable elements. The cognitive process consists of three cognitive elements with local objectives, namely PowerControl, DirectionControl, and RoutingControl, each cognitive element controls one of the three re-

configurable elements. Cognitive elements repeatedly observe the environment, examine the impact of possible action choices, and ultimately make decisions that improve their local objective, and, in turn, the end-to-end objective.

The second problem Thomas addresses in [30], the topology control problem under static and dynamic network conditions, uses both power and channel control to minimize both the maximum transmission power and the number of orthogonal channels required to achieve interference-free connections. For this problem, the cognitive elements are distributed on each radio of the network. Each of the cognitive elements observes the network conditions and then chooses either a reduced power level that still maintains topological connectivity or a channel that allows interference-free connectivity with all desired receivers.

In [4], Friend applies the CN principles to two more problems, namely multichannel topology control for dynamic spectrum access (DSA) and routing in a mobile ad hoc network (MANET). For the first problem a heuristic algorithm is developed and its performance is compared against a heuristic for power-based multichannel topology control and against solutions obtained via a long-running genetic algorithm. For the second problem, Friend presents a model of 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 Markovian Decision Process (MDP).

In this dissertation, we apply the CN concepts to additional applications. The fundamental difference between our work and previous CN research work is the focus on a single distributed reasoning algorithm to be applied to many CN problems. The main focus of our research is then to demonstrate the applicability of our localized variations of the island genetic algorithm as the CN reasoning algorithm. Toward this goal, we apply the CN architecture presented in Chapter 3 to single and cross layer communication and networking problems.

2.4 Position of our Research

Cognitive Networks (CNs) research has recently received much attention because of its potential to solve various hard cross layer communication and networking problems. However, most of the previous work is still in an early stage with the main focus on radio-level problems. We classify our work to be within the cognitive network category where reasoning and learning are distributed across the protocol stack and throughout the network, and not limited to a specific layer or within node-centric goals. It is worth clarifying that we limit our focus in this dissertation to the CN research that is evolving from cognitive radio research and not to other similar research areas such as smart networks, intelligent networks, and adaptive networks.

Our work builds on top of the work presented in [20] by assuming the existence of a reconfigurable platform that allows different levels of reconfiguration ranging from changing a parameter within a layer to changing the entire stack. Moreover, our work extends the work presented in [21] to the networking level through applying distributed reasoning techniques to communication and networking problems that need cooperation among nodes and use network-wide goals instead of node-centric goals. To the best of our knowledge, this work is the first to focus on reasoning in a cognitive network. This work is also the first to suggest the use of one distributed reasoning algorithm, namely the island Genetic Algorithm (iGA) and other localized variations introduced in this dissertation, as the sole CN reasoning algorithm instead of using more complex architectures with several problem-specific optimization toolboxes and heuristic algorithms. This may sacrifice performance for some problems with well-known heuristic algorithms but also greatly simplifies the implementation of the CN.

In our effort to demonstrate the applicability of the iGA as the CN distributed reasoning algorithm, we first present our cognitive node architecture and describe the main functionalities and connections among architecture components. We then apply the iGA and more promising localized variations that we develop to single layer and cross layer communication and networking problems. More specifically, we apply the iGA to channel allocation, joint

channel allocation and power control, and flow routing. We evaluate the performance of iGA (and our developed variations) through simulations and compare the results against the optimal solution in some cases (usually small networks) where computing the optimal solution is feasible. For the flow routing problem, we compare the iGA performance to theoretical work that guarantees a percentage of the optimal solution. Beside the simulation work, we present a real CN implementation and apply it to the channel allocation problem.

The next chapter presents our cognitive node architecture and briefly describes the functionalities and connections among architecture components. Moreover, it introduces the genetic algorithms and island genetic algorithms in general and our localized island genetic algorithm variations in particular.

Chapter 3

The Proposed Cognitive Node Architecture [1]

This chapter presents our cognitive network architecture and the scope of our research for the rest of this dissertation. First, the cognitive node architecture in a cognitive network is presented in Section 3.1 followed by a brief description for the functionalities and connections of major architecture components in Section 3.2. Section 3.3 highlights the reasoning process as our research focus for the rest of this dissertation. Section 3.4 introduces some metaheuristics as candidate algorithms for implementing the cognitive network distributed reasoning process and then justifies our choice of the island genetic algorithm. Section 3.5 provides more details about the genetic algorithm and the island genetic algorithm and then introduces our developed localized variations of the island genetic algorithm that we apply to cognitive network problems in Chapters 6 and 7.

[1] This chapter is based on the work published in [31] and is a result of joint work with Dr. Daniel Friend and Yongsheng (Sam) Shi.

3.1 Cognitive Node Architecture

The Cognitive Network (CN) appears as a promising candidate to solve complexity, heterogeneity, and more challenging problems facing modern communication networks. The CN is expected to work in dynamic heterogeneous environment, make decisions based on uncertain and partial knowledge, and be resistant to malicious and selfish node behaviors. The CN is then expected to solve multiobjective cross layer problems and to improve network overall performance while minimizing complexity. In order for the CN to meet these expectations, a modular design that takes into account current requirements as well as possible future extensions is required.

In Fig. 3.1, we present our view for cognitive node architecture in CN. This architecture is inspired by the work presented in [20] . The work in [20] provides a platform that allows for reconfiguration of the whole protocol stack with different granularity levels. We assume the presence of such platform and focus on augmenting this reconfigurable platform with the necessary components for the CN to perform distributed reasoning. The architecture dedicates independent components for the major tasks of the CN including: exchanging data and knowledge among network nodes, securing the information exchange, monitoring network performance, and controlling the distributed reasoning process. The modular design together with good interfaces among interconnected components will facilitate the implementation of the CN despite its complexity.

Due to the wide range of challenges in investigating and implementing each of the components of the CN architecture, a detailed description and investigation of every component is beyond the scope of our work. The next section briefly describes the architecture major components.

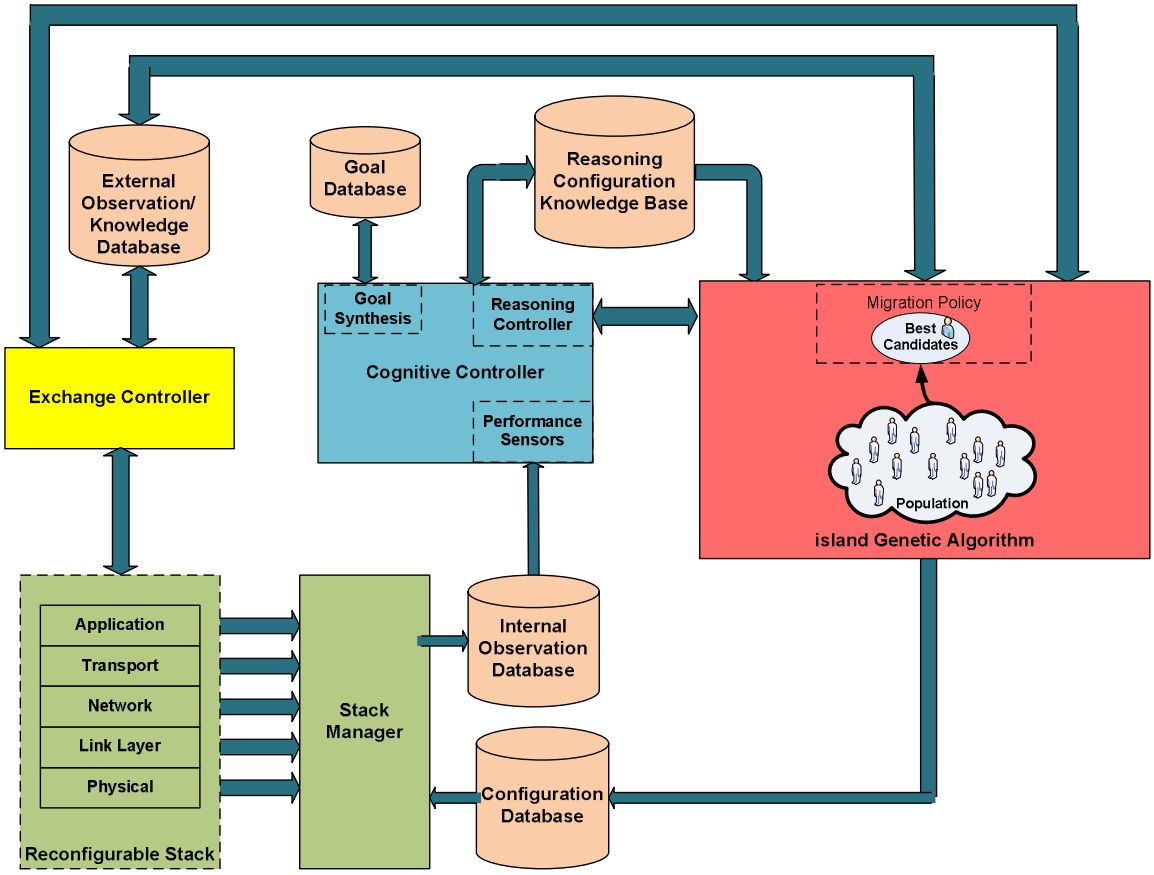

Figure 3.1: Cultural Algorithm-based Cognitive Node Architecture

3.2 Main Components Functions and Connection

The cognitive node architecture shown in Fig. 3.1 has seven major components: reconfigurable stack, stack manager, configuration and observation databases, exchange controller, cognitive controller, goal database, and distributed reasoning process. The distributed reasoning is an island Genetic Algorithm (iGA) based implementation. The next subsections briefly describe the functions and connections of these components.

3.2.1 Reconfigurable Stack

The reconfigurable stack and stack manager are inspired by the like-named components presented in [20]. Each cognitive node has a flexible platform that allows for different levels of reconfiguration, ranging from changing parameters within a component to the replacement of the entire stack. This allows the distributed reasoning process and the cognitive controller to choose from a wide variety of possibilities to adapt the system in response to network conditions.

3.2.2 Stack Manager

The stack manager constructs the stack and reconfigures protocol layers by translating new configurations into corresponding stack reconfiguration commands. This new configuration is based on the outcomes of the distributed reasoning process with the guidance of the cognitive controller. Stack construction and reconfiguration may require some degree of synchronization (internal and inter-node) to ensure smooth transitions between different configurations, especially when multiple nodes are involved in a solution. Moreover, the stack manager handles the flow of internal observations from the reconfigurable stack to the internal observation database and controls the rate at which observations are made.

3.2.3 Configuration and Observation Databases

The Configuration Database is a database that holds internal configuration information. At least two internal configurations are maintained in the configuration database: the current and pending configurations. Additional historical configurations may also be stored in the configuration database to allow for evaluating the corresponding performance and turning this information into knowledge by a learning process.

The Internal Observation Database is a database that holds the internal sensed observations. This is used by the cognitive controller to track overall performance and decide the timing and configuration of the next execution of the distributed reasoning process. The External Observation/Knowledge Database is a database that holds observations and knowledge collected from external nodes. It also holds the observations and knowledge allowed to be shared with external nodes. Keeping this information in a database for a distributed network, rather than internally in the distributed reasoning process or the cognitive controller, facilitates the access and exchange of this information by the exchange controller without interrupting other processes.

3.2.4 Exchange Controller

The Exchange Controller is responsible for securely exchanging information with external nodes based on exchange policies. Policies for exchanging information between cognitive nodes are set by the cognitive controller, based on trustworthiness and resource availability, and stored and enforced by the exchange controller. Thus, when an external node requests information from the external observation/knowledge database, the exchange controller responds based on the established policy without interrupting the cognitive controller. Moreover, requests for external information by the distributed reasoning process or the cognitive controller pass through the exchange controller so that requests are secured and may be packed together to reduce communication cost. Upon receiving the requested external infor-

mation, the exchange controller stores them in the external observation/knowledge database.

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 information, such as new configurations, is maintained.

3.2.5 Cognitive Controller and Goal Database

The Cognitive Controller is the brain of the cognitive node. The cognitive controller is responsible for five main tasks:

- *Trust and Priority Management*: The cognitive controller determines the trustiness of external nodes based on observations of node behavior. These trust levels are then used by the exchange controller to regulate information exchange. External observation/knowledge collected by the exchange controller is also tagged with the trust level of the corresponding source node to indicate the reliability of the information. Moreover, the responsiveness of the cognitive node to external nodes' requests is prioritized based on the trustworthiness of the requesting node, the type and amount of information requested, and the resource constraints of the cognitive node (for example battery power level).

- *Coordination and Negotiation*: The cognitive controller is also responsible for coordinating and negotiating with other cognitive nodes to reach agreement on network-wide reconfigurations. This task is essential especially in dynamic networks in which cognitive nodes may enter and leave the network and with conflicting goals where convergence to exactly the same solution may not be possible in a reasonable amount of

time.

- *Performance Tracking*: The performance monitors inside the cognitive controller keep track of the cognitive node performance in the network. When the performance drops below a threshold, the cognitive controller triggers a new round of the distributed reasoning process to optimize the node configurations to the current network state.
- *Goal Synthesis*: The cognitive controller is also responsible for interacting with the goal database which stores local and network-wide goals and for translating these goals to non-conflicting objective functions. The cognitive controller then communicates these objective functions to the distributed reasoning process to use in evaluating potential solutions.
- *Controlling Distributed Reasoning Process*: The cognitive controller also controls the distributed reasoning process by deciding when it is necessary to re-optimize the network configuration as well as how the distributed reasoning process should be configured.

3.2.6 Distributed Reasoning Process

Cognitive nodes need to cooperate in order to achieve network-wide goals. The distributed reasoning process is responsible for executing and regulating this cooperation. Based on the goals and configuration set by the cognitive controller, the distributed reasoning processes in all or some of cognitive network nodes cooperate to re-optimize the network configurations to achieve these goals. The cooperation degree among nodes can be controlled based on the nature of the problem and the availability of resources. The distributed reasoning processes on different nodes can also be configured with different parameter settings based on their corresponding node capabilities.

The proposed cognitive node architecture features two levels of reasoning. The two reasoning levels are performed inside the distributed reasoning process. One of them is immediate

reasoning, which responds immediately and reconfigures network settings based on the best match (the best solution of the closest similar problem) available from the historical knowledge base. This level of reasoning is triggered when the encountered problem needs an immediate action. The second reasoning level is long-term reasoning, which allows the reasoning algorithm to work for a longer period of time to find a good solution. This level is necessary for newly encountered problems and to improve previous solutions. The cognitive controller, based on environment sensing and problem nature, triggers one reasoning level or both of them to get an immediate action and/or start reasoning for a better solution that fits current network conditions. Triggering the immediate reasoning can be as simple as setting the number of iterations of the genetic algorithm to zero. In this case, the reasoning process responds with the best individual in its initial population, which is initialized using the solutions of similar cases from historical knowledge.

3.3 Our Work Scope

Cognitive Networking is a multidisciplinary research field that encompasses communications, networking, computer architecture, and artificial intelligence. Each of the architecture components requires in depth knowledge in more than one of these areas. For example, the exchange controller requires deep knowledge in networking, security, and good programming skills. Therefore, it is impossible for a Ph.D. dissertation to cover even a fraction of the CNs open problems.

The rest of our work focuses on the distributed reasoning component. We suggest the use of an island Genetic Algorithm (iGA) as the distributed reasoning algorithm. However, we have developed localized variations of the iGA that allow for a compromise between communication cost and achieved performance. Using one reasoning algorithm instead of a set of optimization and heuristic algorithms simplifies the CN implementation. This also sacrifices some of the performance, especially for problems with well-known problem-specific

algorithms. Combining some problem-specific solutions for performance critical problems with the iGA is still an option worth more investigation. The next section explores some parallel distributed metaheuristics as candidate algorithms for implementing the CN distributed reasoning process and justifies our choice for the iGA as the reasoning algorithm for CNs.

3.4 Distributed Reasoning Algorithm

In this section, we first justify our selection for the metaheuristics in general as the main category from which we choose a candidate for the CN distributed reasoning algorithm. We then present some candidate metaheuristics that appear to be applicable to CNs. Finally, we defend the choice of the iGA as the basis for the CN distributed reasoning algorithm.

3.4.1 Reasons for Choosing Metaheuristics

A metaheuristic is a high-level algorithmic framework or approach that can be specialized to solve optimization problems. Metaheuristics can also be used as a high level strategy that guides other heuristic methods in searching for feasible solutions. Metaheuristics are generally applied to problems for which there is no satisfactory problem-specific algorithm or heuristic or when it is not practical to implement such a method. The time complexity of deterministic algorithms to solve these hard problems (to find global optimum) usually grows exponentially in the dimension of the search space. As many networking problems have been shown to be hard problems that do not have polynomial time solutions, metaheuristics in general and parallel metaheuristics [32] in particular appear to be promising techniques for reasoning and learning in CNs with the ability to solve large problems in reasonable time.

Another appealing feature of metaheuristics is the flexibility and possible customization to a wide variety of problems. Most of the metaheuristics are generic optimization techniques

that apply problem-independent operations. This is essential for any CN reasoning technique because of the expected variety of problems that should be addressed by CNs. Having a problem-specific algorithm for each expected problem is infeasible because of the subsequent system complexity and the unpredictability of future networking problems and possible objective functions. Metaheuristics also allow for a controllable trade off between the quality of the final solution and the allocated computation time (or other allocated resources). This is also essential for any CN reasoning technique because of the different timing requirements of different communication and networking problems. The next subsection explores some popular metaheuristics that appear to be of interest for CNs.

3.4.2 Candidate Metaheuristics for Cognitive Networks

Genetic Algorithms One of the first developed and most explored metaheuristic is the Genetic Algorithm (GA). The GA is an optimization and search technique based on the principles of genetics and natural selection [33]. GAs are a class of evolutionary computation algorithms, a rapidly growing area of artificial intelligence, which uses techniques inspired by evolution. The GA was first introduced by Holland through his schema theorem which was the first attempt to develop a theoretical basis for the GA. However, the GA was later popularized by one of Holland's students, David Goldberg, who was able to solve a difficult problem using the GA. The GA evolves a population representing optimization problem candidate solutions through evolutionary operations (crossover and mutation) toward better solutions until a maximum number of generations has been produced or a satisfactory solution has been reached.

Three major classes of parallel GAs are described in [34]: Master-Slave GA (micro-grained), island GA (iGA) (coarse-grained), and cellular GA (fine-grained). The master-slave GA is the simplest where one node controls the communication and evolutionary operations at all other nodes. The iGA divides the population into subpopulations that evolve separately each at different node. Nodes communicate regularly to share their candidate solutions to

increase local diversity using a migration policy that defines the migration rate and topology. The cellular GA is an extreme case where each node has only one individual and interacts with its neighborhood to evolve and apply evolutionary operations. As discussed in [4], the iGA is the best candidate out of the three forms as for the CN reasoning algorithm because iGA does not require a central control node as in the master-slave GA and has less (and controllable) communication cost among nodes than the cellular GA.

Simulated Annealing Simulated Annealing (SA) is another well founded metaheuristic that simulates the annealing process in metallurgy. The SA was first described by S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi in 1983 [35]. SA is a single point stochastic search technique which starts with an arbitrary initial solution (state) and moves toward the global optimum solution. At each iteration, a neighbor is created from the current solution by changing part of the solution. If the neighbor's objective function value is better than that of the existing solution, the neighbor replaces the current solution. Otherwise, the neighbor is accepted probabilistically depending on the current annealing temperature which starts with a high value (translating to a high acceptance probability) to allow the algorithm to explore a broad region of the search space, and gradually decreases with the time to converge to the optimum solution region [36]. The algorithm stops when a satisfactory solution has been reached or a stopping condition has been met. SA can be seen as a hill climbing search that allows for down hill moves to avoid getting trapped in local optima. SA can also be thought as a GA algorithm with a population of size one that only applies mutation and probabilistically accepts the new individual with a timely decreasing probability.

The standard SA algorithm is not appealing for CNs because it can easily trapped by local optima or it may take too long to find a reasonable solution [37]. Moreover, the SA performance is usually affected by the configuration of the starting solution. Different synchronous and asynchronous parallel SA algorithms [32] [38] and hybrid SA algorithms [37] exist in the literature that increase the efficiency of the SA in terms of convergence time and final solution quality. However, it is difficult to parallelize the SA because it is a naturally sequential algorithm. Most of the parallel SA algorithms have high communication cost. Due to this

fact and the fact that it is easier to modify a general GA implementation to have the SA algorithmic effect for problems that SA would give better performance, we prefer the GA over the SA.

Tabu Search Tabu Search (TS) is a popular metaheuristic that is first introduced by F. Glover in 1986 [39]. TS is similar to SA in that it is also a single point search technique that iteratively tries to improve current candidate solution by exploring its neighborhood and selecting the best neighbor as the next candidate solution until a stopping condition has been satisfied. However, TS uses a memory structure to improve its performance and to prevent cycling over previously explored solutions. In its basic form, TS uses a short-term memory (the tabu list) to store the most recent moves (tabus) that have led to the current solution. This memory is then used to prevent cycling when moving away from local optima through non-improving moves. This list also helps the search move away from previously visited portions of the search space and thus explore new (adjacent) portions. Two other memory structures can be used to improve the TS performance: intermediate-term memory and long-term memory. The intermediate-term memory (recency memory) is used to record the number of consecutive iterations that various solution parts have been present in the current solution without modification and then use this information to direct the search to exploit promising areas (intensification). On the other hand, the long-term memory (frequency memory) is used to record the total number of iteration (since starting) that various solution parts have been present in the current solution and use this information to direct the search toward unexplored portions of the search space (diversification).

Many parallelization strategies have been applied to the original TS to improve its performance and reliability [40]. A taxonomy of parallel TS methods is presented in [41] using three dimensions: Search Control Cardinality, Search Differentiation, and Search Control and Communication. The two classes of distributed algorithms that perform asynchronous cooperative communication and thus most appealing for CNs are Collegial and Knowledge Collegial. In collegial algorithms, different nodes (TS processes) exchange promising solutions together. This class of algorithms is similar to the cellular GA considering that TS

is a single point search algorithm. In the knowledge collegial algorithms, new knowledge is also inferred based on the information exchange among nodes (TS processes). TS is a metaheuristic that combines both reasoning and learning and this may result in a better performance. However, it demands high communication cost to be applied in CNs because of its single-point search nature. TS requires the evaluation of the neighborhood of the current solution which may be challenging and costly for large scale and cross layer problems. On the other hand, evaluating (random) part of the neighborhood may lead to performance degradation.

Ant Colony Optimization Ant Colony Optimization (ACO) is a metaheuristic simulating the foraging behavior of real ants that is first proposed by Marco Dorigo in 1992 [42]. ACO is one of the swarm intelligence algorithms which are algorithms inspired by the observation of the behavior of swarms. Swarm intelligence algorithms are made up of simple individuals that cooperate through self-organization without any form of central control over the swarm members which makes them attractive for ad-hoc CNs. ACO is a probabilistic technique in which a number of artificial ants build solutions to the considered optimization problem at hand and exchange information on their quality via a communication scheme that is similar to the one adopted by real ants [43]. At each iteration of the ACO algorithm, each of the artificial ants constructs a candidate solution by adding solution components together from the set of feasible values for each component. The choice of a solution component value is guided by a stochastic mechanism, which is biased by the pheromone (representing the preference level) associated with each of the solution component values. The candidate solutions may be improved through a local search and the pheromone values of the candidate solution components are then updated to increase the pheromone values associated with good or promising solution components and decrease those that are associated with poor solution components.

ACO is a population-based algorithm, and thus any parallel model used in other population-based algorithm can be adapted to ACO. In particular, two parallel models have been identified in [43]: fine-grained and coarse-grained, with the resulting structures are similar to

cellular and island GAs and with the recommendation is given to the coarse-grained parallelization because of the communication overhead reduction compared to the fine-grained parallelization. The coarse-grained ACO shares the same good communication characteristics as the iGA. The ACO fundamental algorithm makes it more applicable for solving problems which can be reduced to finding good paths through graphs such as network routing and load balancing [44] [45]. However, this reduction may not be straightforward for all CN applications and will impose additional complexity and that is why we prefer the GA over the ACO.

Particle Swarm Optimization Another example of the swarm intelligence algorithms is the Particle Swarm Optimization(PSO). The PSO was first described by James Kennedy and Russell C. Eberhart in 1995 [46]. PSO is a population-based algorithm and shares some characteristics with the ACO, except for the information sharing mechanism among individuals (particles/artificial ants). While artificial ants share information about the preferences of solution component values through updating the associated pheromone values, particles in the PSO algorithm communicates their best solutions and use their own previous best solution and the global best solution over all particles to improve current candidate solutions.

In PSO, population individuals are modeled as particles in the multidimensional search space that have a position and a velocity. At each iteration of the PSO algorithm, each particle updates its position and velocity based on its own previous best solution and the global best solution reached so far by any particle. Each particle is initialized randomly to hopefully explore different portion of the search space and then all particles eventually converge toward global best solution. As with all previous metaheuristics, the algorithm stops when a satisfactory stopping condition is met. However, the original PSO algorithm can easily get trapped by a local optima because of a dominant particle.

Various parallelization strategies and improvements have been applied to the PSO. The same synchronous/asynchronous and micro/coarse/fine-grained parallelization strategies as described with previous techniques have been suggested and applied to the PSO [47] [48] [49].

Whilst PSO is shown more efficient than GAs for some problems, it is also more prone to premature convergence than GAs because of the possible domination of an early sub-optimal particle solution [47]. However, we feel that the PSO is still a promising candidate for the CN distributed reasoning algorithm, except that its operations are less generic compared to GAs operations (evolutionary operations).

Although most of the previous parallel metaheuristic implementations target a single machine with parallel processing power, the same concepts can be applied to the CN by mapping processors to network nodes. Several successful applications of metaheuristics to optimization problems in communications and networking are presented in [50]. General parallel strategies and implementation principles for metaheuristics are presented in [51] and [52] with the recommendation is always given to the class of cooperative asynchronous parallelization strategies. While we explored in this section some prominent metaheuristics and their existing or possible parallel implementations, other successful parallel metaheuristics exist in the literature [32]. Each with its upsides and downsides. We have chosen to use the iGA as the basis algorithm for the CN reasoning algorithm. However, we believe that other parallel metaheuristics can also be modified to have the same or similar advantages as the iGA for the CNs. We summarize the reasons for choosing the iGA over other surveyed parallel metaheuristics in the next section.

3.4.3 Reasons for Choosing Island Genetic Algorithm

It is hard to justify the use of one specific algorithm as opposed to all existing algorithms and techniques especially when targeting large number of potential current and future problems as the case in cognitive networks (CNs). Thus, instead of comparing the island genetic algorithm (iGA) against other existing algorithms side by side, we justify our choice of the iGA as the most promising metaheuristic, among metaheuristics described in the previous section, for implementing the CN distributed reasoning algorithm. As discussed earlier in Section 2.4, we focus on a single distributed reasoning algorithm to target all CN problems,

instead of using a toolbox of many problem-specific techniques, to reduce the complexity of the CN and make its implementation easier.

The reasons behind our choice of the GAs in general and iGAs in particular are:

- GAs have been successfully applied for searching complex multidimensional domains which are typical in communication and networking problems. The GA evolution does not require the evaluation of the neighborhood of the current solution as with the TS which yields to less computational cost for large and multidimensional search spaces.
- GAs perfrom parallel search from multiple points in the space which helps exploring more areas of the search space compared to other single point metaheuristics such as SA and TS. Consequently, the initial solution has less impact on the final solution for the GA than for single point metaheuristics.
- As a population-based algorithm, it is naturally easier to parallelize and distribute the GA over CN nodes. The asynchronous cooperative parallelization is one possible strategy to implement the iGA which is recommended as the most general and promising communication-efficient metaheuristic parallelization strategies.
- GAs provide a list of candidate solutions, not just a single solution, which can be useful for the nodes in CNs to have some alternatives without the need to re-run the optimization algorithm.
- Since we are targeting a variety of known and unknown future problems with our CN architecture, we need a flexible technique that can be customized and applied to different problems. GA can handle different types of optimization variables and can be applied to single as well as multi-objective functions. Moreover, GA can accommodate more problems by tuning its parameters to avoid premature convergence and balance the relation between the exploitation and exploration of the search space. As opposed to other metaheuristics described in the previous section, the GA operations and the

parallelization of the iGA provide the most generic, problem-independent option for the CN distributed reasoning algorithm.

- The main reason behind our choice of the iGA is because of the flexibility in defining the migration policy which we can use to balance the performance to the available resources. This also implies reduced communications between network nodes compared to other parallel metaheuristics.

- The GA is less prone to premature convergence compared to the ACO and PSO. The iGA islands' separation concept can further prevent premature convergence by allowing each island to search in different areas of the search space, preventing a single highly fit individual from dominating the entire population.

- With simple modifications to the standard iGA, we have developed localized variations that solve the scalability problem of the iGA which is a major problem in applying iGA to large scale problems as in CNs. These localized variations are then better fit for the CN reasoning algorithm with their reduced communication and truly distributed nature.

- Finally, the No Free Lunch theorem [53] suggests that over all problems, all optimization algorithms will have the same average performance. The option of having multiple optimization algorithms in the node architecture is more complex to implement and needs an additional controller to decide how to best match algorithms to problems.

Of course, the GA is not the best way to solve every problem. However, it is a good fit for our needs for the CN reasoning algorithm considering our decision to focus on only one flexible algorithm for all CN potential problems. The rest of this dissertation attempts to evaluate our selection for the iGA as the distributed CN reasoning algorithm by applying the iGA and our developed localized variations to some of the potential CN problems and studying and analyzing their performance. The next section provides more details about the GA, standard iGA, and our developed localized variations of the iGA.

3.5 Details of Genetic Algorithms

In this section, we first introduce the Genetic Algorithm (GA) fundamentals and main procedure in more details. We then present the island Genetic Algorithm (iGA) and discuss its main differences from the standard GA. Next, our developed localized variations of the iGA are briefly described.

3.5.1 Genetic Algorithm

Fig. 3.2 shows the main loop of the GA. The general idea is to allow a population composed of many individuals to evolve under specified selection rules to a state that maximizes the "fitness" (i.e., minimizes the cost function) [33]. In the following, each of the main loop stages is described:

- *Problem Representation and Fitness Function*: The first stage for the GA starts by defining a genetic representation of the solution domain. A standard representation of the solution uses an array of bits (of zeros and ones). This GA representation is called a Binary Genetic Algorithm and it facilitates the operation of the GA evolutionary operations. The other option is to use a more complex representation which uses any other data type or structure. Most of the time, this representation fits the real world problems better than the binary representation. Moreover, this representation does not require complicated encoding functions to map problem variables to binary values. This comes with more complexity in applying the GA evolutionary operators. The solution domain representation and choice of variable encoding is one of the most important stages. A good representation can far outperform a poor one for exactly the same problem and the same GA evolutionary operations. The next step of the GA is to define the optimization variables and the objective function. The objective function (also called fitness function for maximization problems or cost function for minimization problems) is defined over the genetic representation and measures the

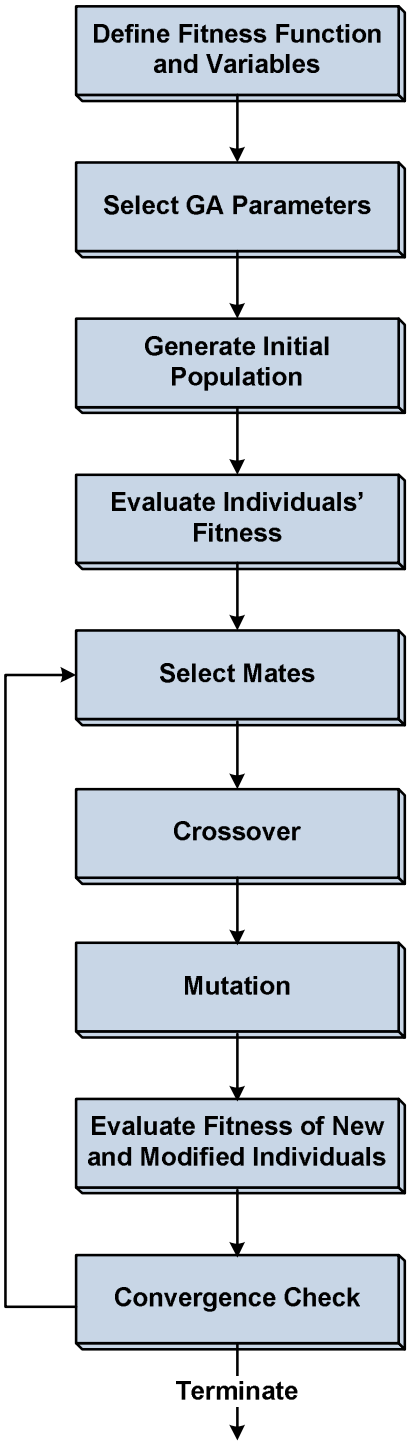

Figure 3.2: The Genetic Algorithm Main Loop

quality of the represented solution. The objective function is always problem dependent and needs to be as simple as possible because the evaluation of the objective function is the most time consuming stage of the GA procedure. Once we have represented the solution domain and defined the objective function and optimization variables, the GA parameter settings need to be decided.

- *GA Parameter Settings*: Perhaps this is the most difficult decision through the chain of decisions required for the GA procedure. Population size, selection rate, mutation rate, and number of iterations are all parameters to set. While some suggestions for the good range of each parameter can be found in the literature, the absolute best parameter settings are impossible to determine because they are problem dependent. This stage always includes some trial and error simulations to decide on acceptable parameter settings. Recently, the parameter-less genetic algorithm is introduced as a way to avoid trial and error experiments by incorporating knowledge of parameter selection and population sizing theory in the genetic algorithm itself [54].

- *Population Initialization*: Unless one has a knowledge base of historical solutions or domain information to help in generating better fitness individuals, random generation of the initial population is usually performed. The randomly generated individuals (solution candidates) are hopefully spread over the entire search space. Occasionally, individuals may be directed through GA evolutionary operations to areas where optimal solutions are likely to be found. It is important to note that finding a balance between exploitation and exploration of the search space is essential to avoid premature convergence to suboptimal solutions.

- *Selection*: Part of the population has to be discarded to vacate space for newly generated individuals (offspring). The generation of new individuals involves the selection of which of the current individuals are selected as parents to breed new individuals. Different selection methods can be used including: Pairing from top to bottom, random pairing, roulette wheel selection, and tournament selection. Roulette wheel and

tournament selection are standard for most GAs [33]. Both methods are probabilistic which helps keep some diversity in the population to prevent premature convergence. In roulette wheel selection, probabilities are assigned to population individuals based on their fitness/cost so that better individuals have better selection chance as parents. A random number then determines which individual is selected. In tournament selection, a small subset of individuals (two or three) from the mating pool is randomly picked, and the best individual in this subset becomes a parent. The tournament is then repeated for every needed parent to generate next generation.

- *Crossover*: Having selected the parents from the pool of current individuals, mating is performed among these parents to generate one or more offspring (usually same number as the number of discarded individuals to keep a fixed population size). The most common form of mating is the crossover that uses two parents to produce two offspring. One or more crossover points are randomly selected to split the parents' genes into pieces and pieces of different parents are then combined to form the new offspring.

- *Mutation*: Crossover may lead to a next generation that explores more areas in the search space. However, it is still tied to the previous generation because it does not introduce any new gene values. On the other hand, mutation can be used to explore search space areas that cannot be explored by crossover. Random mutations are used to alter a certain percentage (depends on mutation rate) of the population individual genes. Mutation is a one way to prevent the GA from converging to a local optimum before searching the entire search space.

- *Convergence Check*: The evolution of individuals through mating and mutation is repeated until a termination condition has been reached. This can be as easy as running the GA for fixed number of iterations. The termination condition can also be performance-based. For example, the GA terminates when an acceptable solution is reached or when the GA individuals converge to the same solution.

The GA has been applied successfully for searching complex domains and solving hard problems with no known good-performance heuristic algorithms. Having introduced the GA fundamentals and operation in this section, the next section presents a parallel version of the GA, namely the island genetic algorithm. The GA can be used as the CN reasoning algorithm but the lack of distributed structure makes it difficult to use the GA for networking problems that are mostly distributed by nature.

3.5.2 Island Genetic Algorithm

The island Genetic Algorithm (iGA) is a class of parallel genetic algorithms originally developed to take the advantage of parallel processing capabilities of modern computers. While the standard GA works on a single processor, parallel GAs distribute the computation over the number of available processors to speed up the GA operations. The same concept can be applied to the CN by mapping processors to network nodes. Fig. 3.3 illustrates the basic concept of the iGA. The iGA can be seen as a computationally distributed version of the genetic algorithm that divides the population into subpopulations, or islands. Each evolves apart from other subpopulations. This separation into subpopulations can prevent premature convergence by allowing each island to search in different areas of the search space, preventing a single highly fit individual from dominating the entire population [33]. If there is never any communication between nodes, then this island model is equivalent to performing many runs of the GA on several small populations at once. However, islands are allowed to interact periodically through the migration of individuals to other islands. This migration is performed according to a migration policy, which defines where and when individuals move [34]. The migration timing can be either synchronous or asynchronous. In synchronous iGA, all islands exchange members at the same time; in asynchronous iGA, each island independently decides when to migrate its member(s). A simple migration policy consists of a migration interval, which is the number of generations that occur between migrations, and a migration topology, which determines where individuals migrate.

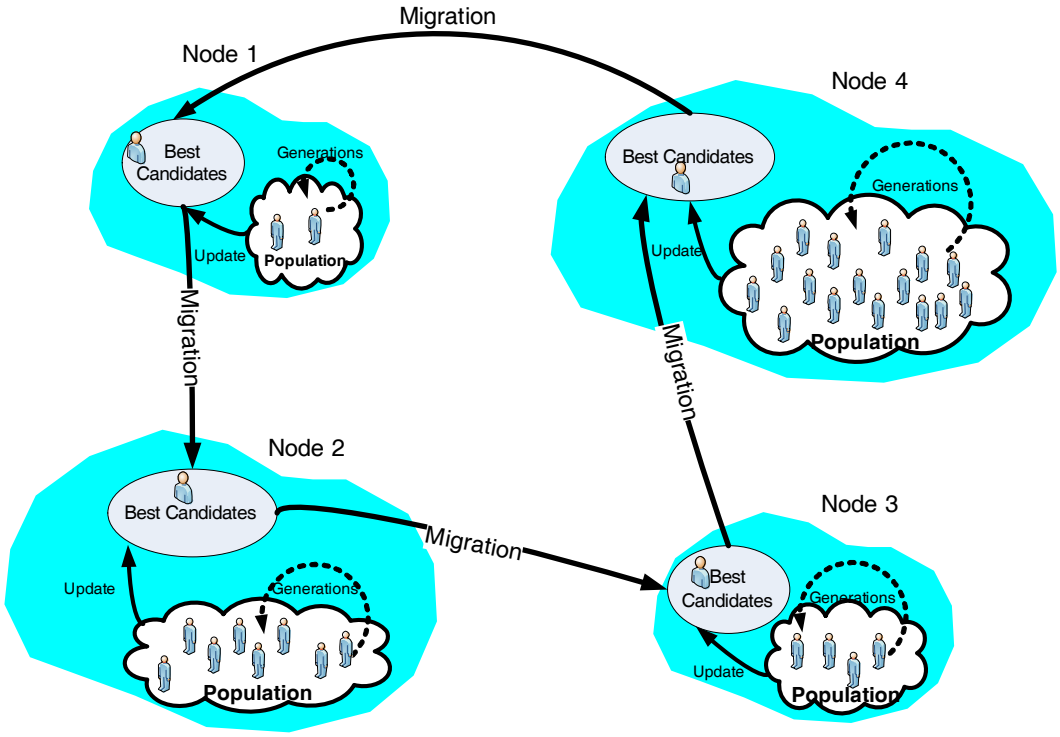

Figure 3.3: Island Genetic Algorithm

Algorithm 1 presents the pseudo-code for the standard iGA to run at each network node. P_t represents the node subpopulation at time t. M is the number of population individuals at each node. Φ is the iGA objective function used to evaluate individuals fitness. μ is the percentage of the current subpopulation that will remain in the next generation (*keep rate*) while ρ is the mutation rate.

Most of the algorithm steps are similar to the main loop of the GA presented in Fig. 3.2 except that they are applied on each subpopulation separately. The iGA also integrates the main loop of the GA with a migration policy that regulates the information exchange among nodes during the iGA iterations and defines how the final solution is propagated to all network nodes. As with the GA, the iGA algorithm at each network node starts by initializing (usually randomly) the first population, P_0. The individuals of the initial population are then evaluated using the objective function Φ. The algorithm then iterates the main evolution and communication loop. At each iteration, a number of current population

Algorithm 1 Island Genetic Algorithm Pseudo Code

$t = 0$
$initialize(P_t)$
$evaluate(P_t, \Phi)$
repeat
  $discard(P_t, \lfloor (1-\mu) \cdot M \rfloor)$
  $P_{t+1} \leftarrow P_t$
  for $i = 1$ to $\lfloor (1-\mu) \cdot M \rfloor$ **do**
    $(parent1, parent2) = select(P_t)$
    $child = crossover(parent1, parent2)$
    $P_{t+1} \leftarrow child$
  end for
  for $j = 2$ to M **do**
    $mutate(P_{t+1}(i), \rho)$
  end for
  $evaluate((P_{t+1}, \Phi))$
  $t = t + 1$
  if $migrate(t)$ **then**
    $broadcast(p^*)$
  end if
  if $immigrant = receive()$ **then**
    $merge(immigrant, P_t)$
    $broadcast(immigrant)$
  end if
until (stopping condition is achieved)
$final = propagate(p^*)$
$apply(final)$

individuals corresponding to the percentage μ is reserved from the current generation to the next generation and the rest is discarded to free some space for new offspring. The new population is the parent population. Selection and crossover are then applied to the parent population to generate new offspring. After that, the mutation is applied to each individual, excluding the best one (highest fitness/lowest cost), with probability ρ. The new individuals are then evaluated using the objective function Φ. It worth noting that some techniques can be used to avoid the sorting of individuals at each iteration [33]. If it is the time for migration, the node shares the best individual of the current generation, p^*, with other network nodes along the migration topology. A received immigrant during the current iteration is merged into the current population and the worst individual is then discarded to reserve the same population size. The node may also rebroadcast the immigrant information based on the used migration topology. When the termination condition is met, all nodes share their final solution along the migration topology, with each node replacing the solution it receives with its own solution if its own has higher fitness and then rebroadcast it. All nodes will then apply the best final solution to solve the problem. A formal algorithmic description of the iGA can be found in [34].

The iGA, in general, uses global information. Each node uses the same length and structure for its population individuals. Thus, the iGA distributes only the computation over all nodes, hoping to find a good solution faster by allowing every node search the whole solution space simultaneously, and exchanging individuals regularly according to a migration policy. To better suit the CN environment, we have modified the iGA in order to use only local information. This solves the scalability problem of the iGA by decomposing the problem into parts (not necessarily disjoint) of roughly the same size and distributing the responsibility of searching each part to a particular network node. We then combine the partial solutions to form one global solution. The next section introduces two localized island genetic algorithm variations we have developed for use with the CN. More details about our localized iGAs are given in Chapter 6 and Chapter 7.

3.5.3 Localized Island Genetic Algorithm

Although the communication cost associated with the standard iGA is less than other parallel GA techniques, it may be a problem when applied to distributed networks because it requires global information, i.e. all network nodes use the same individual structure. Our Localized island Genetic Algorithm (LiGA), in contrast, uses individuals with different length and/or structure at different network nodes. Each node is now concerned about solving part of the problem not the entire network problem. In LiGA, each node tries to find the best solution for nodes in its interference range by searching part of the search space. Disjoint parts of the partial solutions are then combined to form one global solution. The other major modification is the change of the migration policy. Instead of exchanging the entire structure of the best individual (or another member of the subpopulation), each node shares only the solution assignment part for its own parameters from its best individual and broadcasts this information to all nodes in its interference range instead of all network nodes as in the standard iGA. Each node that receives migrated information from any other node will merge this information with all of its individuals if the merged information will give better fitness, otherwise it will merge it with only part of its population. This forces each node to consider solutions of other nodes even if they give worse localized fitness than individuals in the current generation.

In Chapter 6, we apply the LiGA to the joint channel allocation and power control problem and compare its performance to the standard iGA. The LiGA shows competitive results to the standard iGA while working in a fraction of time. Moreover, the LiGA is scalable as the network size grows. More details about the LiGA implementation and its differences from the standard iGA are given in Chapter 6. However, it is hard to believe that every possible communication and networking problem can be solved using only local information despite the possible information dissemination effect of the individuals' migration. Some networking problems by nature require larger scope of information to be solved. Using the LiGA, as described here, with these problems may require a long time to converge or may lead to poor

solutions. This is the reason that we later suggest the use of a generalized version of the LiGA called K-hop iGA.

3.5.4 K-hop Island Genetic Algorithm

The Localized island Genetic Algorithm (LiGA) is a promising technique to solve the standard iGA scalability problem when applied to large networks. However, not every communication and networking problem can be solved with local information. Thus, we have developed the K-hop iGA which is a generalized version of the LiGA that suits a big class of communication and networking problems with its controllable cooperation and migration range.

The main difference between the LiGA and K-hop iGA is the flexibility of the K-hop iGA for each node to set its cooperation and migration range based on the nature of the problem under concern and local resource availability. With K-hop iGA, network nodes can use different localized range for different problems. This ranges from zero-hop to N-hop, where N is the number of network nodes (the maximum value to consider is the maximum number of hops in the network). As the value of K increases, the communication cost needed for the K-hop iGA to operate also increases but each node collects a better image of the entire network state which helps in reaching to better final solutions. If the results of 1-hop iGA for a specific problem are not acceptable, one possible way for the network nodes to improve the results is to increase the cooperation range to 2-hops instead of 1-hop. We apply the K-hop iGA to the flow routing problem and the details are provided in Chapter 7.

3.6 Summary and Contributions

Cognitive networks (CNs) are expected to solve a wide variety of hard problems resulting from the increasing complexity and heterogeneity of modern networks. Thus, CN nodes have

to be designed and implemented with the necessary components to perceive and adapt to different network conditions. CNs integrate distributed reasoning and learning across the protocol stack and throughout the network to meet requirements. The expectations of the CN impose challenges on the CN architecture design.

In this chapter, we have presented our cognitive node architecture considering current and potential future CN challenges. The architecture includes components for performing information exchange, priority management, distributed reasoning, and performance monitoring. Together with a good interface design, our architecture facilitates the implementation of cognitive networks. The functions and connections of major architecture components are then described and the scope of the rest of our work is presented.

We have then suggested and justified the choice of the island genetic algorithm (iGA) as the basic algorithm for the CN distributed reasoning algorithm. Our developed localized variations of the iGA were then briefly described with more details about how we apply them to communication and network problems to be described in Chapters 6 and 7.

In the next chapter, the channel allocation problem is modeled for the dynamic spectrum access environment and the standard iGA is applied. The performance of the iGA is evaluated through simulation.

The work presented in this chapter has resulted in the following publications:

1. Daniel H. Friend, Mustafa Y. ElNainay, Yongsheng Shi, and Allen B. MacKenzie, "Architecture and performance of an island genetic algorithm-based cognitive network," in *Proc. IEEE Consumer Communications and Networking Conference (CCNC'08)*, (Las Vegas, NV), pp. 993–997, 10–12 Jan. 2008.

2. Mustafa Y. ElNainay, Daniel H. Friend, and Allen B. MacKenzie, "Reasoning in Cultural Algorithm-based Cognitive Network," *IEEE Transactions on Wireless Communications*, under review.

Part II

Island Genetic Algorithm to Single Layer Problem

Chapter 4

Channel Allocation for Dynamic Spectrum Cognitive Network [1]

Dynamic spectrum access (DSA) has been the focal application for cognitive radios and cognitive networks. Spectrum sharing is one of the major challenges for the DSA application. Spectrum sharing regulates the way cognitive radios share the available spectrum holes with other secondary users. In this chapter, we apply the island genetic algorithm (iGA) in the context of a dynamic spectrum cognitive network to solve the channel allocation problem. Section 4.1 introduces the dynamic spectrum access channel allocation problem and our system assumptions. Section 4.2 surveys related work on the channel allocation problem. The DSA channel allocation problem is then modeled in Section 4.3. The details of how the iGA is applied to the formulated problem are given in Section 4.4. The performance of the iGA is then evaluated through simulations in Section 4.5. Finally, the chapter is concluded in Section 4.6.

[1] This chapter is partially based on the work published in [31] and is a result of joint work with Daniel Friend and Yongsheng (Sam) Shi.

4.1 Dynamic Spectrum Channel Allocation Problem

Dynamic spectrum access (DSA), as previously discussed in Section 2.2.4.1, requires augmenting radios with new functions to add the capability to sense spectrum holes, match spectrum bands to user needs, and share spectrum holes with other radios. Spectrum sharing is one of the major challenges in enabling DSA. In this chapter, we describe the channel allocation problem in a form that is unique to the DSA cognitive network context, apply the standard island genetic algorithm (iGA) that uses global information to the problem, and provide simulation results. We have chosen the DSA channel allocation problem as our first problem as it is challenging and of primary interest.

Within the cognitive network context, we focus on allocating available channels to communication links independently of specific data communication sessions. Channel availability in DSA is location and time dependent, i.e. the list of available channels at each node may be different because of different radio capabilities and/or different primary usage patterns. We focus only on the frequency domain and assume the existence of a time sharing mechanism - for example, carrier sense multiple access with collision avoidance (CSMA/CA). This time sharing mechanism is used in case of assignment conflicts to allow conflicting links to share the assigned channel. Thus, our solution tries to use the available channels in the best possible way but does not guarantee a conflict-free assignment. Finally, we work on a static network scenario and assume that the algorithm will be re-run in case of performance degradation. The next section surveys related work on the DSA channel allocation problem in various contexts.

4.2 Related Work

Channel allocation has been widely studied, particularly for cellular networks [55]. In the cellular case, the goal of channel allocation is to maximize frequency reuse subject to

interference constraints. Channels are allocated to base stations which in turn allocate channels to active users within their coverage area. More recently, channel allocation has been applied to multi-hop wireless networks, such as 802.11 ad hoc networks [56]. However, neither of these cases accurately reflects the constraints experienced in dynamic spectrum access (DSA). In cellular and 802.11 channel allocation, it is assumed that a channel can be used in any cell, as long as interference constraints are met. This is not the case in DSA because primary user spectrum occupation drives channel availability. Moreover, base stations assign channels to nodes in cellular channel allocation, while 802.11 ad hoc networks have a fixed number channels that are available at all nodes. Again, neither of these applies to DSA because different nodes may detect different sets of available channels because of different front end frequency ranges and capabilities. Different spectrum sensing algorithms at different nodes may also lead to different sensing results. Thus, DSA channel allocation requires a unique formulation and solution.

Some more recent research efforts on the DSA channel allocation problem have been published, including a survey in [57]. In [58], Wang and Liu have formulated DSA channel allocation for cognitive radio networks as a list-coloring problem with the objective of maximizing the total spectrum utilization and developed several distributed algorithms to solve the formulated problem. A similar formulation for the spectrum access problem is presented by Zheng and Peng in [59] as a graph coloring model taking into account heterogeneity in spectrum availability and interference constraints. The authors propose a centralized algorithm that extends graph coloring heuristic solutions to solve the DSA channel allocation problem. In [60], Li and Li have suggested a dynamic sharing algorithm that takes into account both the time domain and the frequency domain allocation. In [61], Barrett *et al.* have proposed sequential and distributed algorithms to solve the strong edge coloring problem which is equivalent to computing a conflict-free assignment of channels or frequencies to pairwise links between transceivers in the network.. A game theory approach is adopted in [62] and [63] to solve the DSA channel allocation problem. In [62], Halldórsson *et al.* model the DSA channel allocation as a game between spectrum providers and identify Nash

equilibria with the solutions to maximal coloring problem. In [63], Nie and Comaniciu have formulated the DSA channel allocation problem as a potential game that converges to a deterministic Nash equilibrium point. Using a conflict graph based interference estimation and minimization technique, Plummer *et al.* propose a distributed channel assignment protocol that uses only local information to solve the DSA channel allocation problem in the context of multi-radio mesh network. However, these formulations still use cellular models in which channels are assigned to base station nodes. For an ad hoc multi-hop cognitive network, a unique formulation of the problem is still needed.

On the other hand, the genetic algorithm (GA) has been successfully applied to channel allocation problems in the past [64]- [65]. Chakraborty and Chakraborty [64] use a centralized GA to compute a fixed channel allocation. Matsui *et al.* [66] apply a distributed GA to a fixed channel allocation problem. Fu *et al.* [65] combine a greedy algorithm with a centralized GA to perform dynamic channel allocation. These applications are also for cellular networks. The next section presents our formulation of the DSA channel allocation problem for an ad hoc multi-hop cognitive network. Later, we apply the island genetic algorithm (iGA) to the formulated problem and evaluate its performance.

4.3 System Model

Consider a cognitive network consisting of a set of N nodes, $i \in \mathcal{N}$, and a set of L directed communication links, $l_{i,j} \in \mathcal{L}_C$, where the subscript i, j indicates a link for which i is the transmitting node and j is the receiving node. Together, these sets define the communication graph $\mathcal{G}_C = (\mathcal{N}, \mathcal{L}_C)$. Defining communication links as directional allows $l_{i,j}$ and $l_{j,i}$ to be assigned different channels so that communication between i and j may be full duplex. In addition to the communication graph, we define an interference graph, $\mathcal{G}_I = (\mathcal{N}, \mathcal{L}_I)$, by augmenting $\mathcal{G}_C$ with a set of directed links that indicate the presence of interference between nodes that do not share a communication link. We assume that nodes transmit

omnidirectionally and with the same power on all channels so that for each link $l_{i,j} \in \mathcal{L}_I$, a transmission on any of i's outgoing links will interfere with any of j's incoming links if the links use the same channel. Therefore, any pair of nodes that shares a communication link can interfere with each other's links, resulting in $\mathcal{L}_C \subseteq \mathcal{L}_I$.

Under this model, interference is a binary condition; either a pair of links interfere, or they do not. In this sense, our interference model is similar to the protocol interference model (also called disc model) [67]; however, our model does allow interference to be determined based on signal-to-noise ratio, as in the physical interference model, or even based on random fading or shadowing. The difference between our model and the physical model of [67] is that we do not account for additivity of interference.

We assume that cognitive nodes are capable of operating on multiple channels simultaneously, both transmit and receive, and that this multi-channel capability extends across the entire spectrum being sensed. We allow nodes to perform full duplex multi-channel communication because this is the most general case. This assumption is justifiable when the cognitive nodes are sensing on the order of tens of MHz. It is less easily justified when sensing several hundred MHz, although multi-band OFDM ultra wideband technology may make such a scenario feasible as well [68].

The sensed spectrum is divided into a set of disjoint channels, $\mathcal{C}$. Each node senses the spectrum independently or cooperatively with other nodes in the same area to determine the set of channels available for local use, $\mathcal{C}_i$ ($i \in \mathcal{N}$). Based on these sets, each link has a set of available channels, $\mathcal{H}_{i,j} = \mathcal{C}_i \cap \mathcal{C}_j$ .We then define the length-L channel assignment vector $\mathbf{h} \in \times_{l_{i,j} \in \mathcal{L}_C} \mathcal{H}_{i,j}$ (the Cartesian product of the available channel sets), which is the assignment of channels to communication links. Denote $h_{i,j}$ the element of $\mathbf{h}$ that is the channel assignment for link $l_{i,j}$.

Based on the interference graph, the channel assignment problem can be modeled as:

$$\max_{\mathbf{h}} \left[f(\mathbf{h}) = \sum_{l_{i,j} \in \mathcal{L}_C} \left(\frac{w(h_{i,j})}{1 + \left| \mathcal{L}_{i,j}^{\mathbf{h}} \right|} \right) \right], \tag{4.1}$$

where $w(h_{i,j})$ is the (fixed) bandwidth of channel $h_{i,j}$ and $\left| \mathcal{L}_{i,j}^{\mathbf{h}} \right|$ is the cardinality of the set of links that cannot be active at the same time as $l_{i,j}$ under channel assignment $\mathbf{h}$. $\mathcal{L}_{i,j}^{\mathbf{h}}$ is determined from the interference and communication graphs by:

$$\begin{aligned} \mathcal{L}_{i,j}^{\mathbf{h}} \quad = \quad & \{ l_{i,y} \in \mathcal{L}_C : h_{i,j} = h_{i,y}, y \neq j \} \cup \\ & \{ l_{y,i} \in \mathcal{L}_C : h_{i,j} = h_{y,i} \} \cup \\ & \{ l_{y,j} \in \mathcal{L}_C : h_{i,j} = h_{y,j}, y \neq i \} \cup \\ & \{ l_{j,y} \in \mathcal{L}_C : h_{i,j} = h_{j,y} \} \cup \\ & \{ l_{y,z} \in \mathcal{L}_C : \exists l_{y,j} \in \mathcal{L}_I, h_{i,j} = h_{y,z} \}, \end{aligned}$$

where the indices i and j are fixed, i $\neq$ j, and the indices y and z are variables. Notice that this set includes all links that have common source or destination with the link $l_{i,j}$ and are assigned to the same channel assigned to the link $l_{i,j}$. This is the same as building a conflict graph, $\mathcal{F}$, for the network communication links, $l_{i,j} \in \mathcal{L}_C$, and then counting the number of conflicting links that are assigned to the same channel assigned to the link $l_{i,j}$.

In general terms, the goal of (4.1) is to maximize total link capacity (in terms of bandwidth) while minimizing number of channel assignment conflicts. The denominator of (4.1) reflects the fact that links which interfere cannot be active simultaneously and, hence, have to time-share the channel to which they are assigned, decreasing the overall throughput. The numerator of (4.1) allows the channels to have non-uniform capacities, though channel capacities are not allowed to be link-dependent. Under ideal conditions, in which the optimal channel assignment, $\mathbf{h}^* = \arg\max_{\mathbf{h}} f(\mathbf{h})$, results in no link conflicts, $f(\mathbf{h}^*)$ is the maximum

sum-capacity that can be achieved. This occurs because $\left|\mathcal{L}_{i,j}^{\mathbf{h}^*}\right| = 0 \; \forall l_{i,j}$, so that (4.1) reduces to $max_{\mathbf{h}} \sum w(h_{i,j})$ over the set of interference-free channel assignments. It is worth noting that we only allow one channel to be assigned to each link.

It is well known that channel assignment problems are generally difficult, with some particular formulations having been proven to be NP-hard (e.g. [56]). Due to the facts that the sets $\mathcal{H}_{i,j}$ are not known a priori and that the number of possible combinations for $\mathbf{h}$ grows exponentially in L, we believe that the DSA channel allocation problem is not easily solved, though we make no claim as to membership in NP.

While we may develop a heuristic algorithm for solving (4.1), this algorithm will only apply to this specific problem. We are interested in solving (4.1) with a method that is suitable for a wide range of problems so that the cognitive nodes are able to tackle variety of communication and networking problems as discussed in Chapter 3. Therefore, we use the island genetic algorithm. The details of applying the island genetic algorithm to the formulated problem are presented in the next section.

4.4 Applying Island Genetic Algorithm to Channel Allocation Problem

In this section, we present the details of how we implement and apply the standard island genetic algorithm (iGA) to the channel allocation problem developed in the previous section. The migration policy used for the islands to exchange best individuals together is described. An enhancement to the standard iGA procedure to avoid premature convergence is also described.

4.4.1 Island Genetic Algorithm Formulation

As described in Section 3.5.1, the first step in applying the island genetic algorithm (iGA) to the dynamic spectrum access (DSA) channel allocation problem is to define the structure of individuals and the fitness function that is used to evaluate the fitness of individuals. In our case, this is simple, as $\mathbf{h}$ is our individual and $f(\mathbf{h})$ as defined in (4.1) is our fitness function. Each individual is then a vector of channel assignment candidate solution with integers, representing channel numbers, are used to represent chromosome genes.

The next step is to define the random crossover and mutation functions. Given two valid channel assignments, $\mathbf{h}_1$ and $\mathbf{h}_2$, we perform crossover by selecting a uniform random value from the integer set $\{2, ..., L-1\}$ and using this as the crossover point for standard 1-point crossover. We use two parents to produce two offspring. The parents used in crossover are chosen by tournament selection. In our tournament selection, we choose two sets of three individuals at random from the parent population and then perform crossover on the two individuals with the highest fitness from each set of 3. Mutation is performed by selection of a uniform random value from the integer set $\{1, ..., L\}$. Supposing that the selected value corresponds to link $l_{i,j}$, $h_{i,j}$ is then replaced with a random selection from the set $\mathcal{H}_{i,j} \setminus \{h_{i,j}\}$. We exclude the best individual from mutation to ensure having it unaltered in the next generation (called elitism).

The next step is to initialize island (node) populations. Populations are initialized randomly, with each $h_{i,j}$ in each $\mathbf{h}$ selected uniformly at random from the set $\mathcal{H}_{i,j}$. The process for generating the next population from the current population for each island/network node is as follows:

- Find the fitness for each of the M individuals in the population.
- Eliminate the worst (i.e. lowest fitness in our case) $\lfloor M/2 \rfloor$ individuals from the population. This new population is called the *parent population*.
- Generate $M - \lfloor M/2 \rfloor$ new individuals from the parent population by tournament se-

lection and crossover.

- Perform mutation on each individual, except the one with highest fitness, with probability ρ.

4.4.2 Migration Policy

In addition to specifying how each population is generated, we must determine a migration policy for the iGA that nodes use to communicate and exchange information together. The migration policy consists of three parts: what, when, and where. The first part defines what information each node needs to share with other nodes. The second part defines when migration will take place, while the third part defines where to send this information. Our migration policy consists of a fixed number of iterations between the sharing of individuals (denoted by t), a migration topology that loops through all nodes (possibly generated using routing information), and the determination that nodes will share the individual with highest fitness.

When iGA process terminates (after fixed number of iterations has been reached in our case), nodes share their final solution along the migration topology, with each node replacing the solution it receives with its own solution if its own has higher fitness. All nodes will then apply this channel assignment to send or forward data to any neighboring node.

4.4.3 Population Re-seeding

If we rely solely on the above procedure without any modifications, the iGA is prone to become trapped in local maxima. Therefore, we have implemented an adaptive iGA by keeping track of the number of iterations for which the maximum fitness has not changed and re-initializing the population if the maximum fitness has not changed for a certain number of iterations. The initial limit on the number of iterations before re-initializing the population, τ, is doubled after every re-initialization. The doubling of τ value is to help

the algorithm converge when the current best individual is really the optimal solution. This approach is similar to re-seeding mechanism used in [69] to maintain population diversity and avoid premature convergence.

The next section provides details of our performance evaluation procedure for the iGA as applied to the channel allocation problem.

4.5 Simulation Settings and Results

Table 4.1 summarizes the simulation settings for network and iGA parameters that are used to run our simulations. In our simulations, the network is generated by uniform random placement of a fixed number of nodes, N, on a square area. The area of the square is adjusted to maintain constant density for all values of N simulated, with the network being 1700 meters on a side for $N = 100$. Nodes have a communication range of 250 meters and an interference range of 500 meters. These settings result in an average number of links per node of about 3.5. The sensed spectrum contains 20 channels, with each channel's bandwidth being 1 unit (the actual value does not affect (4.1) if all channels have the same bandwidth). We randomly selected a set of channels, $\mathcal{C}_i$, for each node from the 20 possible, placing an upper limit of 8 channels and a lower limit of 2 channels on the size of each $\mathcal{C}_i$ to reflect primary user occupation of much of the spectrum.

For the iGA parameters, we set the island population size to 20, $\tau = 10$, a migration interval of 20, and a circular loop migration topology. The mutation rate, ρ, was set to 0.5. Because the iGA is a random search, the results from running the iGA multiple times on the same network may differ. Therefore, results must be viewed statistically. The next sections provide numerical and graphical results of our simulations together with some discussion of the impact of these results.

Table 4.1: Network and Island Genetic Algorithm Parameters' Setting

Network and Nodes Parameters' Setting	
Area	Square w/ constant node density
Number of Nodes (N)	5-100
Node density	34.7 $nodes/km^2$
Max Communication Range ($R^{T_{max}}$)	250 $meters$
Max Interference Range ($R^{I_{max}}$)	500 $meters$
Available Channels	20
Channels Lower Limit	2
Channels Upper Limit	8
iGA Parameters' Setting	
Population Size per island (M)	20
Mating Rate	0.5
Mutation Rate	0.5
Migration Interval	20 *Iterations*

4.5.1 Performance of Island Genetic Algorithm

To evaluate the performance of applying the iGA to the DSA channel allocation problem, we have run experiments on network sizes ranging from 5 to 100 nodes and compared the results of the iGA to the optimal value. Pseudo-optimal $f(\mathbf{h})$ values for $N \geq 50$ were determined by running the iGA for an extended period of time, whereas the optimal values for $N < 50$ were found by exhaustive search. The exhaustive search for networks of size 25 nodes or less was feasible because, even with the large number of available channels and communication links, the upper limit of the available channels at each node is only 8 and the average is 5 channels. The available channels for each communication link is even less than this number because it can only be assigned to a channel that is available at its both end nodes. The large pool of available channels and the random channel assignment procedure used in our simulations presented in this chapter contribute in reducing the search space for our simulated networks because they decrease the number of available channels for each

communication link by decreasing the number of common channels between its end nodes. This results in an average number of available channels per link of about 1.4. Thus for a network of size 25 nodes and with an average number of links per node of about 3.5, the size of the search space equals $1.4^{25*3.5} \approx 6.1 \times 10^{12}$. As the size of the search space grows exponentially with the number of nodes, the exhaustive search was not feasible for larger networks of our simulation settings. In our simulations presented in Chapters 6 and 7, a better channel distribution procedure is used by assigning channels to network subareas instead of network nodes. All nodes within the same subarea are then assumed to have the same available channel set. We seek more correlation among the available channels at neighboring nodes using this channel assignment procedure.

Table 4.2 shows the results of running the iGA repeatedly on the simulated networks. For smaller networks, with fewer than several million possible solutions (indicated by the "Comb." column), the iGA always found the optimal solution. Also, the iGA converged to the optimum solution in about 30 iterations or less on average (indicated by the "Avg. Iter." column). Larger networks experienced some small variations in the final solution, but even with a search space on the order of 10^{74}, the iGA produced excellent solutions of above 96% of pseudo-optimal. Notice that we chose to allocate more simulation time to larger networks because the smaller networks had so little variation in the results. Interestingly, varying the migration interval, t, from 5 to 1000 had little effect on the 100-node network performance; most probably because of the large number of available channels we used in this simulations set.

The performance results in Table 4.2 establish that our iGA is able to achieve excellent performance when run repeatedly on the same set of networks. This indicates the consistency of the random outcome under static network conditions. In order to make sure that the performance was not coincident with the particular generated networks, we also simulated the iGA for a series of different randomly generated networks with 25 nodes. This particular size was selected so that fitness evaluation of all possible channel assignments was feasible and hence we can compute the optimal value using exhaustive search in feasible amount of

Table 4.2: Performance Results of Island Genetic Algorithm on Channel Allocation Problem ©2008 IEEE. Reprinted, with permission, from Friend *et al.*, "Architecture and performance of an island genetic algorithm-based cognitive network," in *Proc. IEEE CCNC'08*, (Las Vegas, NV), pp. 993–997, January 2008

Nodes	Pseudo-Optimal $f(\mathbf{h})$	Avg. $f(\mathbf{h})$	Std. Dev.	Comb.	Avg. Iter.	Rept.
5	9.0	9.0	0.00	3.2E+04	1	10
10	10.85	10.85	0.00	3.3E+05	20.1	10
15	15.12	15.12	0.00	1.3E+06	16.1	10
20	18.09	18.09	0.00	1.2E+07	29.6	10
25	21.63	21.62	0.01	1.5E+08	29.2	10
50	50.44	50.33	0.12	1.8E+29	1000	193
75	67.09	66.66	0.30	1.8E+56	1000	43
100	97.55	96.71	0.39	1.7E+74	1000	30

time. The results are shown in Table 4.3.

As was the case for repeated simulations on a 25-node static network, the iGA was very near the optimal solution every time with the average performance of the iGA as a percentage of the optimal value is above 99.9% and the standard deviation of different iGA runs output is only 0.05. The statistics for the number of iterations required to find the optimal solution are greater because the size of the search space for some of the networks (a function of the number of links and the size of the available channel sets) was one to two orders of magnitude

Table 4.3: Performance Results for Randomly Generated 25-Nodes Networks ©2008 IEEE. Reprinted, with permission, from Friend *et al.*, "Architecture and performance of an island genetic algorithm-based cognitive network," in *Proc. IEEE CCNC'08*, (Las Vegas, NV), pp. 993–997, January 2008

Nodes	25
Avg. % Optimal	99.97
Std. Dev. (%)	0.05
Avg. Combinations	1.9E+10
Avg. Iterations	47.5
Std. Dev. Iterations	36.4

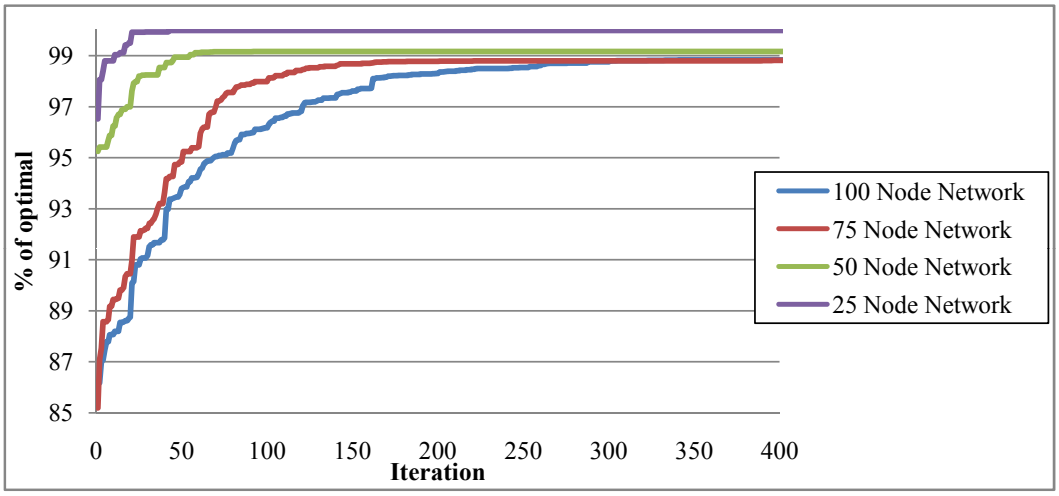

Figure 4.1: Effect of Network Size on Convergence Speed of the island Genetic Algorithm

larger than the 25-node network in Table 4.2.

4.5.2 Effect of Network Size on Convergence Speed

In order to investigate the scalability of the standard iGA and the need for learning iGA parameter settings, we have studied the effect of different network parameters on the iGA convergence speed. We can use these results, for example, to set an appropriate value for the maximum number of iGA iterations when augmenting our current work in cognitive network reasoning with learning algorithms. We have traced the intermediate results of the iGA of one of the simulation runs for each network size to study the effect of network size on the convergence speed of the iGA. Fig. 4.1 shows the curve of the iGA results for different network sizes: 25, 50, 75, and 100, fixing all other parameters.

The figure shows that the convergence speed of the standard iGA depends on the network size, i.e. the bigger the size of the network, the more number of iterations required to reach the performance stability region. This is reasonable because as the network size increases, the search space increases and it is becoming more difficult to reach to an optimal or suboptimal solution especially for the standard iGA that uses global information. The figure also shows

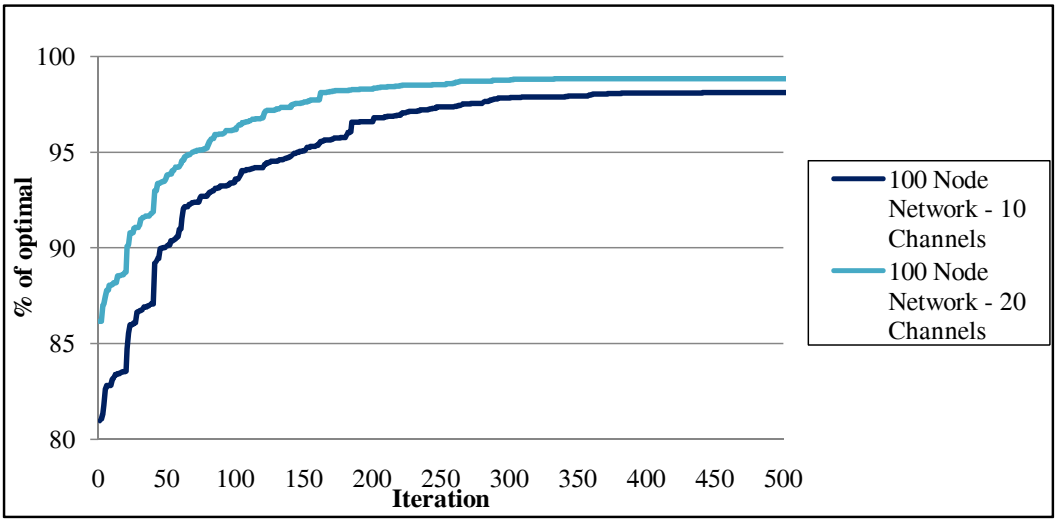

Figure 4.2: Effect of Available Channels on iGA Convergence Speed

that it takes no more than 300 iterations for the largest network size to stabilize. However, our guess is that this number depends on other network parameters including for example the number of available channels. Thus, our next simulations are to study the effect of the number of available channels.

4.5.3 Effect of Number of Channels on Convergence Speed

In these simulations set, we have studied the effect of the number of available channels on the iGA convergence speed by applying the iGA to different randomly generated networks of size 100 nodes with all parameters fixed but with different numbers of available channels, 10 and 20 channels. Fig. 4.2 shows the effect of changing the number of available channels on the iGA convergence speed.

The figure shows that as the number of available channels increases, the iGA convergence speed also increases; this is because it is becoming easier for the iGA to find an optimal solution. This means that we can limit the number of iGA iterations to smaller numbers as the number of available channels increases.

4.6 Summary and Contributions

Dynamic spectrum access (DSA) is one of the most popular applications of cognitive radios and cognitive networks. As discussed in Section 2.2.4.1, spectrum sharing is one of the major challenges for the DSA application. Hence, we have chosen the DSA channel allocation problem as our first cognitive network application. Our objective is to investigate the applicability of the island genetic algorithm (iGA) to solve the DSA channel allocation problem as a first step towards showing the applicability of the iGA as a single cognitive network distributed reasoning algorithm.

In order to clarify the need for developing localized variations of the iGA, we have applied the standard iGA first in this chapter to the DSA channel allocation problem and studied its performance through simulations. The simulated results of the iGA reveal excellent performance for this problem which shows that iGA is a promising algorithm to be used with cognitive networks. However, the major problem with using the standard iGA in communication and networking problems is that it is not scalable as shown from studying the effect of network size on iGA convergence speed. The standard iGA lacks scalability because even with distributing the computation over network nodes; each node still has to search the entire search space for a solution. This is not scalable for large scale problems and large networks. This motivates us to modify the standard iGA to use local information instead of global information.

The next chapter presents the details of a cognitive network implementation using commercial off-the-shelf hardware and open source software defined radio. The implementation applies the iGA to the channel allocation problem and experimental results are provided.

The work presented in this chapter has resulted in the following publication:

1. Daniel H. Friend, Mustafa Y. ElNainay, Yongsheng Shi, and Allen B. MacKenzie, "Architecture and performance of an island genetic algorithm-based cognitive network," in *Proc. IEEE Consumer Communications and Networking Conference (CCNC'08)*,

(Las Vegas, NV), pp. 993–997, 10–12 Jan. 2008.

Chapter 5

Cognitive Network Implementation using USRP and GNU Radio [1]

In this chapter, we present the details of a cognitive network implementation. In this implementation, we show how cognitive radios can be organized to form a cooperative ad-hoc cognitive radio network that utilizes the available spectrum opportunistically and efficiently through dynamic channel allocation. Our Cognitive Radios (CRs) are Software Defined Radios (SDRs) consisting of a Linux laptop and a Universal Software Radio Peripheral (USRP) integrated with our extended implementation of the GNU Radio software package and our island Genetic Algorithm (iGA) implementation for channel allocation. Section 5.1 introduces our system diagram for a cognitive node in a cognitive network. The hardware and software components of the cognitive node system diagram are then described in Section 5.2. Section 5.3 presents experiment settings and parameters and introduces the software tool used to evaluate network performance. Experimental results are then provided. Finally, the chapter is concluded in Section 5.4.

[1] This chapter is based on the work published in [70] and is a result of joint work with Feng (Andrew) Ge and Amr Hilal.

5.1 System Overview

In this chapter, we present the details of implementing a cognitive network (CN) demonstration using commercial off-the-shelf (COTS) hardware, and open source and our custom software. Our Cognitive Radios (CRs) are Software Defined Radios (SDRs) consisting of a Linux laptop and a Universal Software Radio Peripheral (USRP) integrated with our extended implementation of the GNU Radio software package and our iGA implementation for the channel allocation problem. We show how CRs can be organized to form a cooperative ad-hoc cognitive radio network that utilizes the available spectrum opportunistically and efficiently. We achieve this goal through cooperative dynamic channel allocation.

The channel allocation is performed using the iGA as our CN distributed reasoning algorithm to allocate channels to links based on the current network status. An iGA Controller triggers the iGA for a next round of channel allocations either periodically or when one or more of the network nodes suffer from performance degradation, probably because many of previously available channels become occupied by primary users or as new secondary users join the ad-hoc network. Moreover, we have developed signal detection and classification system to detect primary users as they become active and avoid causing interference to primary users. The following section describes briefly each of the hardware and software components of our cognitive node implementation.

5.2 System Hardware and Software Components

Fig. 5.1 shows the cognitive node system diagram for our CN implementation. The following sections briefly describe each of the hardware and software components.

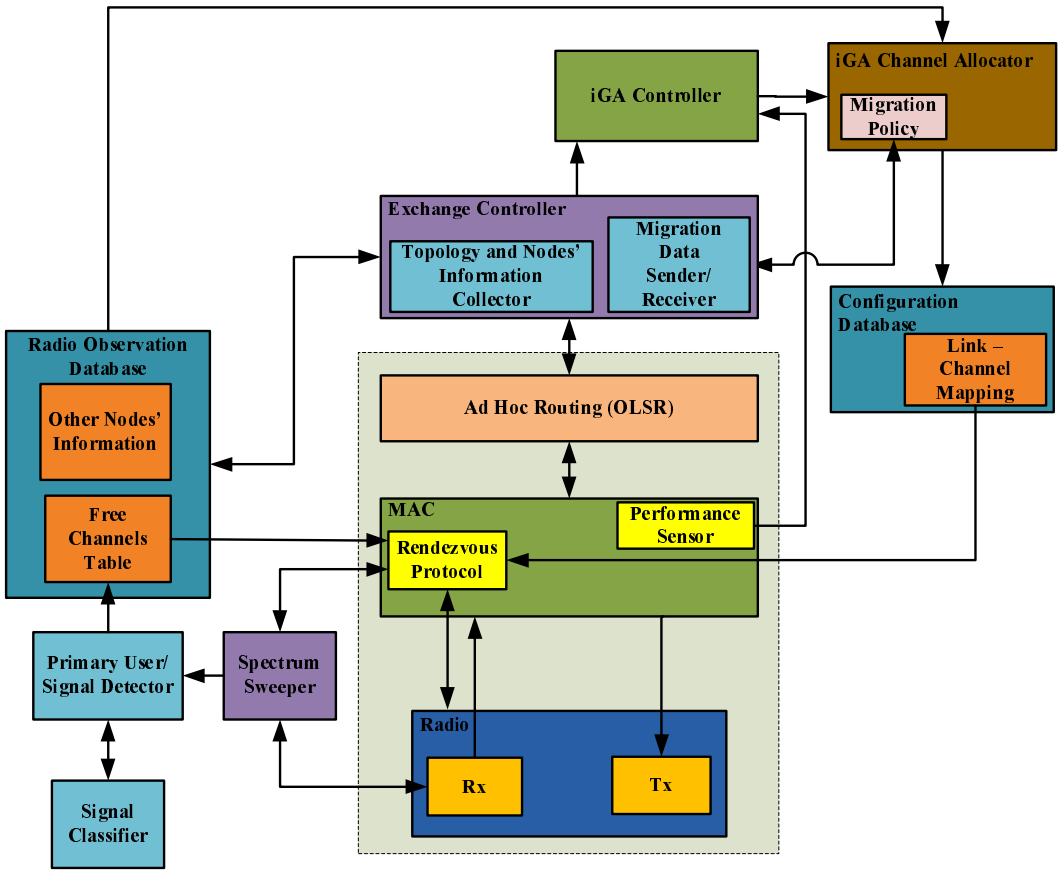

Figure 5.1: Cognitive Node System Diagram ©2009 IEEE. Reprinted, with permission, from ElNainay *et al.*, "Channel Allocation for Dynamic Spectrum Access Cognitive Networks using Localized Island Genetic Algorithm," in *Proc. TRIDENTCOM'09*, (Washington, DC), pp. 1–3, April 2009

5.2.1 Radio

To design a fully reconfigurable Physical/MAC layer, we need a Software Defined Radio (SDR) development platform. We have chosen GNU Radio as our SDR software architecture because GNU Radio is open source and the Universal Software Radio Peripheral (USRP) device as our SDR hardware platform because of the availability of a number of USRPs in the Center for Wireless Telecommunication (CWT) lab. GNU Radio [71] is an open source toolkit for building software radios. The USRP [72] is an openly designed low-price SDR hardware platform which implements the radio front-end, up and down conversion, and A/D (analog-to-digital) and D/A (digital-to-analog) conversion and uses the Universal Serial Bus (USB) to connect to the PC that hosts the device. The USRP is fully supported by the GNU Radio library. In our implementation, we use one USRP at each network node.

5.2.2 Spectrum Sweeper - Signal Detector

Our spectrum sensor, relying on a USRP as the RF front end, is a power spectral density (PSD) sensor. As shown in Fig. 5.2, it is a Fast Fourier Transform (FFT)-based energy detector. It uses GNU Radio and a USRP to collect baseband signal samples, applies a windowing filter, and then calculates the FFT. The resulting discrete power spectrum density is then windowed again to reduce noise irregularity. This averaged spectral data, along with a user specified minimum threshold are used for energy detection. Any spectral energy above the threshold is considered to represent active spectrum. Spectrum with energy below the defined threshold is considered whitespace (spectrum hole). Energy detection is used for two reasons: fast timing restriction and lack of knowledge of the incoming signal.

5.2.3 Signal Classifier

If a signal is detected, our signal classification algorithm [73] shown in Fig. 5.3 is used to classify the received signal and decide whether it is a primary user signal or not. The

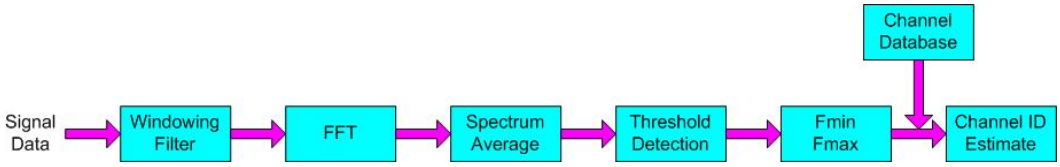

Figure 5.2: A Power Spectral Density (PSD) Sensor ©2009 IEEE. Reprinted, with permission, from ElNainay *et al.*, "Channel Allocation for Dynamic Spectrum Access Cognitive Networks using Localized Island Genetic Algorithm," in *Proc. TRIDENTCOM'09*, (Washington, DC), pp. 1–3, April 2009

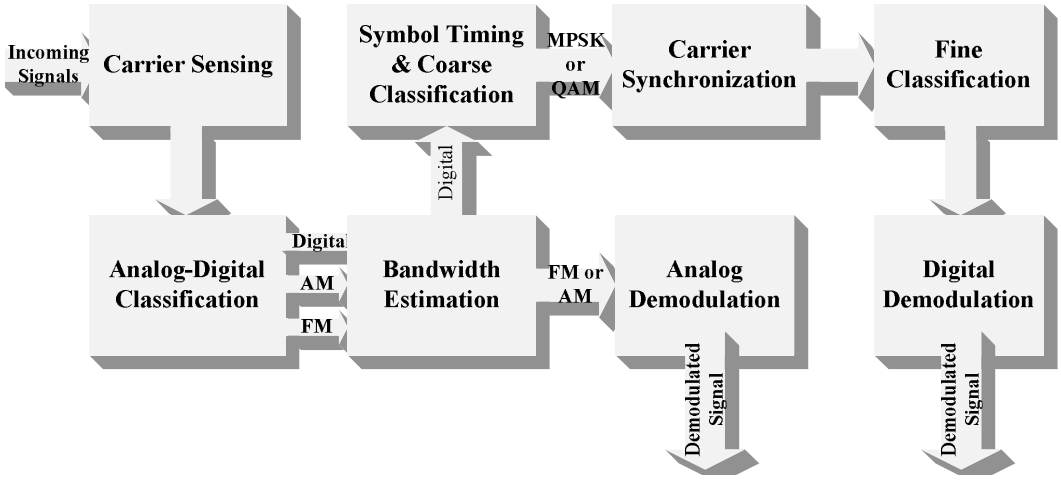

Figure 5.3: Signal Classifier ©2009 IEEE. Reprinted, with permission, from ElNainay *et al.*, "Channel Allocation for Dynamic Spectrum Access Cognitive Networks using Localized Island Genetic Algorithm," in *Proc. TRIDENTCOM'09*, (Washington, DC), pp. 1–3, April 2009

Analog-Digital classification module is used to categorize the signal into one of the following: analog AM, analog FM, or digital modulation. Based on this categorization the next module, bandwidth estimation, will provide a more accurate result. The bandwidth estimated in this module will provide enough information for AM and FM demodulation.

If the received signal is a digital signal, the data will be fed into the symbol timing and coarse classification module. In this module, symbol rate is estimated and signal is sampled at the peak of each symbol; meanwhile, the signal is also classified as QAM or MPSK. This module also tells the order of QAM, for example: 16QAM, or 64QAM. Carrier synchronization is used to estimate and compensate for the carrier frequency offset and phase offset. Now all

the information required for digital demodulation will have been extracted from the incoming signal, and we can finally implement demodulation. Knowing the features of the primary users, all signals that conform to these features are considered as primary users. If the signal is classified as a primary user/signal, an entry will be inserted into radio observation database to mark this channel as unavailable, otherwise this channel will be considered free to be used by secondary users.

5.2.4 Observation Database - Radio Database

Databases of our architecture are implemented in MySQL. The radio observation database combines neighboring nodes' spectrum detection results and stores them in a table that includes active primary users signals' carrier frequency, bandwidth, power, modulation, sensor label (to be replaced by sensor location when available), and the time that each signal was observed. Based on this table, each node is able to identify available channels that can be used by secondary users.

5.2.5 MAC Layer

Since our channel allocation algorithm does not guarantee a conflict free assignment, we use a multiple access technique to access the allocated channel. We use a modified implementation of the Carrier Sense Multiple Access (CSMA) as our MAC protocol. When a node has some data to send, it first checks the link-channel mapping for the allocated channel to the required destination (next hop to destination). If there is no assignment (possibly because of node mobility or new node just joined the network), a Rendezvous protocol is used to communicate with the required destination. If there is an allocated channel, the node then checks the free channel table to ensure that this channel is still available for use and sends the data using our CSMA implementation. If the channel is not currently available because of a primary user activity, the Rendezvous protocol is used to communicate with the required destination

and this link-channel relation is inserted into the link-channel mapping table.

5.2.6 Ad-hoc Routing Protocol

In this work, we use an open source implementation of Optimized Link State Routing (OLSR) [74] as our routing protocol. OLSR is a proactive link-state routing protocol which uses Hello and Topology Control (TC) messages to discover and then discriminate link state information throughout the ad-hoc network. Individual nodes use this topology information to compute next hop destinations for all nodes in the network using shortest hop forwarding paths. With its underlying MultiPoint Relays (MPRs) technique [75], routing control messages and migration information of the island genetic algorithm (iGA) can be delivered to all designated destinations with substantial reduction in the communication cost compared to flooding.

5.2.7 Exchange Controller

The Exchange Controller is responsible for exchanging information (observations/knowledge) with external nodes. The exchange controller uses OLSR-like control messages to relay spectrum information requests and migration information messages from the iGA to neighboring nodes. The exchange controller receives spectrum information replies from other nodes and uses them to maintain other nodes' spectrum information stored in the radio observation database. Moreover, the exchange controller responds to other nodes' spectrum information requests with spectrum detection results stored in the radio observation database. The exchange controller and all iGA components are coded using C++ language. They are coded as two processes and memory sharing is used as the inter-process communication mechanism.

5.2.8 Island Genetic Algorithm Controller

The island Genetic Algorithm Controller is responsible for monitoring the system performance and deciding when it is necessary to optimize the current network configuration as well as how the iGA process should be configured. The controller also communicates with other nodes to trigger their iGA processes and collects topology and spectrum information changes since last trigger.

5.2.9 Configuration Database - Link-Channel Mapping

The configuration database is used to maintain the final solution of the iGA. Once the iGA Channel Allocator finishes its work, the final link-channel mapping is stored in a table in the configuration database. This table is then used as a reference whenever this node wants to communicate with any other neighboring node.

5.2.10 Island Genetic Algorithm Channel Allocator

The main task for the island Genetic Algorithm (iGA) in this implementation is to search for the best possible link-channel mapping to maximize the utilization of the available spectrum. We emphasize here that we plan to use the iGA in a more general way as the distributed reasoning algorithm for cognitive networks (CNs) and that this implementation is acting as a proof of concept.

After the iGA controller decides to trigger another round of the iGA process of this node, the iGA controller communicates through the exchange controller with other neighboring nodes to trigger their iGA processes and collects the necessary topology and spectrum information for the iGA process to work. The iGA then starts with the last population from the last run as the initial population for the next run (the first run/first population is randomly generated). Using the last population to seed the new run improves the iGA performance

in case of few network changes while a diversity technique, similar to the one used in [69], is used to avoid premature convergence.

In our implementation, we use a simple communication-efficient migration policy in which each node shares with its neighbors only the channel assignment of its outgoing links from its best individual, every fixed migration interval. Each node receiving information from any other node will merge this information with all its individuals if the merged information will give better fitness. Otherwise the receiving node will merge it with only part of its population that is proportional to the fitness of the best individual without the merged information to the fitness with the merged information. This forces each node to consider solutions of other nodes even if it will produce lower fitness individuals than are in the current generation.

5.3 Experiment Settings and Results

In this section, we describe the network settings used to evaluate the performance of our cognitive network (CN) implementation and provide the results of our experiments. We first describe the island genetic algorithm (iGA), and network setting and topology used. We then introduce Iperf, the software tool used to measure the network performance. Finally, we provide the experimental results with discussion.

5.3.1 Island Genetic Algorithm Settings

Table 5.1 summarizes the iGA parameters' setting that are used to run our experiments. we have set the population size M to 20 for all nodes, the *keep rate* to 0.5, the *mutation rate* to 0.3, the *migration interval* to 20 iterations, and the total number of iterations to 500.

We had to insert delays into the iGA code to give neighboring network nodes the chance to exchange information. This is mainly because we operate the iGA in asynchronous mode and because the size of the network is very small compared to our simulations presented in

Table 5.1: Island Genetic Algorithm Parameters' Setting

iGA Parameters' Setting	
Population Size per island (M)	20
Number of Iterations	500
Keep Rate	0.5
Mutation Rate	0.3
Migration Interval	20 Iterations

Chapter 4. Without these delays, the iGA code on one node may execute and terminate before getting the chance to receive any migrated individuals from other nodes due to the difference in speed between the iGA execution time and the network information exchange time. This also reveals one of the potential challenges in applying iGA in real networks and suggests the use of the iGA in planning solutions rather than in searching for immediate solutions especially for time critical problems.

5.3.2 Network Settings

In our experiments, we use four Dell laptops running Ubuntu operating system to form a chain topology as shown in Fig. 5.4. Chain topology networks are commonly used in measuring network performance as they are simple to set up. In order to set up a multi hop network in a small area, we have used the linux *rtable* command to add routing rules at each node to block MAC addresses of all but neighboring nodes. This way we have forced end nodes to go through the intermediate nodes to route data instead of sending data packets directly to the other end node. All laptops are core duo 2GHz with 2 GB of RAM. Notice that this solution builds a multihop network but suffers from high interference because we still have all USRPs in a relatively small area.

For ease of implementation, we have built a control channel for exchanging OLSR messages and iGA messages over WiFi network access cards and use only one USRP daughterboard for sending and receiving data. In our physical layer implementation, Gaussian Minimum Shift

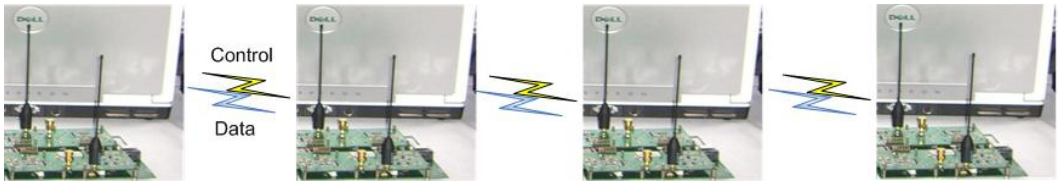

Figure 5.4: Network Topology for the Experiment

Keying (GMSK) modulation has been used. The data rate has been set to 25 kbps. Each node is able to receive on three channels at the same time but transmit on only one channel at a time. This is because the channels are relatively close together and transmitting on one channel overwhelms the three channel bands on the receive side. This is also another reason why we currently switch between transmission and reception on a single daughterboard instead of using two daughterboards, one for transmitting data and one for receiving data. Because of the single channel transmission, we have modified the iGA code to assign channels to nodes instead of links as this simplifies the iGA objective and results in less conflicts.

Each node is assigned two static Internet Protocol (IP) addresses with different subnets, one for the WiFi card and the other for the USRP. Using different subnets come after noticing that the operating system forwards all packets to the WiFi card when using same subnet for both the WiFi card and the USRP (possibly because we activate the WiFi card before the USRP). To work around having the OLSR routing working over WiFi and routing data over USRPs, we have modified the OLSR code to add two routes one for the WiFi network and the other for the USRP network each time it finds a new route or modify an existing one. Each of these two routes is associated with the appropriate interface, either WiFi or USRP, and with a modified destination and next hop IP addresses for the USRP entry. To facilitate adding the second route entry information, we have used a direct mapping between the WiFi IP address and the USRP IP address. With this modification, we have all control packets routed over the WiFi and all data packets routed over the USRP. We have used the Iperf software tool to create data transmissions among nodes and to measure the network performance. The next section introduces the Iperf tool and its capabilities.

5.3.3 Iperf

Iperf is a software tool written in C++ that can be used to measure end to end network performance between client and server nodes running Iperf. Iperf allows the user to set various parameters on different layers ranging from application layer to IP layer that can be used for testing a network. Iperf has a client and server functionality, and can measure the throughput between the two ends, either uni-directonally or bi-directionally. It is open source software and runs on various platforms including linux, unix and windows. It is supported by the National Laboratory for Applied Network Research.

Iperf allows the user to create UDP or TCP data streams. For UDP data streams, the user sets the UDP bandwidth and optionally sets the buffer size and packet size. For TCP data streams, user can set the TCP window size, buffer length, and maximum segment size. Data streams can be limited by either a transmission time period or amount of data. Different performance metrics can be measured with Iperf including: packet loss, network throughput and jitter.

Jperf is a graphical user interface (GUI) for Iperf written in Java to facilitate using Iperf, as shown in Fig. 5.5. Instead of using command line text commands, Jperf allows users to set all options and run Iperf commands through a friendly GUI. However, user may still prefer using text commands through command line windows because this gives more control over the possible usable range of some parameters. We use Iperf along with Jperf frontend to run some communication sessions among network nodes and measure the throughput and jitter of these communication sessions. The next sections describe in more details the data transmissions setting and provide the results of our experiments.

5.3.4 Network Latency

In our experiments, we measure the network latency in terms of Round Trip Time (RTT) using the *Ping* command. Table 5.2 provides the results for our multi-channel cognitive

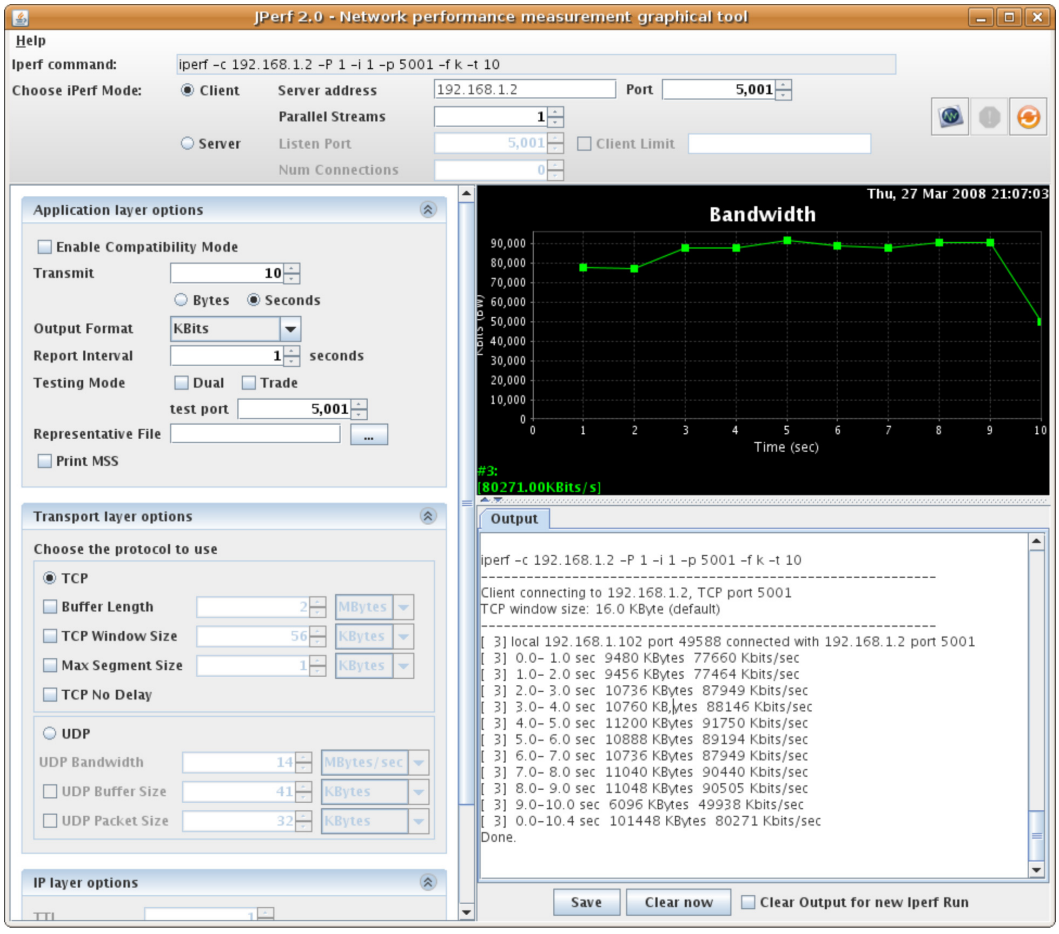

Figure 5.5: Jperf Screenshot

Table 5.2: Multi-Channel Vs Single-Channel Network Latency

	3-hops		**2-hops**		**1-hop**	
	Multi	**Single**	**Multi**	**Single**	**Multi**	**Single**
Avg RTT (ms)	282.01	833.91	188.49	529.67	96.96	213.01
Std. Dev. (ms)	18.09	40.78	14.60	31.27	9.99	20.04
Packet Loss	18%	9%	10%	7%	8%	0%

network compared to the single channel network. We ping the first node of the chain from the fourth node for 3-hops results, the second node from the fourth node for 2-hops results, and the third node from the fourth node for 1-hop results. The table provides the average RTT, the standard deviation of RTT, and the packet loss percentage. Results are averaged over multiple runs and for each run measurements from the first 15 packets are omitted due to transience associated with network startup.

The results show that the RTT of multi-channel CN is much better than the single-channel network for all experiments. The RTT standard deviation of multi-channel CN is about one-half the single-channel network RTT standard deviation. This is intuitive because the use of multiple channels results in fewer MAC collisions and hence less backoff delays. Interestingly, the packet loss percentage of the single-channel network is lower than the percentage of the multi-channel CN. We attribute this to the fact that we have experienced frequency offsets of different amounts with each USRP at each channel. This increases the probability of packet losses in the multi-channel network more than for the single-channel network.

5.3.5 Network Throughput

Using Iperf, we have run User Datagram Protocol (UDP) data transmission sessions from the fourth node of the chain as client to the first node as server. The period of each data transmission session is 100 seconds but the first 20 seconds of each run is then omitted as a transient period. The results are then averaged over periods of 10 seconds. The network throughput and jitter, and the packet loss are analyzed and presented in this and next

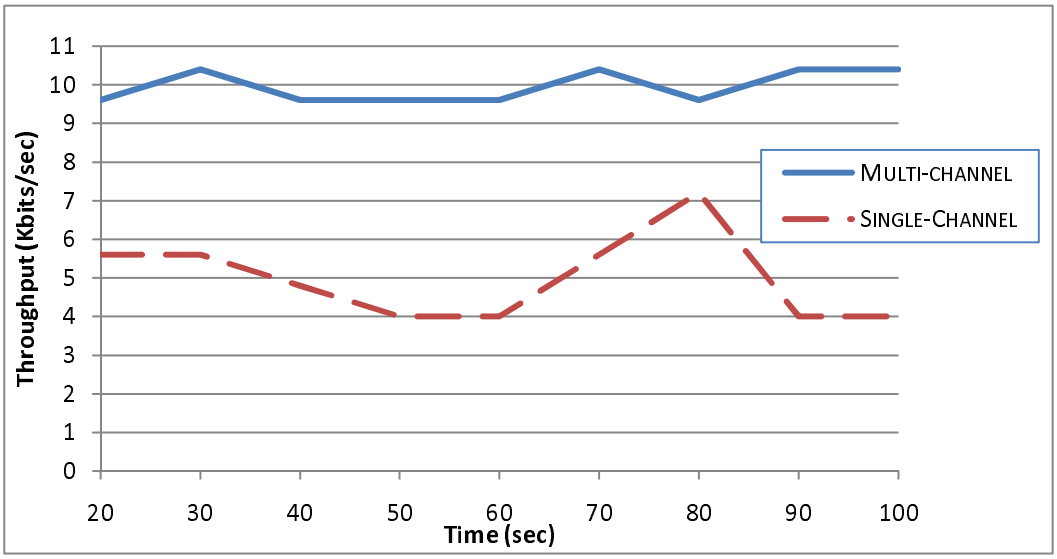

Figure 5.6: Multi-Channel Vs Single-Channel Network Throughput

sections. The UDP bandwidth is set to 10 Kbits/sec, the packet size is set to 1 Kbits, and the UDP buffer size is set to 5 Kbits (5 times the packet size). Fig. 5.6 compares the network throughput of the multi-channel CN to the single-channel network.

While the multi-channel CN can achieve average network throughput that equals to the UDP bandwidth, the single-channel network throughput is about half this value. Fig. 5.6 also shows that the performance of the multi-channel CN is more stable than the single-channel network performance.

5.3.6 Network Jitter

Fig. 5.7 shows the network jitter for the multi-channel CN and single-channel network. While the single-channel network experiences high jitter (over 300 msec), the multi-channel CN has a very small jitter value (less than 10 msec). The single-channel network jitter is also increasing with time because of increasing backoff and queuing delays while the multi-channel CN jitter is stable with time.

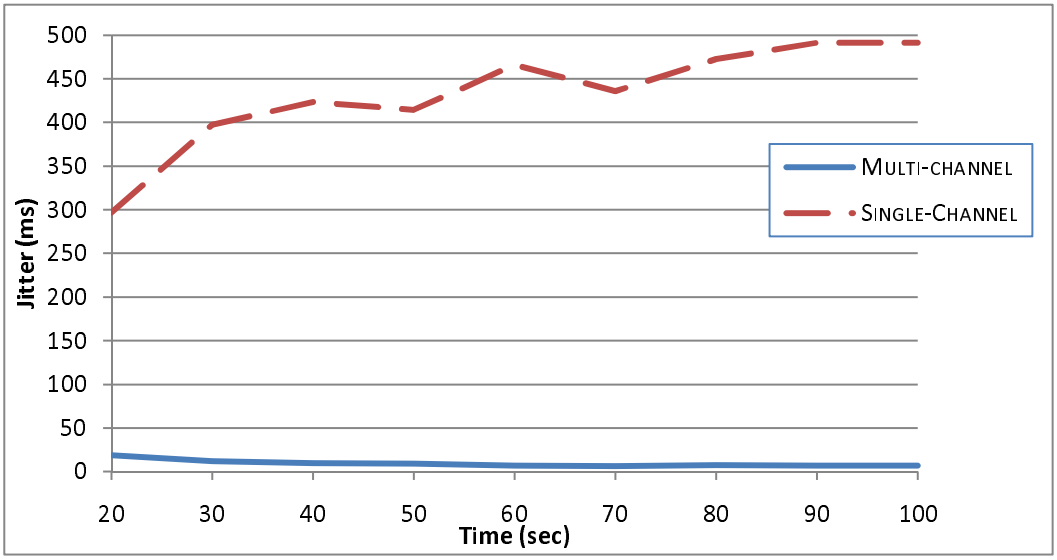

Figure 5.7: Multi-Channel Vs Single-Channel Network Jitter

5.3.7 Packet Loss

Fig. 5.8 shows the packet loss percentage for the multi-channel CN and single-channel network. The packet loss percentage of the multi-channel CN is stable to almost zero all the time except for a spike around 50 second. On the other hand, the single-channel network experiences a high packet loss of about 50%. This is different from our results reported in the network latency section. Two reasons for this change, the first is that the packet size is larger in UDP data transmissions than the ping packet size which increases the backoff and queuing delay for the single-channel network. The other reason is that the UDP data transmission is a continuous transmission that does not wait for a reply for each packet to transmit the next one as in the pinging. This results in more data packets flowing into the network and hence higher possibility for collisions and increasingly backoff and queuing delays especially for the single-channel network.

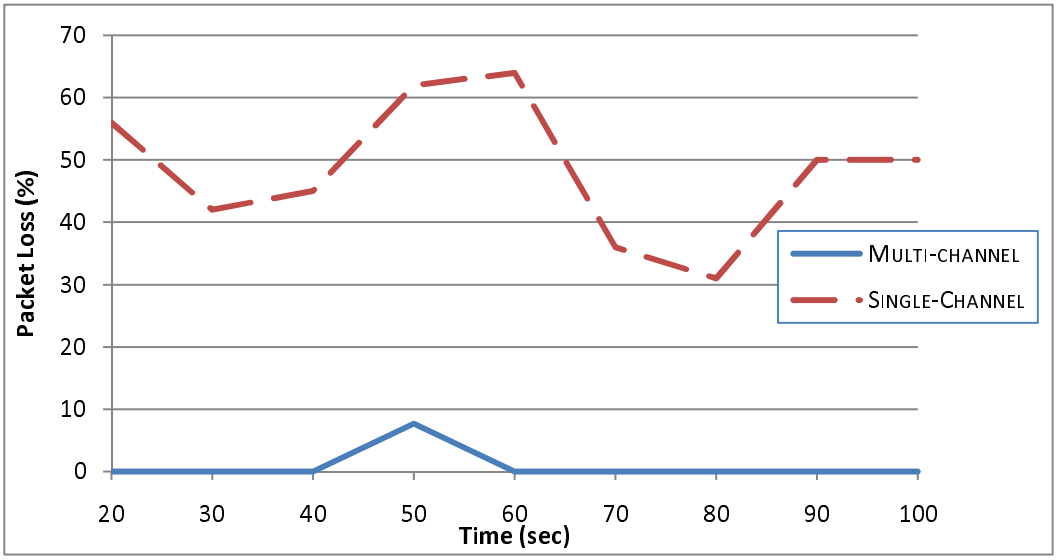

Figure 5.8: Multi-Channel Vs Single-Channel Packet Loss

5.4 Summary and Contributions

In this chapter, we have presented a commercial off-the-shelf implementation for a cognitive node that can cooperate with other network nodes to utilize the available spectrum opportunistically and efficiently through channel allocation. We have then used the island genetic algorithm (iGA) as our distributed reasoning algorithm to perform the channel allocation. We have briefly described each of our node architecture hardware and software components. Experimental results are then provided using Iperf software tool to measure the network performance. The experimental results show that the multi-channel performance is much better and more stable compared to the single channel network.

Having applied the iGA to the dynamic spectrum channel allocation problem both through simulations and in a real, functioning system, the next part of this dissertation presents our localized variations of the iGA and applies them to more complex and cross layer communication and networking problems.

The work presented in this chapter has resulted in the following publication:

1. Mustafa Y. ElNainay, Feng (Andrew) Ge, Ying Wang, Amr E. Hilal, Allen B. MacKenzie, and Charles W. Bostian, "Channel allocation for dynamic spectrum access cognitive networks using localized island genetic algorithm," in *Proc. the 5th International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TRIDENTCOM'09)*, Washington, D.C., pp. 1–3, 6-8 April 2009.

Part III

Localized Island Genetic Algorithms to Cross Layer Problems

Chapter 6

Joint Power and Channel Allocation Problem [1]

In Chapter 4 and Chapter 5, we have applied the standard island genetic algorithm (iGA) to the dynamic spectrum access (DSA) channel allocation problem and studied its performance. Moreover, we have identified the scalability problem of the iGA. Having demonstrated the applicability of the iGA to one of the challenging problems in the DSA application, namely channel allocation, we shift our focus in this chapter and next chapter to developing more promising variations of the iGA to be used as the cognitive network reasoning algorithm and apply them to cross layer problems.

In this chapter, we develop a localized variation of the iGA (LiGA) and apply it to the joint power and channel allocation problem. Section 6.1 introduces the joint power and channel allocation problem. Related work that apply optimization and heuristic algorithms to the joint power and channel allocation problem are surveyed in Section 6.2. A distributed cross layer formulation of the joint power and channel allocation problem is developed in Section 6.3. Section 6.4 then introduces our localized variation of the iGA, LiGA, and presents the details of how the LiGA is applied to the formulated problem. Simulation results that

[1] This chapter is based on the work in [76], [77], and [78].

compare the LiGA to the standard iGA and study the effect of power control on the LiGA performance are provided in Section 6.5. Section 6.5 also provide simulation results for the effect of nodes' non-cooperation on the network performance. Finally, the chapter is concluded in Section 6.6.

6.1 Power Control Problem

Power control has a big influence on the performance of wireless networks in general and cognitive networks (CNs) in particular because it affects both the efficiency and feasibility of the channel allocation, and hence the spectrum utilization. For example, if there is no power control at all, this means that all nodes have to transmit using the default power level (usually the maximum level), which subsequently decreases the number of possible interference-free channel assignments and results in performance degradation. Power control gives each node the option to adjust its transmission power to compromise between transmission rate/range and the interference it causes to other nodes. This results in better overall network performance by making more channel assignments feasible.

In this chapter, we consider how an ad-hoc CN in a dynamic spectrum environment can utilize the available spectrum efficiently by applying a localized variation of the island genetic algorithm (LiGA) to the joint power and channel allocation problem. We formulate the distributed cross layer channel allocation and power control problem for an ad-hoc CN in dynamic spectrum access (DSA). We then apply the LiGA to the cross layer channel allocation-power control problem. For comparison purposes, we also implement the standard island genetic algorithm (iGA), which works with global information (complete knowledge), and compare it to the localized iGA to analyze the effect of working with partial knowledge on network performance. Simulation results are provided.

Moreover, we study the effect of non-cooperation on the CN performance. Most of the work in CN and our work so far assume the willingness of network nodes to share information and

cooperate neglecting two practical issues. The first issue is the existence of non-cognitive nodes, which are not capable of such cooperation, because, even if we assume that cognitive nodes will dominate at some point in the future, there will be a transition period (of several years) during which cognitive nodes have to work with non-cognitive nodes, i.e. in a mixed network of cognitive and non-cognitive nodes. The second problem is the possible existence of selfish nodes among the cognitive nodes. We study the effect of these two problems on the performance of the CN as applied to the joint power and channel allocation problem.

6.2 Related Work

In this section, we review some closely related work that applies optimization and heuristics algorithms to the joint power and channel allocation problem in the context of cognitive radio networks (CRNs) and ad-hoc networks. Other work on the joint channel allocation and power control problem exists in the literature, but we limit our review to work related to ours.

In [79] and [80], Hoang and Liang propose two different solutions to improve the network throughput of CRNs through channel and power allocation. In [79], Hoang and Liang propose a heuristic channel allocation-power control algorithm to maximize the spectrum utilization of a CRN based on constructing a dynamic interference graph. They also formulate a control framework that guarantees protection of the primary users from interference. They provide a realistic interference model based on Signal-to-Interference plus Noise Ratio (SINR). In [80], Hoang and Liang also propose a Two-Phase Resource Allocation (TPRA) scheme to improve network throughput. In the first phase of their scheme, channels and power are allocated to base stations. Then in the second phase, each base station allocates channels within its cell. However, all solutions still use the cellular networks model in which channels are assigned to base station nodes. In [81], Digham models the joint power and channel allocation problem for cognitive radios and proposes a near-optimal yet simple algorithm with

linear complexity to solve a relaxed convex version of the original problem targeting total sum capacity maximization of a cognitive radio network. However, all these algorithms are working in a centralized offline mode in which all channel allocation-power control decisions are determined offline and signaled first to base stations and then signaled from base stations to cognitive nodes. Hence, these algorithms cannot work in a dynamic spectrum/network environment and have scalability problems as the network size grows. Moreover, the spectrum utilization and network throughput in models presented in [79] and [80] are defined as the total number of nodes the base station serves which is not suitable for a multihop ad-hoc cognitive network.

In [82], Ma and Tsang present a cross layer design including channel allocation and power control. They develop a binary integer linear program formulation for the cross layer problem. They simplify their formulation by assuming that all nodes transmit at the same power level. Moreover, they assume static node locations and static set of available channels at each node. They work in a centralized mode in which a centralized server collects information from all network nodes and determines the optimal channel-power level assignment, and then informs the decisions to nodes. However, these assumptions limit the applicability and efficiency of their approach. The required centralized server again limits the scalability and imposes high communication cost to exchange information with all other nodes. Assuming a common transmission power to all nodes simplifies the formulation but also limits efficiency. In fact in [83], Behzard and Rubin prove that for the special case where all nodes use the same transmission power, the maximum throughput is achieved when all nodes use the maximum transmission power level and this is independent of nodal distribution, traffic pattern, and offered traffic load.

In [84] and [85], Huang and Letaief develop a cross layer scheduling and power control framework for wireless ad-hoc network. Then, they formulate the power control problem as a non-cooperative game to facilitate distributed implementation of the cross layer problem. Under this game, each node assumes that interference from other nodes does not change and takes its decision to maximize its own utility function. Although the paper presents

a distributed implementation of a solution to the cross layer scheduling and power control problem, the implementation is tested with small size networks, 10 nodes in [84] and up to 30 nodes in [85], which does not give a real picture of efficiency. Moreover, this model targets wireless ad-hoc networks where every node has the same number and the same set of available channels. In CRN, each node has different number and set of available channels because available channels are location and time dependent based on primary users' interaction patterns.

In [86], Chin *et al.* propose a method utilizing both power control and channel assignment to satisfy the minimum Quality of Service (QoS) required for an ad-hoc network. The model again is for ad-hoc networks and essentially based on cellular network model and both of them do not model the dynamic spectrum accurately. Moreover, they assume a static environment and work in a centralized mode.

Our work presented in this Chapter is different from all previous work in that we are applying a *localized distributed* algorithm, localized island genetic algorithm (LiGA), to a cross layer channel allocation and power control formulation that is unique for *ad-hoc networks* in a DSA environment. The next section presents our system model for the joint power and channel allocation problem.

6.3 System Model

We first introduce the channel allocation and power control models separately. Then, we show how they are combined to represent the cross layer problem.

6.3.1 Channel Allocation Modeling

Consider a cognitive network (CN) consisting of a set of nodes, $\mathcal{N}$. Each node $i \in \mathcal{N}$ is able to transmit and receive simultaneously on different channels. Denote $l_{i,j}$ as the directed

communication link for which node i is the transmitting node and node j is the receiving node. Using directed rather than undirected communication links allows for full duplex transmissions by assigning different channels to links $l_{i,j}$ and $l_{j,i}$. We allow nodes to perform full duplex multi-channel communication because this is the most general case. Although current technology may be more restrictive, the state of the art in wideband multichannel radios is quickly approaching this scenario. Half-duplex operation may be modeled by forcing the channel assignment for half-duplex link pairs to be the same. This modification cuts the dimension of the search space in half, but requires no significant changes to our solution.

Assuming that the available spectrum consists of a set of non-overlapping channels, $\mathcal{C}$, and that each node i has detected (perhaps jointly with other nodes) a set of channels available for local use, $\mathcal{C}_i \subseteq \mathcal{C}$. Based on these node-centric sets, each link $l_{i,j}$ has a set of valid channel assignments, $\mathcal{H}_{i,j} = \mathcal{C}_i \cap \mathcal{C}_j$. Define $\mathbf{h} \in \times_{l_{i,j} \in \mathcal{L}_C} \mathcal{H}_{i,j}$ as the channel assignment vector, where $\times_{l_{i,j} \in \mathcal{L}_C} \mathcal{H}_{i,j}$ is the Cartesian product of the available channel sets for all communication links and $\mathcal{L}_C$ is the set of all directed communication links. Denote $h_{i,j}$ as the element of $\mathbf{h}$ that is the channel assigned to link $l_{i,j}$.

6.3.2 Power Control Modeling

In this work, we follow an interference model that assumes that interference is a binary condition, i.e. either pair of links interfere with each other or they do not. In this sense, our interference model resembles the protocol (disc) model. However, we differ from the protocol model in that we allow interference to be determined based on interference power, random fading, and shadowing. Since we do not account for additivity of interference, our model falls somewhere between the protocol model and the physical model. We believe that taking the additivity of interference into account may change the absolute results of our solution approach but not the conclusions.

Under this interference model, consider the general power control case where each node i can transmit with any transmission power up to a node-dependent maximum transmission

power, P_i. Define $g_{i,j}$ as the channel gain from a transmitting node i to a receiving node j,

$$g_{i,j} = d_{i,j}^{-\gamma}, \tag{6.1}$$

where $d_{i,j}$ is the distance from node i to node j, and γ is the path loss index. Following the approach of [87], we assume that a transmission from node i to node j is considered successful only if the received power at node j exceeds a certain threshold, say α. Then, for a successful reception the following must be satisfied:

$$p_{i,j} \cdot g_{i,j} \geq \alpha, \tag{6.2}$$

where $p_{i,j}$ is the transmission power assigned to link $l_{i,j}$. Similarly, we assume that the interference resulting from node i when transmitting with $p_{i,j}$ is non-negligible only if it exceeds a certain threshold, say β. Thus, link $l_{i,j}$ interferes with any link terminating at node k when the following condition holds:

$$p_{i,j} \cdot g_{i,k} \geq \beta. \tag{6.3}$$

Now define $p_{i,j}^{min} = \alpha / g_{i,j}$ as the minimum transmission power required for node i to transmit successfully to node j. Then the set of allowable transmission power values for link $l_{i,j}$ is $\mathcal{P}_{i,j} = \{p : p_{i,j}^{min} \leq p \leq P_i\}$. We then define $\mathbf{p} \in \times_{l_{i,j} \in \mathcal{L}_C} \mathcal{P}_{i,j}$ as the transmission power assignment vector.

6.3.3 Cross Layer Modeling

In this section, the channel allocation model and the power control model are combined to represent the cross layer channel allocation and power control problem. First, recall that $\mathcal{L}_C$ is the set of all directed communication links that can be established in the network when nodes transmit at maximum power. Then define $\mathcal{G}_C = (\mathcal{N}, \mathcal{L}_C)$ as the communication

graph. Similarly, define $\mathcal{G}_I = (\mathcal{N}, \mathcal{L}_I)$ as the interference graph where $\mathcal{L}_I$ is the set of all directed interference links indicating which nodes are interfered with when a node transmits at maximum power. In more formal terms, $l_{i,j} \in \mathcal{L}_I$ if $P_i \cdot g_{i,j} \geq \beta$. Maximum power conditions are used to capture the largest possible set of communication and interference links.

The graphs $\mathcal{G}_C$ and $\mathcal{G}_I$ require global information to construct. Since we are interested in localizing the information needed for each node to perform its computations, we define the set of communication links with which node i can potentially interfere as:

$$\begin{aligned} \mathcal{L}_C^i &= \{l_{j,i} \in \mathcal{L}_C\} \cup \\ &\quad \{l_{k,m} \in \mathcal{L}_C : \exists l_{i,m} \in \mathcal{L}_I\}. \end{aligned}$$

Notice that this set includes all links that terminate at i. Each link is assigned a channel and a power level, resulting in the pair $(h_{j,k}, p_{j,k}) \in \mathcal{H}_{j,k} \times \mathcal{P}_{j,k}$. The set of all allowable channel-power assignment pairs for the links in $\mathcal{L}_C^i$ is denoted by:

$$\mathcal{V}_i = \times_{l_{j,k} \in \mathcal{L}_C^i} (\mathcal{H}_{j,k} \times \mathcal{P}_{j,k}). \tag{6.4}$$

The localized channel-power assignment vector for node i is then denoted by $\mathbf{v}_i \in \mathcal{V}_i$.

We use $\mathcal{L}_C^i$ as the largest possible set with which node i need consider. However, in order to evaluate a particular $\mathbf{v}_i$, the node must determine the interference relationship between links under this particular localized channel-power assignment vector. To do this, we define

the set of links that interfere with link $l_{j,k}$ under the channel-power level assignment $\mathbf{v}_i$ by:

$$\begin{aligned}\mathcal{L}_{j,k}^{\mathbf{v}_i} &= \{l_{j,y} \in \mathcal{L}_C^i : h_{j,k} = h_{j,y}, y \neq k\} \cup \\ &\quad \{l_{y,j} \in \mathcal{L}_C^i : h_{j,k} = h_{y,j}\} \cup \\ &\quad \{l_{y,k} \in \mathcal{L}_C^i : h_{j,k} = h_{y,k}, y \neq j\} \cup \\ &\quad \{l_{k,y} \in \mathcal{L}_C^i : h_{j,k} = h_{k,y}\} \cup \\ &\quad \{l_{y,z} \in \mathcal{L}_C^i : \exists l_{y,k} \in \mathcal{L}_I, \\ &\quad h_{j,k} = h_{y,z}, d_{y,k} \leq R_y^I(p_{y,z})\},\end{aligned}$$

where the indices j and k are fixed, j ≠ k, and the indices y and z are variables. $R_y^I(p_{y,z})$ is the interference range of node y while transmitting over link $l_{y,z}$ with transmission power $p_{y,z}$. This can be calculated as a percentage of the maximum interference range of node y when transmitting with its maximum power P_i. Notice that this set includes all links that originate or terminate at j or k and are assigned to the same channel assigned to the link $l_{j,k}$ (As noted in Section 4.3, we can define $\mathcal{L}_{j,k}^{\mathbf{v}_i}$ in terms of the conflict graph, $\mathcal{F}$).

The localized objective for node i is then to optimize the following:

$$\max_{\mathbf{v}_i} \left[f(\mathbf{v}_i) = \sum_{l_{j,k} \in \mathcal{L}_C^i} \left(\frac{p_{j,k}}{1 + \left|\mathcal{L}_{j,k}^{\mathbf{v}_i}\right|} \right) \right], \tag{6.5}$$

In general terms, the goal is to maximize total links capacity while minimizing number of interferences taking into account power constraints. $p_{j,k}$ is the power assigned to $l_{j,k}$ and $\left|\mathcal{L}_{j,k}^{\mathbf{v}_i}\right|$ is the cardinality of the set of interfering links under assignment $\mathbf{v}_i$. The denominator of (6.5) reflects the fact that links which interfere with each other cannot be active simultaneously and, hence, have to time-share the channel to which they are assigned, decreasing the overall throughput (conversely, because link capacity increases monotonically with transmit power, increasing $p_{j,k}$ increases instantaneous link capacity). We avoid dealing explicitly with SINR by assuming that the medium access control (MAC) protocol prevents interference. The

reduction in average link capacity due to this no-interference policy is captured by the scaling factor $1/(1 + \left|\mathcal{L}_{j,k}^{\mathbf{v}_i}\right|)$.

The ideal scenario for the objective function (6.5) is to have enough available channels to assign different channels to all conflicting links, resulting in a conflict-free channel assignment. In this case, all nodes could transmit simultaneously using their maximum transmission power without causing any conflicts. Under this condition, the value of the localized objective function would be $\sum_{l_{j,k} \in \mathcal{L}_C^i} P_j$. This represents an upper limit on the objective function for any possible scenario. The next section presents the details of how we apply the localized island genetic algorithm to the problem formulated in this section.

6.4 Applying Localized Island Genetic Algorithm to Cross Layer Problem

In this section, we first introduce our Localized island Genetic Algorithm (LiGA). Then we present the details of how we implement and apply the LiGA to the distributed cross layer channel allocation-power control problem that we developed in the previous section. After that, we briefly discuss two of the tradeoffs that we faced during the design and implementation of the LiGA.

6.4.1 Localized Island Genetic Algorithm Formulation

Although the communication cost associated with the standard iGA is less than other parallel genetic algorithm techniques, it is still a problem when applied to large networks. The standard iGA needs global information to operate, increasing the communication cost and causing a scalability problem as the network size increases. For these reasons, we have modified the standard iGA individual structure and migration technique to use local information instead of global information. This solves the scalability problem.

For our Localized island Genetic Algorithm (LiGA), each node may have population individuals with different length and/or structure than other nodes in the network. Each node is now concerned about solving part of the problem, not the entire network problem. In LiGA, each node tries to find the best solution for nodes in its interference range by searching part of the search space. Disjoint parts of the partial solutions are then combined to form one global solution. These solution parts are guaranteed disjoint by allowing each node to apply the part of its final solution that is associated with its own configurations. This also reduces the communication cost required to exchange final solutions among network nodes. For the joint power and channel allocation problem, for example, each node allocates the channel and power level assignment of its final solution to its outgoing communication links. The other major modification in LiGA is the change of the migration policy. Instead of exchanging the entire structure of the best individual (or another member of the subpopulation), each node shares only the solution assignment part for its own parameters from its best individual and broadcasts this information to all nodes in its interference range instead of all network nodes as in the standard iGA. Each of the LiGA implementation steps is described in the following:

- ***LiGA individuals:*** In our implementation of the LiGA, each node in the network searches the space for the best channel-power allocation for all outgoing communication links that originate from any node with which it may interfere. Thus, we use $\mathbf{v}_i$, as defined in the previous section, as the structure of each of M individuals at node i, i.e. each individual is a series of tuples (channel, power level), one corresponding to each communication link that originate from any node with which node i may interfere. The required channel and power availability information can be collected either as an initial phase of the migration policy or through the periodic maintenance operation of the ad-hoc routing protocol. The initial population is generated randomly, with each link $l_{j,k} \in \mathcal{L}_i^C$ being assigned a random channel $h_{j,k}$ from its set of available channels $\mathcal{H}_{j,k}$ and a random power level $p_{j,k}$ from its set of available power levels $\mathcal{P}_{j,k}$. Note that because $\mathcal{P}_{j,k}$ is an infinite set, we assume that a finite number of power levels within

this set are available so that the optimization is over a finite space.

- ***LiGA fitness function:*** We use the function $f(\mathbf{v}_i)$ as defined in (6.5) as our fitness function. A better individual will give a higher $f(\mathbf{v}_i)$. At the beginning of every iteration of the LiGA, we compute the fitness of each individual in the population.

- ***Mating:*** We use the standard 1-point crossover as our mating form. The parents that are used in the crossover are chosen by tournament selection. In our tournament selection, each of the two parents is chosen as the best individual out of three randomly selected individuals from the parent population pool. Based on the fitness of each individual and a parameter called the *keep rate*, we eliminate the worst $\lfloor(1 - \textit{keep rate}) \cdot M)\rfloor$ individuals of the population and use the remaining individuals as the parent population to create the next generation. All next generations are then guaranteed feasible since the first generation is feasible.

- ***Mutation:*** We use a parameter called the *mutation rate* to define the amount of mutation performed at each iteration. After finishing the mating, mutation is performed by selecting an individual at random, excluding the one with the best fitness, and then selecting a link at random from that individual and replacing its assigned channel and power level by other values selected randomly from the set of available channels and power levels for this link. Performing the mutation this way, we guarantee feasible individuals for the next generation.

- ***Convergence:*** The current implementation terminates the algorithm after a fixed number of iterations. Although this may not be the best choice, this is currently enough for us to study the feasibility of using the LiGA to solve the cross layer channel allocation and power control problem. In the future, we plan to complete the cognitive node implementation presented in Chapter 3 by adding learning capabilities that control LiGA parameters and operation.

- ***Migration policy:*** For the LiGA implementation, we must also define the migration

policy that nodes will use to communicate and exchange information. The migration policy consists of three parts: what, when, and where. The first part defines what information each node needs to share with local nodes; the second part defines when migration will take place; and the third part defines where to send this information. In our implementation, we tried to use a simple communication-efficient migration policy in which each node shares only the channel and power level assignment of its outgoing links from its best individual, every *migration interval*. No synchronization is required for the algorithm to work, i.e. each node can perform migration at different timing, as long as the algorithm runs for enough time to stabilize or reach acceptable results. Using the multipoint relay mechanism [75], nodes broadcast this information after each migration interval to all nodes with which they may interfere. Each node receiving information from any other node will merge this information with all its individuals if the merged information will give better fitness. Otherwise it will merge it with only part of its population that is proportional to the fitness of the best individual before and after merging the migrated information. This forces each node to consider solutions of other nodes even if it will produce lower fitness individuals than are in the current generation.

6.4.2 Local Information vs. Global Information

In order to study the effect of working with partial knowledge (LiGA) and compare it to the performance of working with global information, we have also implemented the standard iGA, which uses global information, i.e. each node has full knowledge of the interference graph and the channel-power levels available for each link. All nodes in the standard iGA implementation have exactly the same individual structure, which contains channel-power level assignment for all communication links in the network. Moreover, each node works on finding a solution for the whole network, not only for its vicinity. All nodes apply a migration policy in which each node shares information with all other nodes in the network.

Clearly, the iGA with global information requires much more communication than the LiGA, because of the number of nodes that must receive each migrated individual (all network nodes) and the size of individuals (for each node to learn the whole network structure). Moreover, the standard iGA is not scalable and requires more computation power from each node. On the other hand, working with global information should increase the performance and accelerate the iGA convergence speed. The simulation results section shows the difference in performance of the two implementations.

6.4.3 Communication Cost vs. Stability

Another tradeoff that we have to make in the implementation of the LiGA is the communication cost versus performance and stability. The more information we exchange among nodes during migration, the better the performance, the faster the convergence, and the higher the stability. This is because as the amount of exchanged information increases, the uncertainty about other nodes' best individuals is reduced, and this speeds the algorithm convergence. We have chosen to save the communication cost by limiting migrated information for each node to the channel and power level assignment of its outgoing links from its highest fitness individual. At the same time, we have tuned the LiGA parameters to achieve better stability while not sacrificing much performance. The next section presents the details of simulation settings and provides simulation results for comparing the LiGA to the standard iGA.

6.5 Simulation Settings and Results

Table 6.1 summarizes the simulation settings for network and iGA parameters that are used to run our simulations. Network size ranges from 20 to 200 nodes and the network area is always adjusted to keep a constant node density, with the network area side being 2400 meters for a network of size $N = 200$. The ad-hoc network is generated by randomly placing network nodes uniformly over the network area. The maximum transmission range is set

Table 6.1: Network and Localized Island Genetic Algorithm Parameters' Setting

Network and Nodes Parameters' Setting	
Area	Square w/ constant node density
Number of Nodes (N)	20-200
Node density	34.7 $nodes/km^2$
Max Communication Range ($R^{T_{max}}$)	250 $meters$
Max Interference Range ($R^{I_{max}}$)	500 $meters$
Available Channels	10
Channels Lower Limit	3
Channels Upper Limit	8
Maximum Power (P)	20 dBm
Number of Power Levels ($\|Q\|$)	16
Path Loss Index	4
LiGA Parameters' Setting	
Population Size per island (M)	40
Number of Iterations	1000
Keep Rate	0.5
Mutation Rate	0.3
Migration Interval	20 Iterations

to 250 meters and maximum interference range is set to 500 meters. We set the maximum number of available channels in the sensed spectrum to 10 channels. We divide the network area into square subareas with the subarea side being 100 meters. We then randomly select a set of available channels at each subarea from the 10 available channels with an upper limit of 8 channels and a lower limit of 3 channels. Each node i is allocated a set of available channels $\mathcal{C}_i$ the same as its subarea.

Although our implementation is general enough to assign different values for the maximum power at each node in the network, we preferred to assign same value to simplify the analysis of the simulation results. We set the maximum power to 20 dBm, the number of power levels to 16 levels linearly spaced (in mW) ranging from no power (level 0) to maximum power

(level 15), and the path loss index to 4 for all nodes.

For the LiGA parameters, we set the population size M to 40 for all nodes, the *keep rate* to 0.5, the *mutation rate* to 0.3, the *migration interval* to 20 iterations, and the total number of iterations to 1000. Because the outcome of the LiGA is a random variable, the results for the same network may differ from one run to another. Thus, each result reported in this work is the average over at least 10 runs.

To provide some insight as to how large the search space is, we analyze the size of the problem search space. The upper bound of possible channel-power level assignment for each link in the network equals the product of the number of available channels, $|\mathcal{C}|$, times the number of available power levels, which we denote by $|\mathcal{Q}|$. Knowing that our network simulation settings result in an average number of communication links per node of about 4.5, the size of the search space, A, for a network of N nodes is given by:

$$|A| = (|\mathcal{C}| \cdot |\mathcal{Q}|)^{4.5N} \tag{6.6}$$

For our simulation settings and for a network of size 200, the search space size equals $|A| = (10 \times 16)^{4.5\times 200} = 160^{900} \approx 5.1 \times 10^{1983}$, which is impossible to search completely. Note that the upper bound for the number of available channels at each node for our simulation settings is 8 and the average number of available channels at each node is 5.5. The number of available channels for each link is even less because a communication link can only be assigned to a channel that is sensed free at its two end nodes. These factors decrease the size of the search space but it is still impossible to search it completely.

In the following sections, we provide simulation results for different network sizes, comparing the performance of the LiGA to the iGA with global information and then discussing the effect of network size on the LiGA convergence speed. We then show the effect of changing the number of available channels on the LiGA convergence speed. After that, we study the effect of changing the number of available power levels at each node on the LiGA performance. Finally, we study the effect of different nodes' non-cooperation behaviors on the performance

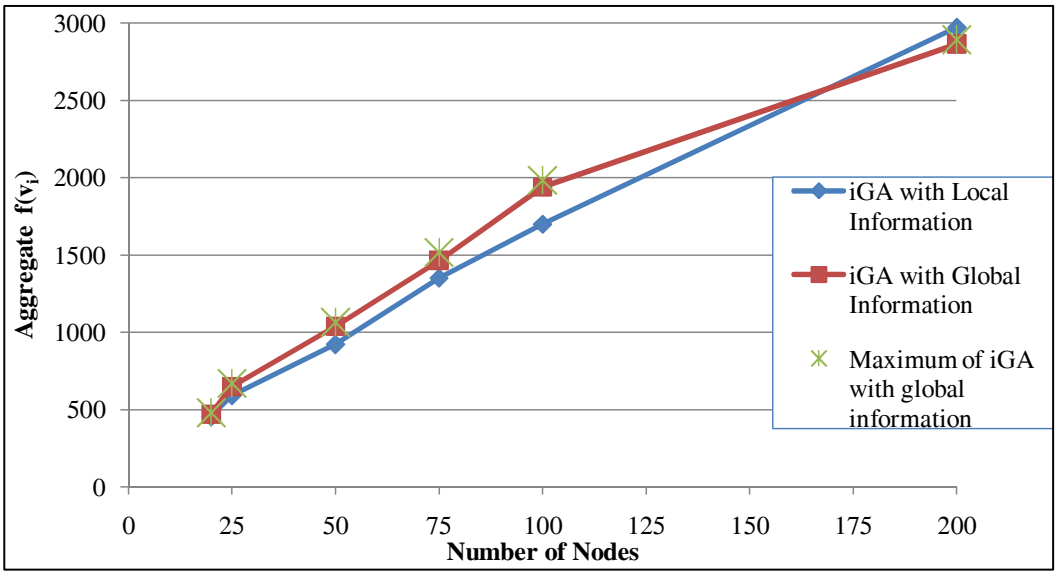

Figure 6.1: Performance of Localized iGA Vs standard iGA with global information ©2008 IEEE. Reprinted, with permission, from ElNainay *et al.*, "Channel allocation & power control for dynamic spectrum cognitive networks using a localized island genetic algorithm," in *Proc. IEEE DySPAN'08*, (Chicago, IL), pp. 1–5, October 2008

of the LiGA.

6.5.1 Effect of Localization

In this section, we compare the performance of the LiGA to the standard iGA by running simulations of both versions over the same network for network sizes ranging from 20 to 200 nodes. Fig. 6.1 shows the overall network aggregate throughput, in terms of $f(\mathbf{v}_i)$ summed over all nodes, averaged over 10 runs of the LiGA, the standard iGA, and also shows the maximum of all runs of the standard iGA.

For all simulations except for the network of size 200, the standard iGA with global information gives better results than the LiGA. This was predictable because each node is working with full knowledge and tries to find a solution for the whole network rather than working on a piece of the big problem. Interestingly, this does not hold for 200 node networks. For the 200 node networks, the LiGA gives slightly better results than the standard iGA. Our

explanation for this phenomenon is that because the problem search space for the 200 node network is so big, it becomes difficult for the network nodes to work on a global solution and converge to a good result in just 1000 iterations. On the other hand, each node in the localized version concentrates on finding a good solution for a part of the big problem and then merges this with other nodes to form a global solution and this gives a better final result. We have supported this explanation by running the standard iGA for 2000 iterations instead of 1000 iterations and the result was better than the LiGA for 1000 iterations (the aggregate throughput was about 3300).

6.5.2 Effect of Network Size on Convergence Speed

In this section, we discuss briefly the effect of the network size on the LiGA convergence speed. We traced the intermediate results of the LiGA for different network sizes and recorded the number of iterations required for each run to reach to 95% of its maximum output and then we averaged this value over 10 runs.

Fig. 6.2 shows the average number of iterations to reach 95% of the maximum for different network sizes, with all other parameters fixed. The figure shows that the average number of iterations increases, the convergence speed of the LiGA decreases, as the network size increases from 20 nodes up to 75 nodes. This is reasonable because as the network size increases the search space increases and it becomes more difficult to reach to an optimal or suboptimal solution. However, the curve then stabilizes, around the value 550 iterations, as the network size increases over 75 nodes. This proves the scalability of our LiGA with the network size without increasing the time complexity.

6.5.3 Effect of Number of Channels on Convergence Speed

In these simulations, we have studied the effect of the number of available channels on the LiGA performance by applying the LiGA to networks of size 50 nodes but with the number

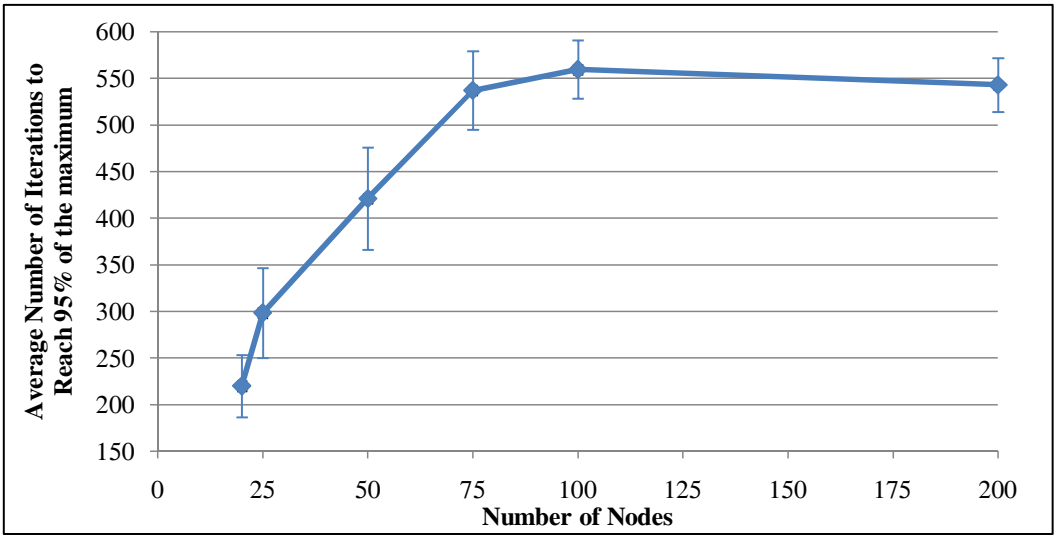

Figure 6.2: Effect of Network Size on Convergence Speed of the island Genetic Algorithm (along with 95% confidence intervals)

of available channels ranging from 5 to 50.

Fig. 6.3 shows the effect of changing the number of available channels on the LiGA performance. The figure shows that as the number of available channels increases, the overall network aggregate throughput increases. For networks where the number of available channels is 40 or more, the network aggregate throughput appears to stabilize to almost the same value, around 3700. This matches the average number of interference links per node for our simulated network density which was also about 40 interference links per node.

6.5.4 Effect of Number of Power Levels on Performance

Allowing for power control gives each node the option to adjust its transmission power to compromise between transmission rate/range and the interference it causes to other nodes. This should result in better overall network performance by making more channel assignments feasible. In this section, we study the effect of changing the number of available power levels on the LiGA performance and convergence speed. All simulations run the LiGA on the same 75 node network but with the number power levels, $|Q|$, ranging from 1 to 16.

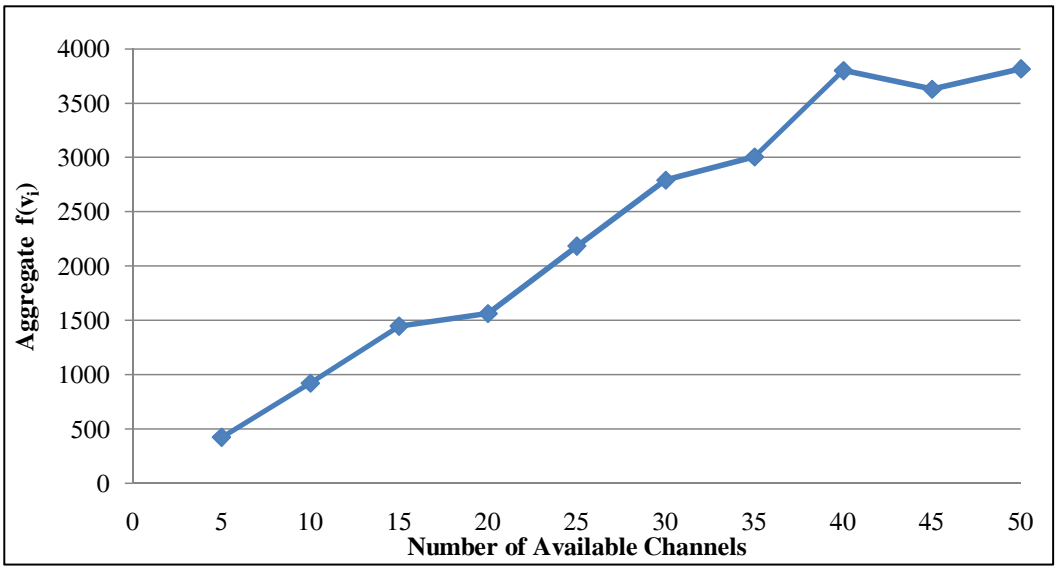

Figure 6.3: Effect of the Number of Available Channels on LiGA Performance for 50 Nodes Networks

Fig. 6.4 shows the effect of changing the number of power levels on the LiGA performance. The figure shows that when the number of power levels is low, 1 (no power control) and 2 (very limited), the performance is worse than when the number of power levels increases to 4. The figure also shows that as the number of power levels increases beyond 4, there is little improvement in the performance. However, our intuition is that the best number of power levels depends on the network size and this should be set by a learning mechanism inside the cognitive controller of the CN that we intend to add in the future.

We have also traced the convergence speed for each number of power levels in terms of the average number of iterations required to reach 95% of the maximum value of each run. Fig. 6.5 shows the effect of the number of power levels on the LiGA convergence speed. The figure shows that the convergence for a low number of power levels is faster than with a high number of power levels because the search space (number of possible assignments) is relatively small. Therefore, the LiGA reaches the best possible assignment with this limited number of power levels faster. The figure also shows that the increase of the average number of iterations is slower as the number of available power levels increases over 6 compared to

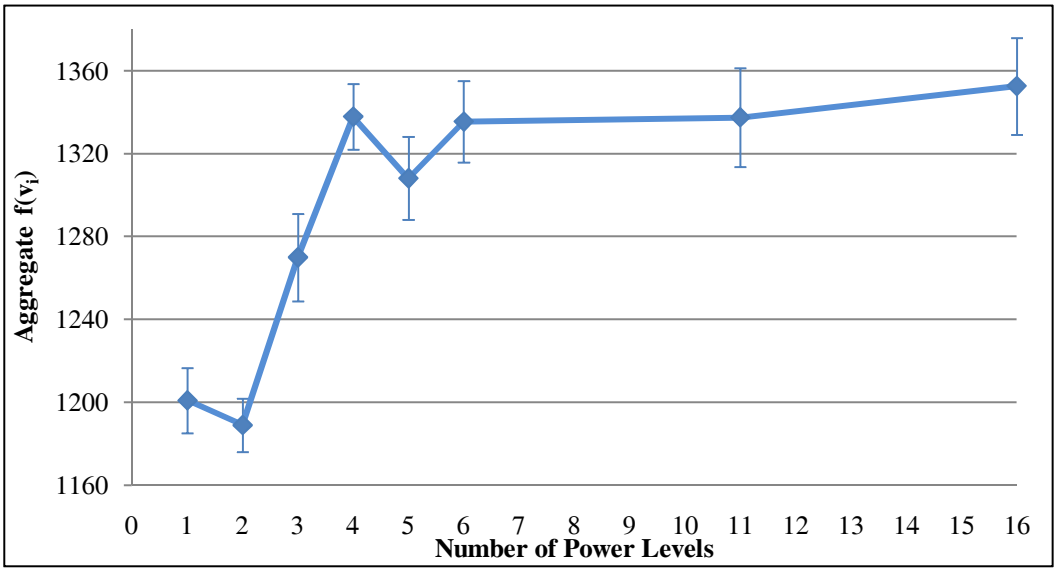

Figure 6.4: Effect of the Number of Power Levels on LiGA Performance for a 75 Nodes Network (along with 95% confidence intervals) ©2008 IEEE. Reprinted, with permission, from ElNainay *et al.*, "Channel allocation & power control for dynamic spectrum cognitive networks using a localized island genetic algorithm," in *Proc. IEEE DySPAN'08*, (Chicago, IL), pp. 1–5, October 2008

the increase from 2 up to 4. Notice from Fig. 6.4 that there is little improvement in the LiGA performance as the number of power levels increases beyond 4.

6.5.5 Effect of Non-Cooperation

All previous simulations assume the willingness of network nodes for cooperation in order to achieve better network performance. In this section, we study the effect of two practical problems that partially invalidate this assumption. The first problem is the existence of non-cognitive nodes, which are not capable of such cooperation. Even if we assume that cognitive nodes will dominate at some point in the future, there will be a transition period (of several years) during which cognitive nodes have to work with non-cognitive nodes, i.e. in a mixed network of cognitive and non-cognitive nodes. Some efforts have been made previously to study the impact of selfish behaviors and selfish algorithms on the CN performance mostly in the context of the topology control problem [88] (more information can be found in [89]).

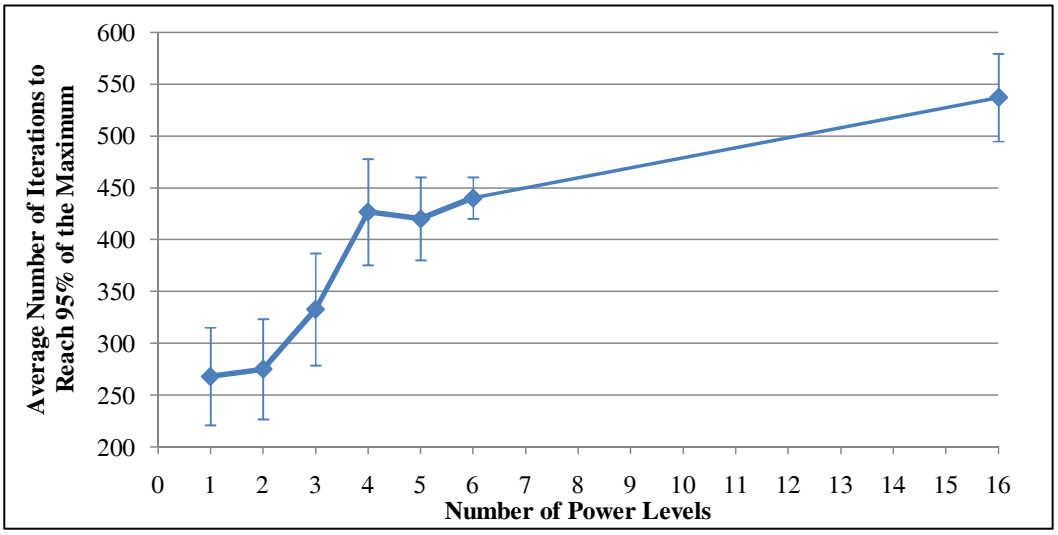

Figure 6.5: Effect of the Number of Power Levels on LiGA Convergence Speed for a 75 Nodes Network (along with 95% confidence intervals)

The second problem is the possible existence of selfish nodes among cognitive nodes. As defined in [90], a selfish node takes advantage of the network, namely the other nodes, for its communication and does not cooperate with others, making network resources unavailable for legitimate nodes. This kind of node saves its own resources, battery power and bandwidth, while using others resulting in network unfairness and performance degradation.

In the next two subsections, we study the impact of the two aforementioned practical problems through two simulation studies. In the first simulations set, mixed networks of cognitive and non-cognitive nodes are generated with different percentages for the number of non-cognitive nodes to the number of cognitive nodes and the overall network performance is monitored and analyzed. In the second simulations set, cognitive network of selfish and unselfish nodes is generated and the impact of different selfish behaviors is studied.

6.5.5.1 Effect of Non-Cognitive Nodes

In order to study the effect of the existence of non-cognitive nodes on the performance of a cognitive network (CN), we have made the following assumptions about non-cognitive and

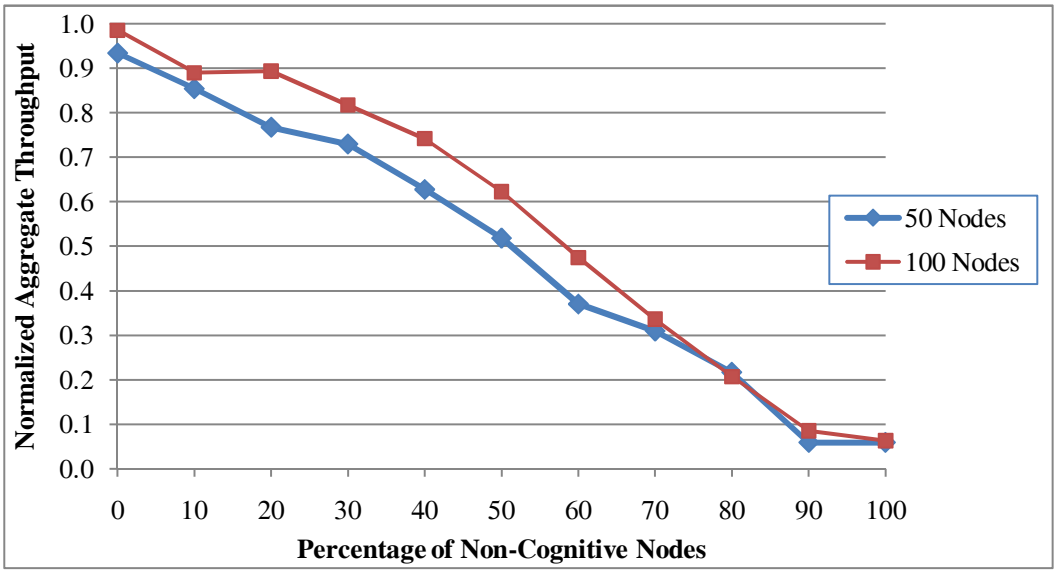

Figure 6.6: Effect of Mixing Cognitive and Non-Cognitive Nodes on Network Performance [Mustafa Y. ElNainay and A. B. MacKenzie, "Effect of Non-Cooperation on Dynamic Spectrum Cognitive Networks," *to appear in Proc. of IWCMC'09*, June 2009] ©2009 ACM, Inc. Reprinted by permission.

cognitive nodes:

- We assume that channels used by non-cognitive nodes are available to all cognitive nodes (could be in the open ISM band).
- All non-cognitive nodes use their maximum power level to communicate with other nodes.
- We assume that cognitive nodes can collect topology information about cognitive and non-cognitive nodes, possibly through the routing protocol.

We have run simulations on networks of size 50 and 100 nodes; with the percentage of non-cognitive nodes ranging from 0% (i.e. all nodes are cognitive) up to 100% (i.e. all nodes are non-cognitive). Fig. 6.6 shows the effect of mixing cognitive and non-cognitive nodes on the overall normalized aggregate throughput of the network.

Fig. 6.6 shows that independent of the network size, increasing the percentage of the non-cognitive nodes decreases the overall network throughput. The figure also suggests that this decrease is roughly linear in the percentage of the non-cognitive nodes. Moreover, the all non-cognitive nodes network performance is more than 90% lower than the all cognitive nodes network performance.

6.5.5.2 Effect of Selfishness

In most cases, CNs assume some level of knowledge sharing in order to achieve network-wide efficiency. This gives importance to studying the effect of selfish and malicious behaviors on the CN performance. This ranges from refusing to share information in order to save power, working on selfish goals instead of the network-wide goal to save computational power, to sharing false information in order to ruin the entire network and achieve selfish goals.

In this section, we have studied the effect of selfish behaviors on the performance of CN. We have implemented two different scenarios for the selfish behavior with networks. Both scenarios consist only of cognitive nodes. For the first scenario, the selfish nodes work on their own goals instead of the network-wide goals, i.e. performance of candidate solutions is judged relative to node-centric goals and does not consider goals of other nodes. Moreover, selfish nodes do not share information with other nodes probably to save their own power, and accept and merge other nodes' information/solution only if this supports their own goals. In the second scenario, selfish nodes are assumed to share information with other nodes but still do not consider other nodes' solutions except for a better own goals. In this scenario, selfish nodes are willing to share their information hoping that unselfish nodes will try to avoid conflicts with them and yet achieve a better own performance. We have run simulations on different network sizes and with the percentage of selfish nodes ranging from 0% (i.e. no selfish nodes) up to 100% (i.e. all nodes are selfish).

Fig. 6.7 shows the effect of the first scenario described on the overall network throughput normalized to the maximum achieved value in all runs. Fig. 6.7 shows that as the percentage

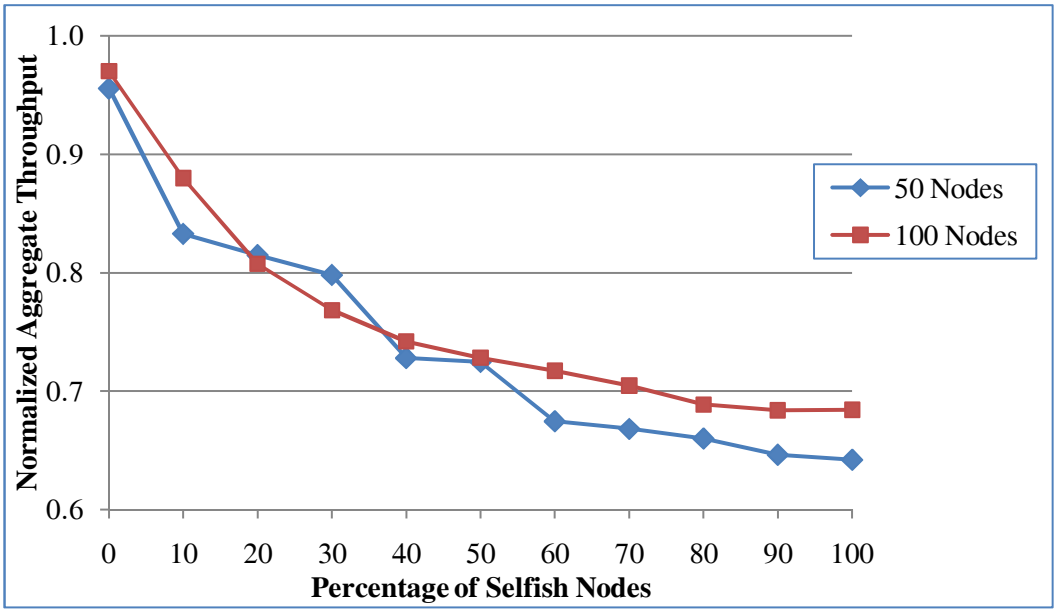

Figure 6.7: Effect of Selfish Behavior on Network Performance [Mustafa Y. ElNainay and A. B. MacKenzie, "Effect of Non-Cooperation on Dynamic Spectrum Cognitive Networks," *to appear in Proc. of IWCMC'09*, June 2009] ©2009 ACM, Inc. Reprinted by permission.

of the number of selfish nodes increases the network performance decreases. The network performance drops from about 97% to about 65% when all nodes shift from being non-selfish to being selfish. It is also obvious from comparing Fig. 6.6 and Fig. 6.7 that the all cognitive nodes network performance even with selfish nodes is better than the all non-cognitive nodes network performance. This highlights the importance of the cognitive radio and cognitive network research.

We have also applied the second scenario, in which selfish nodes share their information, to the same set of networks and Fig. 6.8 compares the performance of the first and second scenarios normalized to the maximum outcome of all simulations. The results are for two different network sizes, 50 and 100, and for percentage of the number of selfish nodes ranging from 10% up to 100%. The 0%, i.e. no selfish nodes, is excluded since there is no difference in both scenarios at this percentage.

Fig. 6.8 shows that as selfish nodes share their information, the overall network performance

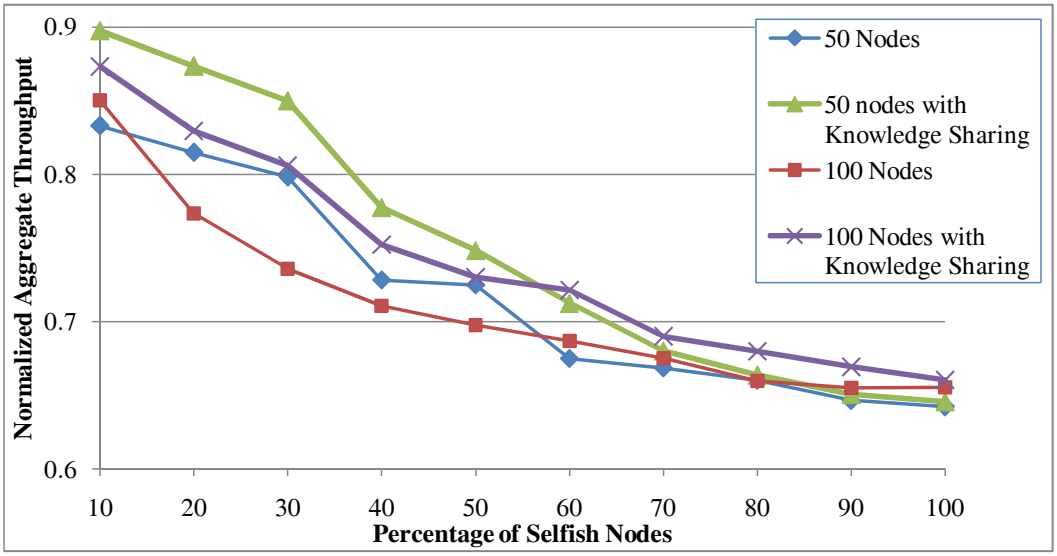

Figure 6.8: Effect of Information Sharing of Selfish Nodes on Network Performance [Mustafa Y. ElNainay and A. B. MacKenzie, "Effect of Non-Cooperation on Dynamic Spectrum Cognitive Networks," *to appear in Proc. of IWCMC'09*, June 2009] ©2009 ACM, Inc. Reprinted by permission.

is better for the same network settings. For example, for the 50 node network, the normalized aggregate throughput for the first scenario, with no knowledge sharing, gives 0.81 while the second scenario, with knowledge sharing, gives 0.87 at 20% of selfish nodes. Thus, this fact can work as an incentive for the selfish nodes to at least share their own information to help other nodes avoiding conflicts with them and hence improve their own performance. Moreover, the figure shows that as the percentage of the number of selfish nodes increases, the performance gap between the two scenarios diminishes. The reason is as the percentage of selfish nodes increases, the usefulness of knowledge sharing decreases since selfish nodes do not get use of this knowledge (except when this knowledge results in a better self performance).

6.6 Summary and Contributions

Power control has a big influence on the performance of wireless networks in general and cognitive networks in particular because it affects both the efficiency and feasibility of the channel allocation, and hence the spectrum utilization. Power control gives each node the option to adjust its transmission power to compromise between transmission rate, transmission range, and interference caused to other nodes, resulting in better overall network performance by making more channel assignments feasible.

In this chapter, we have formulated a model of the cross layer channel allocation and power control problem in the context of an ad-hoc cognitive radio network in a dynamic spectrum environment. We have implemented a localized variation of the island genetic algorithm, LiGA, which uses local information instead of global information. We have then applied the LiGA to the cross layer channel allocation-power control problem. For comparison purposes, we have also implemented the standard iGA that uses global information. One of the main advantages of the LiGA over the iGA with global information is its scalability with the network size without increasing the time complexity. The LiGA gives very promising and comparable results to the standard iGA. The LiGA can sometimes even give better results in the same number of iterations for large networks because the search space becomes so big for the standard iGA to get good results in low number of iterations. We have also provided simulation results showing the effect of power control on the channel assignment efficiency and the LiGA performance. Moreover, the effect of various network and genetic algorithm parameters on the performance and convergence speed of the LiGA has been studied to inspect the robustness of the LiGA against these parameters. This also contributes toward our future work on incorporating reasoning with learning algorithms as we will introduce in Chapter 8

Furthermore, we have studied the impact of two forms of non-cooperation that cognitive networks may face. The first is the existence of non-cognitive nodes and the fact that cognitive nodes have to be able to interoperate with non-cognitive nodes. The second problem

is the possible existence of selfish nodes among cognitive nodes. We have studied both non-cooperation forms through different simulation scenarios. Simulation results are provided that reflect the performance degradation resulting from each case. The results show that the all cognitive nodes network even with selfish nodes achieves better performance than the all non-cognitive nodes network. The second study suggests the importance of an enforcement mechanism to prevent cheating or a reputation mechanism to avoid it.

The LiGA is a promising technique to solve the standard iGA scalability problem when applied to large networks. However, not every communication and networking problem can be solved with local information. Thus, we have developed the K-hop iGA which is a generalized version of the LiGA that suits a larger class of communication and networking problems with controllable cooperation and migration scope. The next chapter presents the K-hop iGA and applies it to the flow routing problem.

The work presented in this chapter has resulted in the following publications:

1. Mustafa Y. ElNainay, Daniel H. Friend, and Allen B. MacKenzie, "Channel Allocation & Power Control for Dynamic Spectrum Cognitive Networks using a Localized Island Genetic Algorithm," in *Proc. IEEE New Frontiers in Dynamic Spectrum Access Networks (DySPAN'08)*, (Chicago, IL), pp. 1–5, 14–17 Oct. 2008.

2. Mustafa Y. ElNainay and Allen B. MacKenzie, "Effect of Non-Cooperation on Dynamic Spectrum Cognitive Networks," in *Proc. the 5th International Wireless Communications and Mobile Computing Conference (IWCMC'09)*, (Leipzig, Germany), 21–24 June 2009.

3. Mustafa Y. ElNainay, Daniel H. Friend, and Allen B. MacKenzie, "Reasoning in Cultural Algorithm-based Cognitive Network," *IEEE Transactions on Wireless Communications*, under review.

Chapter 7

Joint Flow Routing, Channel Allocation and Power Control using K-hop Island Genetic Algorithm [1]

In Chapter 6, a localized variation of the island genetic algorithm (iGA) is presented, namely the LiGA. The LiGA uses only local information to solve problems which reduces the communication cost but at the same time limits its applications. In this chapter, we introduce a generalized version of the LiGA, namely K-hop iGA, and apply it to the flow routing problem considering both channel allocation and power control in the context of ad-hoc cognitive network (CN). The K-hop iGA implementation is flexible and general enough to solve more communication and networking problems with its distributed computation and its controllable cooperation range. We present a distributed formulation for the cross layer flow routing, channel allocation, and power control problem and then apply the K-hop iGA to the cross layer formulation. Through simulations, we evaluate the performance of the K-hop iGA by comparing it to a theoretical technique.

The Chapter is organized as follows: Section 7.1 introduces the flow routing problem and

[1] This chapter is based on the work in [91].

discusses its new challenges with multi-hop CNs. Section 7.2 surveys related works on flow routing in the context of CNs. In Section 7.3, we present the K-hop iGA and discuss some of its design issues and tradeoffs. The cross layer problem formulation is presented in Section 7.4. Section 7.5 describes the details for applying the K-hop iGA to the flow routing problem. The simulation results are provided in Section 7.6.

7.1 Flow Routing Problem

The flow routing problem in multi-hop ad-hoc network is concerned with finding the routes for network flows that satisfy flow rate requirements and wireless links' capacity constraints. Given sources and destinations of network flow sessions along with their requested flow rates, the flow routing algorithm finds the paths to relay the data packets from sources to destinations considering links' capacity constraints. This is a networking problem that is tightly coupled with the physical and link layers. Thus, most of the flow routing problem solutions include either cross layer optimization or information sharing with the underlying physical and link layers.

The dynamic spectrum cognitive network (CN) imposes new challenges to flow routing and to wireless routing protocols in general. In traditional multi-hop networks, each node has the same number and set of available channels. Thus, there is no connectivity problem imposed by spectrum availability as all nodes have same channels in common. In dynamic spectrum CN, the set of available channels is location and time dependent. Each node may have different number and different set of available channels. Therefore, flow routing and channel allocation should be jointly considered to decide both route path and link channels. Moreover, power control is of great importance to make more channel assignments feasible and hence increase the network performance as shown in Chapter 6.

In this chapter, we introduce a new variation of the island genetic algorithm (iGA), namely the K-hop iGA, and apply it to the flow routing problem considering both channel alloca-

tion and power control in the context of dynamic spectrum multi-hop CN. We formulate a distributed cross layer model for the flow routing, channel allocation, and power control for an ad-hoc CN. We then apply the K-hop iGA implementation to the cross layer problem and compare its performance to the theoretical branch and bound technique presented by Shi and Hou in [87]. The next section surveys related work on flow routing in the CN context.

7.2 Related Work

In this section, we review related work on cross layer optimization that includes flow routing, or routing in general, within the optimization problem. We limit our review to the work provided in the context of cognitive radio networks (CRNs) and cognitive networks (CNs).

In [87], Shi and Hou develop a mathematical formulation for the cross layer power control, scheduling, and flow routing problem to support a predefined set of user communication sessions in CRN. Then they apply a solution procedure based on the branch-and-bound technique and convex hull relaxation on their model. Using this solution procedure, they can provide a solution with a fitness within a factor of (1- ϵ) of the optimal solution, where ϵ reflects the accuracy required. Through simulations, they show the efficacy of their solution procedure and demonstrate the positive impact of power control on channel scheduling feasibility and bandwidth efficiency. In [92], Shi *et al.* change the interference model to the physical model instead of the protocol interference model presented in [87] but apply the same centralized branch and bound solution procedure. Although they provide an excellent mathematical model that includes the power control as a part of the optimization space and hence can be used in other wireless communication problems, this work is theoretical and the complexity of the proposed solution is high. Hence the solution cannot be practically applied in a real network.

Using a similar formulation as the previous work but considering bidirectional links instead of unidirectional links, Ma and Tsang [93] present a cross layer model for spectrum sharing

and flow routing for CRNs. For bidirectional links, interference from both the sender and the receiver needs to be considered, rather than from only the sender as in unidirectional link modeling. The cross layer problem is modeled as a mixed integer linear program and solved using commercial linear programming software. Through simulations, the authors show that interference-free and fair routing solutions can be guaranteed. However, the authors assume the existence of a centralized server and solve the problem in a centralized, offline manner which is not practical especially in dynamic network environments and requires high communication cost.

In [94] and [95], Cheng *et al.* jointly consider on-demand routing and channel allocation. An analytical model that takes into account delays introduced by the multi-frequency assignment and frequency switching in dynamic spectrum environment is proposed. An on-demand protocol for routing and channel allocation that considers both path and node delays is then proposed. Each network node is assumed to be equipped with two interfaces, a traditional 802.11 interface and a software defined radio, and the availability of a common control channel among all nodes is assumed. In [96], Ma *et al.* present similar ideas using a single transceiver at each node with no common control channel. In [97], Yang *et al.* augment the work presented in [94] and [95] with a local coordination scheme to support flow redirection at heavy loaded intermediate relaying nodes. However, none of these solutions consider power control as part of the optimization problem.

Our work is different from previous work in solving the flow routing problem using the distributed island genetic algorithm (iGA) and considering power control as part of the optimization problem. To the best of our knowledge, this is the first work to use the iGA to solve the flow routing problem. Moreover, we propose the K-hop iGA as a generalized variation of the localized iGA (LiGA) presented in Chapter 6. The K-hop iGA allows for adjustable tradeoff between solution efficiency and communication cost. We also note that while previous solutions are dedicated to solving the flow routing problem in specific, we try to prove the applicability of using the iGA as distributed reasoning algorithm for CNs with acceptable performance. The next section presents the concept of the K-hop iGA.

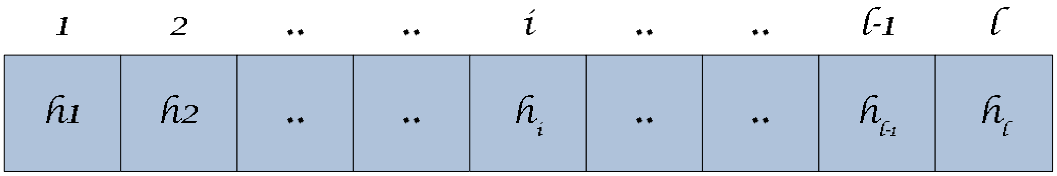

Figure 7.1: Individual Structure for 1-hop iGA Channel Allocator

7.3 K-hop Island Genetic Algorithm

In this section, we introduce the K-hop island genetic algorithm (K-hop iGA) for distributed reasoning in CNs and follow this with a discussion of its potential benefits to communication and networking problems in general.

7.3.1 K-hop Island Genetic Algorithm Concept

Some communication and networking problems cannot be solved using only local information. At the same time, solutions using global information have high communication cost and suffer from scalability problem. Something in the middle between these two options may provide the best tradeoff for the solution algorithm to reach acceptable results. This is the basic idea behind the K-hop iGA which can be considered as a generalized version of the LiGA we introduced in Chapter 6. With the K-hop iGA, network nodes can use different localized scopes for different problems. This ranges from zero-hop to M-hop, where M is the network diameter. As the value of K increases, the communication cost needed for the K-hop iGA to operate also increases. With K-hop iGA, each node has a different individual structure that depends on the problem and the value of K. For example, if network nodes are trying to solve the channel allocation problem with 1-hop iGA, the individual structure at one network node will look like that shown in Fig. 7.1, where the length of the individual, l, equals the number of outgoing communication links in the 1-hop range of this node with each gene corresponding to channel allocated, h_i, to one communication link.

With a complete CN architecture like the one we proposed in Chapter 3, a cognitive engine

can assess the performance of the 1-hop iGA using some performance metrics. If the results of 1-hop iGA for a specific problem are not acceptable, the results may be improved by increasing the cooperation range to 2-hops instead of 1-hop.

7.3.2 Zero-hop and M-hop Island Genetic Algorithms

The two extreme cases of the K-hop iGA are when K equals zero and when K is greater than or equal to M (the network diameter). With K equal to zero, each node in the network works on solving the problem without exchanging any information with any other node. In this case, the K-hop iGA converges to a local node-centric algorithm in which each node tries to optimize its own objective with no migration among nodes. This implementation can be useful for optimizations that need no information about other nodes. The zero-hop iGA can also be used in case of bandwidth or other network resource shortages. The zero-hop iGA decreases the communication cost of the algorithm by eliminating migration, but this comes with the cost of limiting its applications and/or the efficiency of the final results.

The other extreme case is the M-hop iGA, where M equals the network diameter. In this case, the M-hop iGA is exactly the same as the standard iGA. Every node needs to collect information and cooperate with all other nodes in the network. This increases the communication cost but is necessary for some problems that need global information to be solved. The M-hop iGA can be used when high efficiency of the final solution is of concern. The general rule for cognitive nodes in a CN should probably be to start with limited cooperation (knowledge) using zero-hop or 1-hop iGA unless the problem under concern is known to require global information.

7.3.3 Advantages of K-hop Island Genetic Algorithm

The K-hop iGA offers many advantages as a reasoning algorithm for CNs. Some are inherited from genetic algorithms (GAs) and island genetic algorithms (iGAs), while others are unique

to the idea of the K-hop iGA. Following are some of these unique advantages:

- With the K-hop iGA, the cooperation range of each node can be adjusted to the requirements of the problem and to network resource availability.
- Using the zero-hop iGA, we have a local algorithm that can work quickly with no communication cost, while with setting K to the network diameter, the K-hop iGA converges to the standard iGA to take the advantage of global information.
- The idea of the K-hop iGA, with the inherited distributed computation characteristic from the iGA, together with a good implementation within a CN framework, gives this algorithm its promise as a candidate solution for a wide variety of communication and networking problems.

In the next section, we present a distributed model of the joint flow routing, channel allocation, and power control problem as a candidate application of the K-hop iGA. Later, we provide the details of how we apply the K-hop iGA to this cross layer problem.

7.4 System Model

We first present our network model and then introduce the power control, channel allocation, and flow routing mathematical models separately. Note that these models are similar to the models presented in Chapter 6 but repeated here for completeness. After that, we show how they are combined to represent the complete cross layer problem as a distributed model.

7.4.1 Network Modeling

Consider an ad hoc cognitive network consisting of a set of nodes, $\mathcal{N}$. Each node $i \in \mathcal{N}$ is able to transmit and receive simultaneously on different channels. Denote $l_{i,j}$ as the directed

communication link for which node i is the transmitting node and node j is the receiving node. Using directed rather than undirected communication links allows for full duplex transmissions by assigning different channels to links $l_{i,j}$ and $l_{j,i}$.

7.4.2 Power Control Modeling

In this work, we still follow the protocol interference model in which interference is a binary condition [67], i.e. either a pair of links interfere with each other or they do not. Under this interference model, consider the general power control case where each node i can transmit at any transmission power up to node-dependent maximum transmission power, P_i that may depend on node capabilities. Define $g_{i,j}$ as the channel gain from a transmitting node i to a receiving node j,

$$g_{i,j} = d_{i,j}^{-\gamma}, \tag{7.1}$$

where $d_{i,j}$ is the distance from node i to node j, and γ is the path loss index. Following the same derivation as in [87], we assume that a transmission from node i to node j is considered successful only if the received power at node j exceeds a certain threshold, say α. Define $p_{i,j}$ as the transmission power assigned to link $l_{i,j}$, then for a successful reception we have

$$p_{i,j} \cdot g_{i,j} \geq \alpha, \tag{7.2}$$

Define $R_i^T(p_{i,j})$ as the transmission range for node i when transmitting with power $p_{i,j}$. Then, using (7.1) and (7.2), $R_i^T(p_{i,j})$ can be computed as follows:

$$R_i^T(p_{i,j}) = \left(\frac{p_{i,j}}{\alpha}\right)^{\frac{1}{\gamma}}, \tag{7.3}$$

Similarly, we assume that the interference resulting from node i when transmitting with $p_{i,j}$ is non-negligible only if it exceeds a certain threshold, say β. Then, similar to $R_i^T(p_{i,j})$, we can compute $R_i^I(p_{i,j})$, the interference range for node i when transmitting with power $p_{i,j}$,

as follows:

$$R_i^I(p_{i,j}) = \left(\frac{p_{i,j}}{\beta}\right)^{\frac{1}{\gamma}}, \tag{7.4}$$

Denote $R_i^{T_{max}}$ the maximum transmission range for node i corresponding to using the maximum transmission power P_i (R here stands for Range). Similarly, denote $R_i^{I_{max}}$ the maximum interference range for node i corresponding to using the maximum transmission power P_i. We can then compute $R_i^{T_{max}}$ and $R_i^{I_{max}}$ by substituting P_i for $p_{i,j}$ in equations (7.3) and (7.4) respectively,

$$R_i^{T_{max}} = \left(\frac{P_i}{\alpha}\right)^{\frac{1}{\gamma}}, \tag{7.5}$$

and

$$R_i^{I_{max}} = \left(\frac{P_i}{\beta}\right)^{\frac{1}{\gamma}}. \tag{7.6}$$

Define $p_{i,j}^{min}$ as the minimum transmission power required for node i to transmit successfully to node j. Then, from equations (7.1), (7.2) and by substituting for α from equation (7.5) we have:

$$p_{i,j}^{min} = \left(\frac{d_{i,j}}{R_i^{T_{max}}}\right)^{\gamma} \cdot P_i, \tag{7.7}$$

Thus, for a successful transmission from node i to node j, we have:

$$p_{i,j}^{min} \leq p_{i,j} \leq P_i, \tag{7.8}$$

Then, define $\mathcal{P}_{i,j}$ as the set of allowable transmission power values for link $l_{i,j}$, i.e., $\mathcal{P}_{i,j} = \{p : p_{i,j}^{min} \leq p \leq P_i\}$. We then define $\mathbf{p} \in \times_{l_{i,j} \in \mathcal{L}_C} \mathcal{P}_{i,j}$ as the transmission power assignment vector, where $\mathcal{L}_C$ is the set of all directed communication links.

7.4.3 Channel Allocation Modeling

Considering only frequency allocation, we assume that the spectrum is divided into a set of non-overlapping channels, $\mathcal{C}$, and that each node i has detected (perhaps jointly with other nodes) a set of channels available for local use, $\mathcal{C}_i \subseteq \mathcal{C}$. Based on these node-centric sets,

each link $l_{i,j}$ has a set of valid channel assignments, $\mathcal{H}_{i,j} = \mathcal{C}_i \cap \mathcal{C}_j$. Define $\mathbf{h} \in \times_{l_{i,j} \in \mathcal{L}_C} \mathcal{H}_{i,j}$ as the channel assignment vector, where $\times_{l_{i,j} \in \mathcal{L}_C} \mathcal{H}_{i,j}$ is the Cartesian product of the available channel sets for all communication links. Denote $h_{i,j}$ as the element of $\mathbf{h}$ that is the channel assigned to link $l_{i,j}$ and denote $w(h_{i,j})$ as the bandwidth of $h_{i,j}$.

In our solution using the K-hop iGA, we search for the best possible solution in a time limit. This does not guarantee a conflict free solution, i.e. two links in the same interference range may still be assigned to the same channel in the final solution. We assume that conflicting links will use a sensing/time-sharing medium access control (MAC) mechanism to share the assigned channel. The advantage of this solution is that we guarantee finding an applicable solution even when there is no feasible conflict-free solution based on the current network state. This is also why we do not add scheduling constraints to our channel allocation modeling.

7.4.4 Flow Routing Modeling

Assume a link-disjoint multipath routing protocol that works independently of our joint problem to maintain a routing table of all different possible next-hops to every (or on demand) potential destination at each node in the network. Multipath routing techniques are superior over single-path routing protocols in ad-hoc networks as they reduce the routing overhead [98]. Using multipath routing algorithms also increases network reliability and capacity by breaking one flow into multiple independent paths that can be active in parallel. Assuming the existence of a routing algorithm instead of exploring all possible paths is more realistic. This also decreases the search space. The choice of link-disjoint instead of node-disjoint is because the node-disjoint is a subset of the link-disjoint and we are trying to consider the more general case. In a dynamic spectrum environment, primary users' behavior pattern is the main factor that determines the channels availability at each node. Thus, losing a channel because of a primary user appearance may lead to a link failure (at least for some time till this link is assigned to another channel) and does not necessarily

lead to a node failure. In this case, a link-disjoint multipath routing may be sufficient and a node-disjoint may be overly restrictive. On the other hand, if the node mobility is high, a node-disjoint multipath routing protocol is preferable [99]. Li and Cuthbert [99] provide a good stability analysis of both link-disjoint and node-disjoint multipath routing. Unfortunately, they consider the mobility as the main reason for instability, and thus they favor node-disjoint multipath routing. We prefer to use link-disjoint protocol to take advantage of every possible unique route in the network. In order to simplify the routing table at each node, we assume a multipath routing protocol that provides only the next hops to each destination instead of the entire paths.

Adopting the same flow model as in [100], consider a set $\mathcal{E}$ of different active user communication sessions among network nodes. Denote $s(e)$ and $d(e)$ the source and destination of session $e \in \mathcal{E}$ respectively. Denote $f(e)$ the flow rate requirement for session e. Denote $\mathcal{N}_i(e) \subset \mathcal{N}$ the set of different possible next hops available at node i for routing the data flow of session e toward its destination $d(e)$. Then denote $\mathcal{N}^i(e) \subset \mathcal{N}$ the set of different possible previous hops available at node i back to the source $s(e)$ of session e. Note that $\mathcal{N}_i(e)$ and $\mathcal{N}^i(e)$ can be easily constructed using the information we have assumed to be available in the routing table of node i.

Based on the required rate, we may have to split the flow over multiple paths. Thus, denote $f_{i,j}(e)$ the flow rate of session e that is assigned to be routed from node i to node j. Then for flow balance, we have:

$$\sum_{j \in \mathcal{N}_{s(e)}(e)} f_{s(e),j}(e) = f(e), \tag{7.9}$$

$$\sum_{i \in \mathcal{N}^{d(e)}(e)} f_{i,d(e)}(e) = f(e), \tag{7.10}$$

and

$$\sum_{i \in \mathcal{N}^j(e)} f_{i,j}(e) = \sum_{k \in \mathcal{N}_j(e)} f_{j,k}(e) \ (for\ j \neq s(e)\ and\ j \neq d(e)). \tag{7.11}$$

It can be easily shown by summing equations (7.9) and (7.11) or summing equations (7.10) and (7.11) that we only need equation (7.11) and one of (7.9) or (7.10) in the final problem formulation. The third flow balance equation will be automatically satisfied.

Beside the flow balancing equations, the total flow passing through any link should not exceed the capacity of this link. The capacity of each link depends on the allocated channel and the assigned power to this link. The link capacity also depends on the interference model. Denote $c_{i,j}$ the capacity of link $l_{i,j}$. Thus, for a link $l_{i,j}$ and based on our channel allocation and power control modeling, we have:

$$\sum_{e \in \mathcal{E}} f_{i,j}(e) \leq c_{i,j} = w(h_{i,j}) \log_2 \left(1 + \frac{g_{i,j}}{\eta w(h_{i,j})} p_{i,j}\right). \tag{7.12}$$

where η is the is the ambient Gaussian noise density. Note that the denominator inside the log function contains only $\eta w(h_{i,j})$ because of our assumed interference model. This equation assumes a conflict-free channel assignment which is not always guaranteed by our solution algorithm. Hence, the $c_{i,j}$ from equation (7.12) can be considered the conflict-free link capacity. In the next section, we change this equation in the final problem formulation to account for the possible channel assignment conflicts.

7.4.5 Cross Layer Modeling

In this section, the power control, channel allocation, and the flow routing models are combined to represent the cross layer problem. First, recall that $\mathcal{L}_C$ is the set of all directed communication links that can be established in the network when nodes transmit at maximum power. Then define $\mathcal{G}_C = (\mathcal{N}, \mathcal{L}_C)$ as the communication graph. Similarly, define $\mathcal{G}_I = (\mathcal{N}, \mathcal{L}_I)$ as the interference graph where $\mathcal{L}_I$ is the set of all directed interference links indicating which nodes are interfered with when a node transmits at maximum power. In more formal terms, $l_{i,j} \in \mathcal{L}_I$ if $P_i \cdot g_{i,j} \geq \beta$. Maximum power conditions are used to capture the largest possible set of communication and interference links.

Define the set of communication links with which node i can potentially interfere, $\mathcal{L}_C^i$, the set of all allowable channel-power assignment pairs for the links in $\mathcal{L}_C^i$, $\mathcal{V}_i$, and the localized channel-power assignment vector for node i, $\mathbf{v}_i \in \mathcal{V}_i$, the same way as in Section 6.3. Define also the set of links that interfere with link $l_{j,k}$ under the channel-power level assignment $\mathbf{v}_i$, $\mathcal{L}_{j,k}^{\mathbf{v}_i}$, the same way as in Section 6.3.

Following our interference model, we denote $R_j^I(p_{j,k})$ the corresponding interference range of node j when transmitting over link $l_{j,k}$ using transmission power $p_{j,k}$. We can calculate $R_j^I(q_{j,k})$ in terms of $R_j^{I_{max}}$ by substituting for β from (7.6) into (7.4) to get:

$$R_j^I(p_{j,k}) = \left(\frac{p_{j,k}}{P_j}\right)^{\frac{1}{\gamma}} \cdot R_j^{I_{max}} \tag{7.13}$$

Since our objective here is to support a predefined set of active user communication sessions, we use the minimization of the required network resources as our objective function. We consider a localized version of the bandwidth-footprint-product (BFP) as our objective function in this work. The BFP was first introduced in [101] by Liu and Wang who prove that it better characterizes the spectrum and space occupancy for cognitive radio networks than maximizing the network capacity. Based on the BFP, the localized objective for node i is to optimize the following:

$$\min_{\mathbf{v}_i} \left[f^{'}(\mathbf{v}_i) = \sum_{l_{j,k} \in \mathcal{L}_C^i} w(h_{j,k}) \cdot \pi \left(R_j^I(p_{j,k})\right)^2 \right], \tag{7.14}$$

Substituting for $R_j^I(p_{j,k})$ from (7.13) into (7.14) and excluding the constant π from the objective function, then we have the following distributed cross layer formulation:

$$\min_{\mathbf{v}_i} \left[f(\mathbf{v}_i) = \sum_{l_{j,k} \in \mathcal{L}_C^i} w(h_{j,k}) \left(\frac{p_{j,k}}{P_j}\right)^{\frac{2}{\gamma}} \cdot (R_j^{I_{max}})^2 \right], \tag{7.15}$$

s.t.

$$\sum_{j \in \mathcal{N}_{s(e)}(e)} f_{s(e),j}(e) = f(e),$$
$$(l_{i,j} \in \mathcal{L}_C,\, e \in \mathcal{E}),$$

$$\sum_{k \in \mathcal{N}^i(e)} f_{k,i}(e) = \sum_{j \in \mathcal{N}_i(e)} f_{i,j}(e),$$
$$(i \neq s(e), i \neq d(e)\ ,\ l_{i,j} \in \mathcal{L}_C,\ ,\ l_{k,i} \in \mathcal{L}_C,\, e \in \mathcal{E}),$$

$$\sum_{e \in \mathcal{E}} f_{i,j}(e) \leq c_{i,j} = \frac{w(h_{i,j}) \log_2\left(1 + \frac{g_{i,j}}{\eta w(h_{i,j})} p_{i,j}\right)}{1 + \left|\mathcal{L}_{i,j}^{\mathbf{v}_i}\right|},$$
$$(l_{i,j} \in \mathcal{L}_C,\, e \in \mathcal{E}),$$

where $p_{j,k}$ is the power assigned to $l_{j,k}$, P_j is the maximum transmission power for node j, γ is the path loss index, $R_j^{I_{max}}$ is the maximum interference range for node j corresponding to using the maximum transmission power P_j, $w(h_{i,j})$ is the bandwidth of $h_{i,j}$ which is the channel assigned to link $l_{i,j}$, and $f_{i,j}(e)$ is the flow rate of session e that is assigned to be routed from node i to node j (a non-negative quantity, i.e. $f_{i,j}(e) \geq 0$). $\left|\mathcal{L}_{i,j}^{\mathbf{v}_i}\right|$ is the cardinality of the set of interfering links with link $l_{i,j}$ under assignment $\mathbf{v}_i$. The denominator in the link capacity equation reflects the fact that all links which interfere with link $l_{i,j}$ cannot be active simultaneously and, hence, have to time-share the channel to which they are assigned, decreasing the maximum link capacity. Notice that the link capacity equation here is changed from equation (7.12) to account for the possible channel assignment conflicts.

7.5 Applying Island Genetic Algorithm to Cross Layer Problem

In this section, we provide the details of how we apply the K-hop iGA to the cross layer formulation of the previous section. We then describe the migration policy and how the feasibility checks are implemented. Finally, we present an overflow feedback mechanism that is used with the K-hop iGA to solve the overflow problem.

7.5.1 K-hop Island Genetic Algorithm Formulation

The implementation of the K-hop iGA as applied to the flow routing cross layer optimization problem are briefly described as follows:

- ***iGA individuals:*** As described in section 7.3.1, the length of each individual in the K-hop iGA depends on the value of K and the problem under concern. Fig. 7.2 shows the structure of a K-hop iGA individual at node i for the cross layer problem described in the previous section. For each outgoing communication link $l_{i,j}$ in the K-hop neighborhood, the K-hop iGA at node i needs to allocate channel and power level only if part of a flow is assigned to this link. We limit the use of other nodes' information in our implementation of the evaluation of individuals' fitness in order to decrease the communication cost. The initial population at each node is generated using random values from the set of available channels and power levels after distributing the flow rates randomly over possible routing paths (next hops). Only links with positive total flow are assigned channel and power level. Each node i generate an initial population only if it is a source of a communication flow or another node j assigns part of a flow to the communication link $l_{j,i}$ and migrates an individual with this information to node i.

- ***iGA Objective Function:*** We use the function $f(\mathbf{v}_i)$ as defined in (7.15) as our

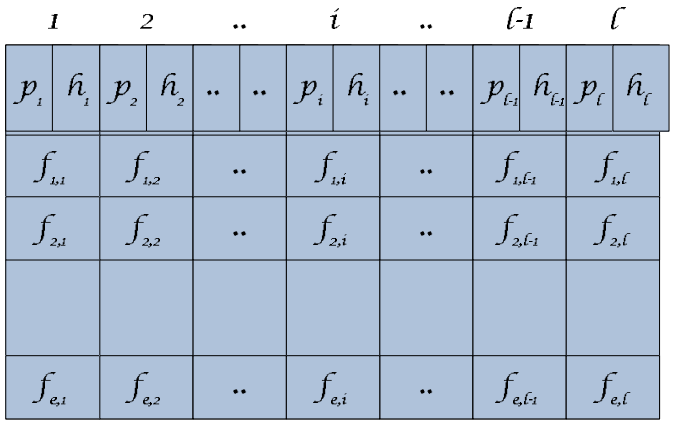

Figure 7.2: Individual Structure for the cross layer flow routing, channel allocation, and power control problem using K-hop iGA

objective function. A better individual will give a lower $f(\mathbf{v}_i)$. At the beginning of every iteration of the K-hop iGA, we compute the fitness of each individual in the population and then sort the individuals based on their objective values.

- ***Mating:*** We use the standard 1-point crossover as our mating form. The parents that are used in the crossover are chosen by tournament selection. In our tournament selection, each of the two parents is chosen as the best individual out of three randomly selected individuals from the parent population pool. Based on the fitness of each individual and a parameter called *keep rate*, we eliminate the worst $\lfloor(1 - \textit{keep rate}) \cdot M)\rfloor$ individuals of the population and use the remaining individuals as the parent population to generate next M individuals generation, i.e. generate $\lfloor(1 - \textit{keep rate}) \cdot M)\rfloor$ new individuals. Fig. 7.3 illustrates the crossover operation for the K-hop iGA as applied to the cross layer problem. The crossover is performed in way that increases the probability of getting feasible individual. This is clear for the channel and power allocation since values are always chosen from the available set. For the flows mating, the crossover increases the feasibility of the new individuals by making a horizontal crossover, i.e. first new child gets the flow distribution from the first flow to a randomly selected flow from the first parent and the rest of flow distribution from the second parent while the other child gets the first part from the second parent and second part

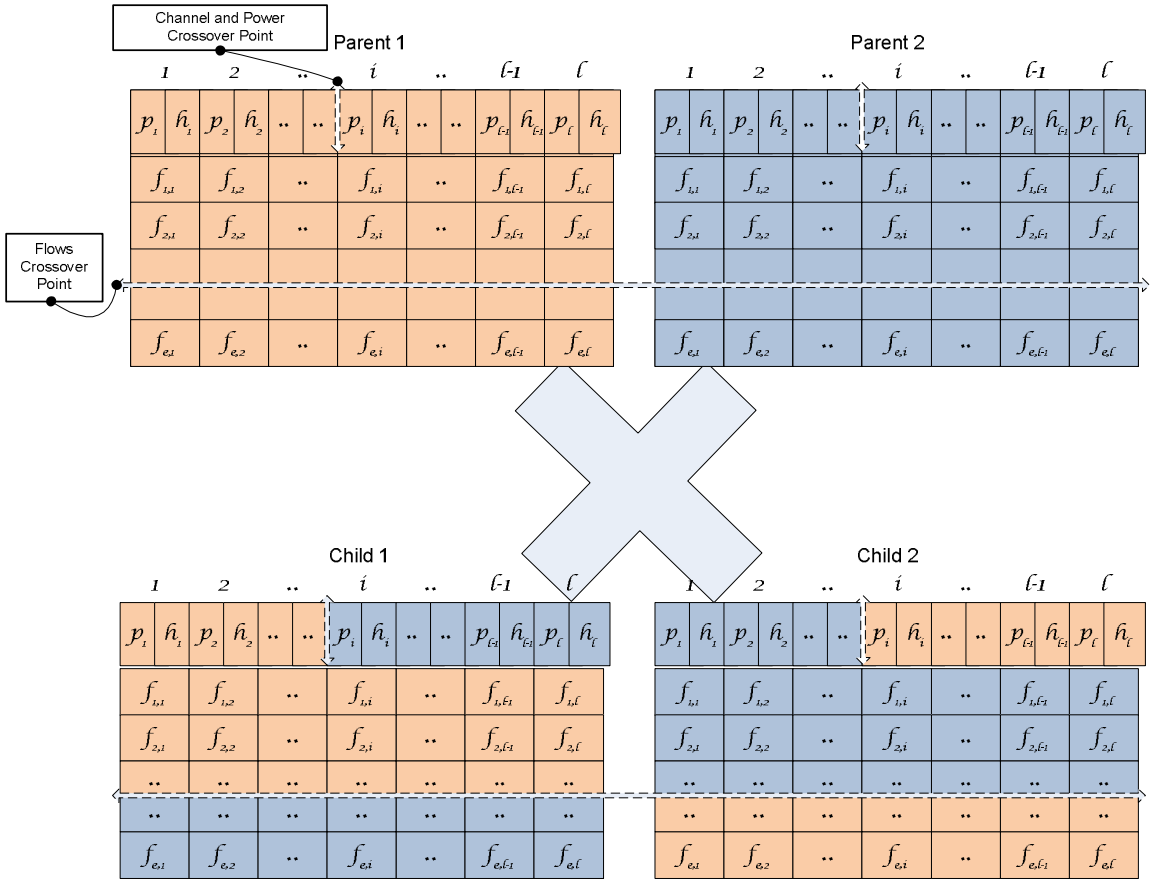

Figure 7.3: K-hop iGA One-Point Crossover

from the first parent. This increases the probability of getting feasible individual by reserving the flow integrity but does not eliminate the possibility of getting an infeasible individual when the total flow at each link is computed.

- ***Mutation:*** We use a parameter called *mutation rate* to define the amount of mutation performed at each iteration of the K-hop iGA. We also use a decreasing mutation rate technique to increase the convergence probability. After finishing the mating, mutation is performed with the *mutation rate* by selecting an individual at random from M excluding the one with the best fitness (lowest bandwidth-footprint-product) and then perform a channel mutation, power mutation, or flow mutation. The channel and power mutation are performed by selecting a link with positive flow rate (one of the active

links) at random from that individual and replacing its assigned channel or power level by other values selected randomly from the set of available channels and power levels of this link. The flow mutation is performed by randomly selecting one of the flows that passes through the chosen individual and then redistributing the flow among possible next hops. If the flow redistribution results in activating a link that was not active, channel and power level are assigned to this link from the set of available channels and power levels of this link. Performing mutations these ways, we again increase the probability of getting feasible individuals for next generation by maintaining flow integrity, but this does not eliminate the possibility of getting infeasible individuals. This infeasibility problem is solved using the feasibility checks discussed later in this section. Excluding the best chromosome from mutation is known as elitism [33] and helps rapidly increase the performance by keeping the best individual (or a few best chromosomes) in next generation.

- ***Convergence:*** The current implementation terminates the algorithm after a fixed number of iterations. Although this may not be the best way, it is sufficient to study the feasibility of using the K-hop iGA to solve the cross layer problem. In the future, we plan to use the cultural algorithm model [102] to add a learning mechanism that controls the K-hop iGA parameters of the K-hop iGA.

7.5.2 Individual Migrations

For the K-hop iGA implementation (as with the standard iGA), we need to define how individuals migrate from one node to another. In our implementation, we use a communication-efficient scheme in which each node shares its best individual with its K-hop neighbors every fixed number of iterations; we call this parameter *migration interval*. Each node receives migrated information from any other node will either increase or decrease its flow distributions based on the difference between previous and current flow assignment from its previous hop. The increase/decrease of the flow distribution is performed by randomly adding or subtract-

ing the flow difference from possible next hop current assignments. As mentioned in the mutation operation, this may result in activating a link(s) that was not active. In this case, channel and power level are assigned to this link randomly from the set of available channels and power levels of this link.

7.5.3 Overflow Feedback Mechanism

In some cases, none of the current population at one node provides a feasible solution. This may be because of assignment to one or more of network flows that result of overflow in one or more of the node's outgoing communication links. In this case, an overflow feedback mechanism is applied. When it is the time for migration, a node checks the feasibility of its best individual. If the best individual is infeasible, all population individuals are also infeasible. A node with an infeasible best individual sends an overflow message to its 1-hop neighbors with information about the overflow problem. This includes the flow numbers and amount of overflow. These amounts are chosen randomly by decreasing each of the flows passing through overflowed links to obtain a feasible flow allocation. Each node in the 1-hop neighborhood decreases the assignment of flows to this overflow node if any. The decreased flow amounts are rerouted through alternative paths if there are any, or propagated to the previous hop as an overflow.

To avoid cycling through this infeasibility problem again, each node keeps an estimate of the next hop capacity and decreases this value by a small random amount when it receives a message indicating an overflow problem with this next hop. This estimate also helps the K-hop iGA converging faster by decreasing the possibility of infeasible flow assignments.

7.5.4 Feasibility Checks

When it is the time for individuals' evaluation, the feasibility of each individual is checked using the link capacity equations from the previous section (see equation (7.15)). The total

flow assignment of each link with positive flows of every population individual is compared to the link capacity and the individual is marked as infeasible if any of its links' assignment is infeasible (exceeds link capacity). To exclude infeasible individuals faster from the population, a large infeasibility penalty is added to infeasible individuals' objective values. The power level assigned to an infeasible link is increased and the channel assigned to this link is changed to try to solve the link overflow problem. In case all population individuals of one node are infeasible, the overflow feedback mechanism described previously is used to move individuals of next generations back into the feasibility region.

7.6 Simulation Settings and Results

Table 7.1 summarizes the simulation settings for network and K-hop iGA parameters that are used to run our simulations. The authors of [87] provided us with the source code of their branch and bound algorithm[2]. Thus, we use the same simulation parameters as in [87] to compare our work with the theoretical bounds presented in this reference. Network size ranges from 5 to 50 nodes and the network area is adjusted to keep a constant node density, with the network area side being 50 meters for a network of size $N = 20$. The ad-hoc network is generated by randomly placing network nodes uniformly over the network area. The maximum transmission range is set to 20 meters and maximum interference range is set to 40 meters.

We set the maximum number of available channels in the sensed spectrum to 10 channels. The bandwidth of each channel is 50. We divide the network area into square subareas with the subarea side being 10 meters. We then randomly select a set of available channels at each subarea from the 10 available channels with an upper limit of 9 channels and a lower limit of 6 channels. Each node i is allocated a set of available channels $\mathcal{C}_i$ the same as its subarea. We set the maximum power to 20 dBm, the number of power levels to 16 levels linearly

[2]We would like to thank Dr. Yi Shi and Dr. Thomas Hou for providing us with a copy of this code for performance comparison purposes.

Table 7.1: Network and K-hop Island Genetic Algorithm parameters' setting

Network and Nodes Parameters' Setting	
Area	Square w/ constant node density
Number of Nodes (N)	5-50
Node density	8000 $nodes/km^2$
Max Communication Range ($R^{T_{max}}$)	20 $meters$
Max Interference Range ($R^{I_{max}}$)	40 $meters$
Available Channels	10
Channels Lower Limit	6
Channels Upper Limit	9
Channel Bandwidth	50
Maximum Power (P)	20 dBm
Number of Power Levels ($\|Q\|$)	16
Path Loss Index	4
Flow Rate Lower Limit	10
Flow Rate Upper Limit	80
K-hop iGA Parameters' Setting	
Population Size per island (M)	50
Number of Iterations	1000
Keep Rate	0.5
Mutation Rate	0.15
Migration Interval	20 Iterations

spaced (in mW) ranging from no power (level 0) to maximum power (level 15), and the path loss index to 4 for all nodes. For each generated flow session, the source and destination are randomly selected from the set of network nodes, $\mathcal{N}$. The rate of each flow is uniformly selected with an upper limit of 80 and a lower limit of 10. Note that the unit of the flow rate and the channel bandwidth are of no importance as long as they are scale matched (for example, kHz for the channel bandwidth and kbits/sec for the flow rate).

For the K-hop iGA parameters, we set the population size M to 50 for all nodes, the *keep rate* to 0.5, the *mutation rate* to 0.15, the *migration interval* to 20 iterations, and the total number of iterations to 1000. Each result reported in this work is the average over at least 10 runs for each of 10 different network instances.

The next sections evaluate the performance of the K-hop iGA through simulations and provide simulation results. We compare the K-hop iGA performance to the performance of the branch and bound algorithm in [87]. For the ease of comparison, we normalize all results exactly the same way the results in [87] are normalized (normalize all units for distance, bandwidth, rate, and power). We study the effect of localization by comparing the performance of the K-hop iGA for different K values. We then study the effects of network density and flow density on the K-hop iGA performance. Finally, we compare the performance of the K-hop iGA with underlying node-disjoint routing algorithm to the performance of the K-hop iGA with link-disjoint routing.

7.6.1 Performance of K-hop Island Genetic Algorithm

In this section, we compare the K-hop iGA performance to the performance of the branch and bound algorithm presented in [87]. We compare the bandwidth-footprint-product (BFP) of solutions of each algorithm over networks with sizes ranging from 5 to 25 nodes. The number of generated flow sessions is 2 for the 5-node network and increasing linearly up to 6 flow sessions for the 25-node network. Since the branch and bound algorithm is a centralized algorithm, we compare it to the performance of the M-hop iGA. We later show the effect of

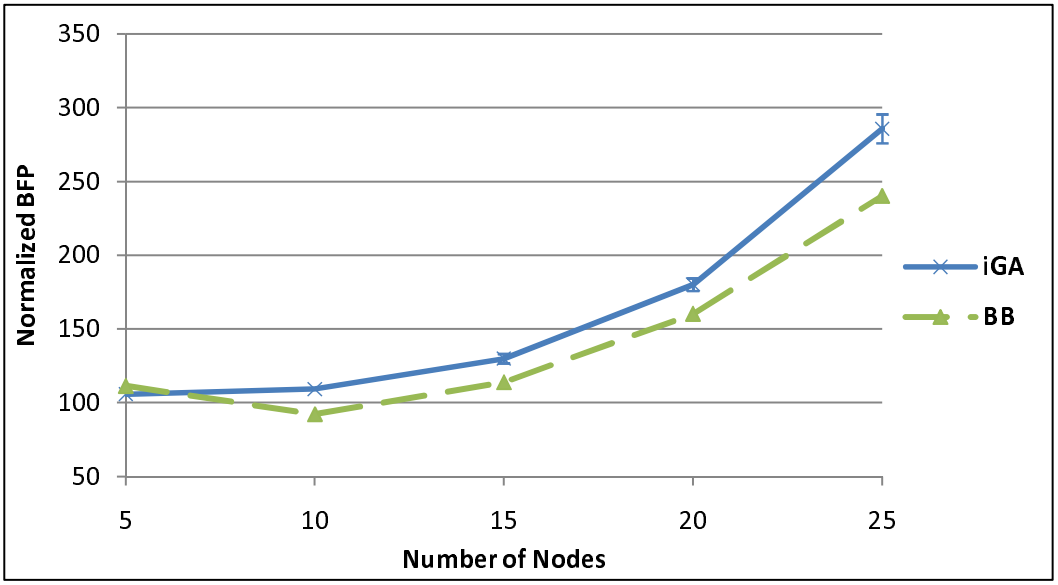

Figure 7.4: Performance of M-hop iGA (along with 95% confidence intervals) vs. Branch-and-Bound

using smaller K values. Fig.7.4 shows the average (normalized) BFP of the M-hop iGA and the branch and bound technique.

Fig.7.4 shows that the average BFP of the M-hop iGA is about 85% of the branch and bound BFP which in turn is guaranteed above 95% of the optimal. The M-hop iGA has better results for the network of size 5 nodes because for this small network, the M-hop iGA converges to the optimal solution while the branch and bound technique stops as soon as it gets to a solution that is above 95% of optimal. For small network sizes, the M-hop iGA always converges to a good solution but we noticed that this solution is limited by the available routing paths discovered by the link-disjoint routing algorithm. This is the main reason for the performance difference between the M-hop iGA and the branch and bound technique. The small 95% confidence interval at each point in the figure shows the consistency of the iGA outcome from one run to another with the different randomly-generated initial populations. Unfortunately, we cannot draw any general conclusions out of this comparison because it is based on a single network instance for each network size. This is because while it takes few minutes for the M-hop iGA to converge to a final solution, the branch and

bound algorithm takes more than a week for some network instances to terminate. On the other hand, the low complexity of the iGA compared to the branch and bound algorithm can be seen as a big advantage even if the final solution is about 85% of the branch and bound solution. One way to improve the performance further is to use a local coordination mechanism similar to the one presented in [97]. A more thorough performance comparison is needed in the future to generalize our conclusions in this section. It is also worth mentioning here that while the iGA provides a relatively good performance compared to the branch and bound technique, we have experienced poor performance of the iGA when the flow rate requirements of the optimal solution are close to the link capacities because of the random flow distribution of the iGA. In other words, the iGA unnecessarily activates more links to route small portions of one flow while they could fit (but tightly) with other flow(s) portions routed on a previously activated link(s).

7.6.2 Effect of Localization

In this section, we compare the performance of the M-hop iGA to the performance of 1-hop and 3-hop iGA to study the effect of localization on the iGA performance. Note that the maximum number of hops, and hence the value of M, per simulated network ranges from 4 to 6 hops. We applied the three versions to networks of size ranging from 10 to 50 nodes, each with 5 communication flow sessions. Fig. 7.5 shows the average performance of each version over all simulated networks.

Fig. 7.5 shows, as expected, that the M-hop iGA gives the best performance, lowest BFP, for all network sizes. The 3-hop iGA also gives a better performance than the 1-hop iGA performance for all network sizes except at 50 nodes. On the other hand, the 1-hop iGA is the fastest because of the less amount of processing performed by each node. Interestingly, the performance difference is only 8.6% on average between the 1-hop and M-hop iGA; perhaps because of the underlying routing protocol. However, the main difference is that the 1-hop iGA leads to infeasible solutions in some simulations because of the localized decisions of

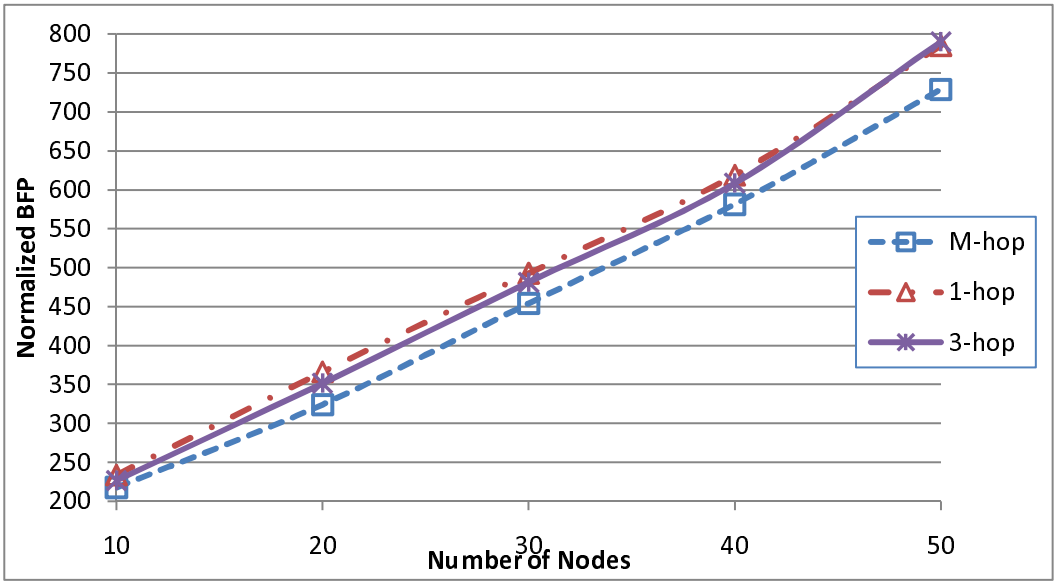

Figure 7.5: Performance of M-hop iGA vs. 1-hop iGA and 3-hops iGA

network nodes. This reveals that in case of few network flows with low rate requirements, the 1-hop iGA can be safely used without a big drop in the network performance. Note that unlike the exponential increase in Fig. 7.4, the BFP curve appears to increase linearly in Fig. 7.5. This is because we fix the number of communication flows to 5 for all network sizes for Fig. 7.5. As the M-hop iGA gives the best performance among simulated versions in this section, we use the M-hop iGA for all performance studies presented in next sections.

7.6.3 Effect of Network Density

In this section, we study the effect of the network density on the K-hop iGA performance by applying the M-hop iGA to 20 nodes networks of area side ranging from 30 to 90 meters. The number of generated flow sessions for each network is 5 sessions. Fig. 7.6 shows the performance of the K-hop iGA as the network area changes.

Fig. 7.6 shows that as the network density decreases (area increases), the BFP of the K-hop iGA increases and then stabilizes for area side of 80 meters. The increase of the K-hop iGA BFP is reasonable since for larger areas, data flows need longer paths and/or higher power

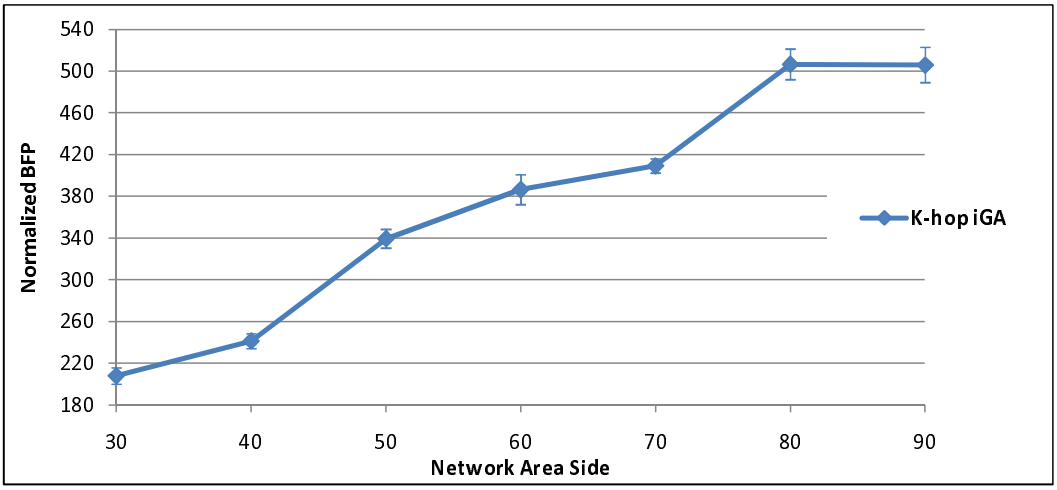

Figure 7.6: Effect of Network Density (along with 95% confidence intervals)

levels to reach their destinations. At some point, the BFP stabilizes when all route nodes are required to use their maximum transmission power to deliver data to their next hops and the number of possible routing paths decreases to 1 or 2 because of nodes distribution. The iGA solutions also experience less variation as the network density decreases because the number of possible routing paths decreases. Reducing the network density (increasing the network area) further results in disconnected network instances with no feasible solutions.

7.6.4 Effect of Number of Flows

In this section, we study the effect of number of flow sessions on the performance of the K-hop iGA by applying the M-hop iGA to networks of the same size, 20 nodes, but with number of flow sessions increasing from 2 to 8 flows. Fig. 7.7 shows the effect of increasing number of flow sessions on the K-hop iGA performance.

The figure shows that the K-hop iGA BFP increases as the number of flows increases. This is because more links are activated to route flows. However, the curve then stabilizes when the number of activated links is enough to route all data flows. Increasing the number of flows any more does not increase the BFP but results in infeasible network instances (networks

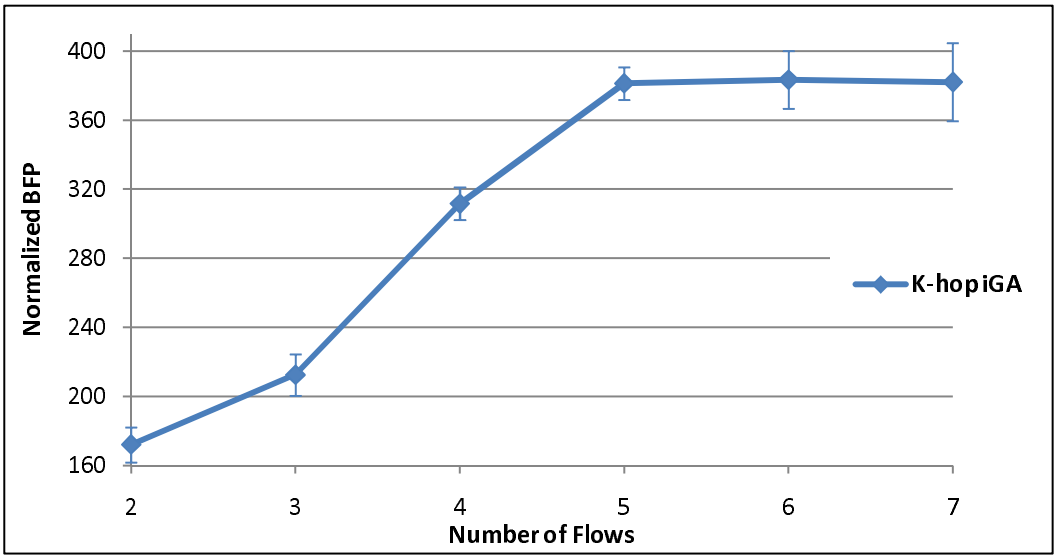

Figure 7.7: Effect of Number of Flows (along with 95% confidence intervals)

with no feasible solutions).

7.6.5 Effect of Underlying Routing Algorithm

In this section, we study the effect of changing the underlying routing protocol from link-disjoint to node-disjoint multipath routing. We apply the M-hop iGA with link-disjoint and node-disjoint routing protocols to networks of size ranging from 10 to 50 nodes. Table 7.2 shows that the average number of paths per flow for link-disjoint multipath routing is greater than the average number of paths per flow for node-disjoint multipath routing for all simulated network sizes. Fig. 7.8 shows the performance of the M-hop iGA when using link-disjoint and node-disjoint underlying multipath routing.

The figure shows that there is almost no performance gained from using the link-disjoint rather than the node-disjoint multipath routing. Interestingly, using the node-disjoint routing rather than the link-disjoint multipath routing leads to better stability. We explain this by noting that for most of our simulations the M-hop iGA could not completely converge in only 1000 iterations. Decreasing the number of possible flow routes by using a node-

Table 7.2: Average Number of Paths per Flow for Link and Node Disjoint Routing Protocols

Number of Nodes	Link-disjoint	Node-disjoint
10	4.06	3.70
20	5.64	4.30
30	4.64	3.12
40	6.14	4.16
50	4.91	3.80

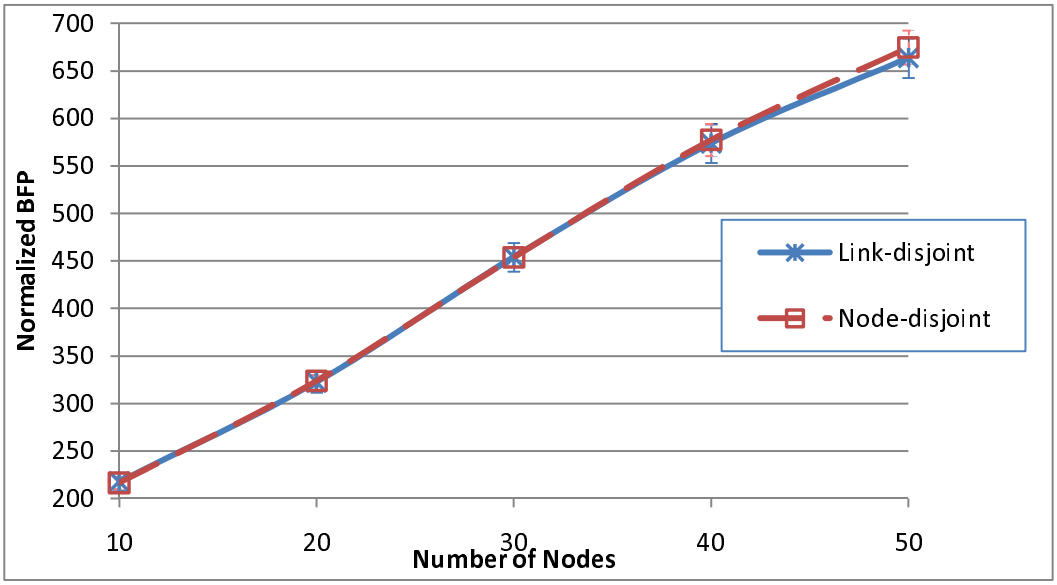

Figure 7.8: Effect of Routing Algorithm (along with 95% confidence intervals)

disjoint routing algorithm reduces the size of the solution space, resulting in better stability. On the other hand, the node-disjoint routing paths are a subset of the link-disjoint routing paths thus increasing the likelihood of infeasible solutions in high-requirement flow routing problems because of the fewer number of possible routing paths in the node-disjoint case.

7.7 Summary and Contributions

In this chapter, we introduced a new variation of the island genetic algorithm (iGA), K-hop iGA and applied it to the flow routing considering both channel allocation and power

control. The K-hop iGA implementation is flexible and general enough to solve a large class of communication and networking problems with its controllable cooperation range. We presented a distributed formulation for the cross layer flow routing, channel allocation, and power control problem. We then applied the K-hop iGA to the cross layer model.

We evaluated the K-hop iGA performance through computer simulations. Comparing the K-hop iGA performance to a theoretical solution based on the linear relaxation and branch and bound techniques revealed a good performance of the K-hop iGA considering the complexity reduction and short running time as compared to the theoretical approach. We studied the performance of the K-hop iGA for different K values and the effect of different factors on the performance of the K-hop iGA including the network and flow densities and the underlying routing protocol. The K-hop iGA performance experiences some variations for different simulation settings. This urges the need for incorporating a learning mechanism to choose the best K-hop iGA parameter settings based on the problem and network perception. For example, with fewer flow sessions or lower network density, the number of iterations and the value of K can be decreased. The next chapter concludes the dissertation, discussing the need for incorporating learning mechanisms into the cognitive node architecture and suggesting suitable learning techniques for the new learning processes.

The work presented in this chapter has resulted in the following manuscript:

1. Mustafa Y. ElNainay and Allen B. MacKenzie, "Flow Routing using K-hop Island Genetic Algorithm," *IEEE Transactions on Mobile Computing*, under review.

Chapter 8

Conclusions and Future Work

Modern wireless networks face many challenges due to the tremendous increase in wireless network applications and the continued growth of the number of users. The complexity and heterogeneity of these networks urges the creation of new networks that are capable of observing the surrounding environment and adapting to network changes. The Cognitive Network (CN) is a promising solution to future network challenges, integrating distributed reasoning and learning with a reconfigurable protocol stack and performance monitors. This Chapter summarizes our research contributions to the CN field and presents potential future short term and long term extensions to our work.

8.1 Summary and Contributions

In this section, we summarize the primary contributions of this dissertation. The main focus of this dissertation was on two related objectives. The first was to design an architecture for a cognitive node in a cognitive network (CN) that can face current and future networking challenges. CNs should have the ability to observe the surrounding environment, autonomously plan solutions for encountered problems, and adapt to network changes. The

design of the cognitive node architecture has to take the CN expectations into account and integrate all necessary components to facilitate the operation of the CN while allowing for secure information exchange among network nodes. The cooperation and information exchange among network nodes is essential for achieving the network-wide goals instead of ending with solutions based on node-centric goals.

The second objective was to choose a good candidate for the CN distributed reasoning algorithm and justify our choice. We have chosen the island genetic algorithm (iGA) as the basis for the distributed reasoning algorithm and justified our choice of the genetic algorithm in general and iGA in particular through discussions in Section 3.4. We have then applied the iGA and our developed localized variations of the iGA to three communication and networking problems that are of high importance to the dynamic spectrum CN. Moreover, we have implemented a proof of concept implementation for the dynamic spectrum CN using available hardware and open source and our own software packages. The following list summarizes the major contributions of this dissertation to the CN research:

- Chapter 2 presented our taxonomy for the current research work coupling communication and networking with intelligence into three categories: cognitive radios (CRs), cognitive radio networks (CRNs), and cognitive networks (CNs). We then described the definition of each category. Moreover, we surveyed the literature on cognitive node and cognitive network architectures.

- Chapter 3 presented our proposed cognitive node architecture followed by a brief description for the functions and connections of major architecture components. We then identified our research focus on the distributed reasoning component of the cognitive node architecture. Following suggestions from a previous study by Friend in [4], we chose the island genetic algorithm (iGA) as the basis for the CN reasoning algorithm. Through brief discussion, we justified why we think the iGA is a promising distributed reasoning algorithm for CNs.

- Chapter 4, 6, and Chapter 7 investigated the applicability of iGA as the distributed

reasoning algorithm though applying the standard iGA and our localized variations of the iGA to various communication and networking problems and studying their performance through computer simulations. In Chapter 4, we developed a model of the channel allocation problem that is unique to dynamic spectrum CNs. We considered the fact that each node may sense a different number and set of available channels. We then applied the standard iGA to the formulated channel allocation problem and compared its performance to the optimal solution through simulations. The results revealed excellent performance for this problem, showing that iGA is a promising algorithm to be used with CNs. However, studying the effect of network size on the iGA convergence speed uncovered a major problem with applying the standard iGA to communication and networking problems: the scalability problem. The standard iGA lacks scalability because while it distributes the computation over network nodes, each node still requires global network knowledge in order to search the entire search space for a solution. Clearly, this is not scalable for large networks.

- In Chapter 5, we described our iGA-based cognitive network prototype implementation. In this implementation, we showed how cognitive radios can be organized to form a cooperative ad-hoc cognitive radio network that utilizes the available spectrum opportunistically and efficiently through dynamic channel allocation. We used cognitive radios that are software defined radios (SDRs) consisting of a Linux laptop and a universal software radio peripheral (USRP) integrated with our extended implementation of the GNU Radio software package and our iGA implementation for the channel allocation. We formed a chain topology using four laptops and studied the network performance using the Iperf software tool. Because of the limited available hardware in terms of laptops and USRPs, the main objective of this chapter was to describe the prototypical implementation rather than to perform a detailed experimental study.

- Having applied the standard iGA to one of the challenging problems for the dynamic spectrum access (DSA) application and determined the iGA scalability problem, we

shifted our focus in Chapters 6 and 7 to developing more promising variations of the iGA that better suit the needs of CNs and applying them to more complex cross layer problems. In Chapter 6, we developed a localized variation of the iGA (LiGA) that solves the scalability problem. We formulated a model for the joint power and channel allocation problem and applied the LiGA to this cross layer problem. Through simulations, we compared the LiGA to the standard iGA and showed that the LiGA performance is competitive to the standard iGA while being much more scalable. Furthermore, we studied the impact of two types of non-cooperation on the CN performance. The first type of non-cooperation is the existence of non-cognitive nodes in the network, given the fact that cognitive nodes must interoperate with non-cognitive nodes. The second type of non-cooperation is the possible existence of selfish nodes among the cognitive nodes. We studied both types of non-cooperation through simulation. The simulation results showed that the cognition increases network performance even with the existence of selfish nodes. Moreover, cooperative network achieves much better performance than non-cooperative network. Both studies suggest the importance of an enforcement mechanism to prevent cheating or a reputation mechanism to avoid it.

- The LiGA is a promising technique to solve the standard iGA scalability problem when applied to large networks. However, not every communication and networking problem can be solved with local information. Thus, we developed the K-hop iGA which is a generalized version of the LiGA that suits a larger class of communication and networking problems with controllable cooperation and migration range. Chapter 7 presented the details of the K-hop iGA and applied it to the flow routing problem. The flow routing problem is formulated in the context of dynamic spectrum CN with both power control and channel allocation. We assumed the existence of an underlying multipath, link-disjoint routing protocol. The K-hop iGA is then limited to use routing paths provided by the routing protocol. This practical assumption helps the K-hop iGA converges faster because it decreases the search space size. However, it limits the

maximum performance the K-hop can achieve because it limits the number of flow routing paths between sources and destinations. Through simulations, we compared the performance of our approach to a theoretical algorithm that guarantees a solution within a percentage of the optimal. Simulation results showed that the K-hop iGA can achieve about 85% of the optimal while running in a fraction of time compared to the other algorithm.

While working on each of the above contributions, more open problems and necessary improvements become apparent. The next sections discuss some possible short term and long term improvements and developments to our work.

8.2 Short Term Future Work

Here, we present a list of our suggested short term future work:

1. In Chapter 6, we briefly studied the effect of two types of non-cooperation on the performance of cognitive networks (CNs). However, the study was simple and tied to the problem presented in this Chapter. A more thorough study for the selfish and non-cooperation behaviors is needed to understand the impact of these behaviors. Based on this understanding, investigation of the necessary modifications for the current proposed reasoning algorithm is needed to directly incorporate uncertainty into the reasoning algorithm processing to overcome the impact of these behaviors.

2. Conclusions of the CN prototype implementation in Chapter 5 were limited to the available hardware used in experiments. Running experiments over larger network sizes will draw more realistic conclusions. A network testbed with experiment reproducibility will allow for better performance analysis. Emulab [103] and ORBIT [104] are two candidates for conducting these experiments with their ability to produce controllable, predictable, and repeatable network environments.

3. In all our work, we used binary interference models and did not account for additive interference. A physical interference model that takes the additivity of interference into account would be more accurate. We do not expect this will change our conclusions regarding the efficiency of the island genetic algorithm (iGA) performance but there is still a need to study the effect of this more accurate model on the iGA performance.

4. Our work in this dissertation is developed with a static network. This assumption simplifies our modeling and simulations but limits our conclusions regarding network dynamics. Studying the effect of link failures, node mobility, and other network dynamics is needed to generalize our current results and conclusions, or to investigate the necessary modifications to overcome the impacts of network dynamics. As mentioned before in Chapter 5, we assume a cognitive controller that monitors the network performance and re-runs the iGA as needed. However, we need to evaluate this solution in a dynamic network.

5. The iGA implementations presented in Chapter 4 through Chapter 7 were customized to problems investigated in each chapter. As we made our point of the applicability of the iGA as the basis of the CN reasoning algorithm, a generalized implementation of the iGA that can be applied to current and future communication and networking problems needs to be developed. This implementation should receive a problem description from the cognitive controller and build the iGA individuals and the iGA objective function based on this description. This generalized reasoning algorithm would require significant effort to implement. However, building a small prototype should be feasible in few months but was beyond the scope of this dissertation.

Having presented some of short term future works, the next section discusses two main problems in the current CN reasoning approach which suggest the need for incorporating learning mechanisms into the cognitive node architecture. We consider this as a long term goal because more research is needed to choose suitable candidate learning algorithms and to implement, incorporate, and evaluate the performance gain of these algorithms.

8.3 Integration of Learning

Learning from past experiences saves time in finding solutions for new problems and also helps in making immediate decisions in cases similar to past solved problems. In a real time environment, like a cognitive network, two levels of reasoning are required, one for taking immediate decisions based on past experience and the other for searching for suitable solutions for newly encountered problems. Coupling learning with reasoning is necessary to learn from past experience and then maintain this information in a knowledge base that helps the reasoning algorithm make appropriate decisions in cases require immediate action.

Two main problems of using the island genetic algorithm (iGA) and our localized variations motivate the need for enriching the reasoning of the cognitive node architecture with learning. The first is a general problem with GAs and hence the iGA, namely the convergence speed. Although iGA distributes the computation over all network nodes, iGA still converges relatively slowly for large scale problems. This may not be suitable to meet the timing requirements of communication and networking problems. Tuning the GA parameters to make the iGA converge fast, for example by decreasing mutation rate, may lead to a premature convergence that is far from optimal. Thus, there is a need for a learning process that perceives the problem domain and maintains historical knowledge to guide the iGA to better fast solutions.

The other problem is that the iGA parameters' setting are manually configured. This violates the autonomy objective of cognitive networks (CNs). Moreover, successful parameter setting depends on both the problem under concern and current network and resource conditions. Thus, there is a need for a learning process that can learn successful iGA configurations and maintain a configuration knowledge base. This knowledge base can be used later to automate the parameter settings according to the perceived network conditions and the problem under concern. The next two subsections investigate possible solutions for the aforementioned problems.

8.3.1 Improving the island Genetic Algorithm Performance

Techniques to accelerate the genetic algorithm (GA) convergence speed and/or improve the GA performance while reserving the balance between exploitation and exploration to get better results exist in the literature. In [105], Herrera *et al.* propose the use of two fuzzy tools to improve GA performance: The authors propose the use of fuzzy connectives to design new crossover operators to improve the GA results and propose the use of fuzzy logic to control the population diversity and the exploitation-exploration balance. In [69], Rasheed and Hirsh use case-based learning to improve GA performance. Using a history of previously evaluated points, they evaluate a new point only if it is promising, i.e. close to at least one previously evaluated point that has a fitness that is better than some threshold. On the other hand, they use a "Diversity Maintenance Module" to preserve the balance between exploitation and exploration and avoid premature convergence. A similar idea is presented in [106] which introduces a concept called the "Intelligent Fitness Function" to improve GA performance by avoiding repeated evaluation for the same point. Intelligent fitness functions maintain short term and long term memories and use this information to save time wasted by evaluating same points again.

In [107], Rondeau provides a discussion about the potential benefits of coupling the GA with case-based learning. Rondeau presents an example of using case-based system to select a better initial GA population. Moreover, the author points to the potential use of case-based learning to adjust optimization parameters like the population size and terminating conditions but without further details. In [108], Collins and Joslin propose the use of domain-specific knowledge to transform the search space of a particular problem so that it is easier for the GA to converge faster to a good solution. In [109], Liou *et al.* propose a new genetic algorithm called the Incremental Improving Genetic Algorithm (IIGA) which use problem decomposition technique to generate better initial population and refine the evolution of the GA by retaining illegal solutions in the population space to avoid premature convergence.

A more general framework that match our CN goals is the Cultural Algorithm (CA) frame-

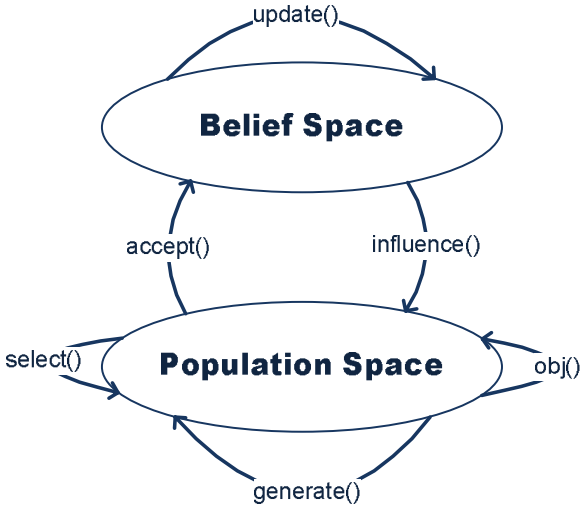

Figure 8.1: The Framework of Cultural Algorithm

work [110]. The CA is an evolutionary computation model derived from the cultural evolution process. With CA, an individual can share acquired knowledge with all other individuals in the population, as opposed to other evolutionary models in which individual's knowledge transfers only to its offspring. CA has been applied successfully to solve various constrained optimization problems [111], [112], [113] and problems in dynamic environments [114].

The CA framework, shown in Fig. 8.1, has three main components: Population Space, Belief Space, and Interaction Protocol. The first component is the population space which models the natural evolutionary systems and can be implemented using any population-based computational model, such as GA and Particle Swarm Optimization. The second component is the belief space which is the main addition of the CA. The Belief Space models the "culture" that serves as a repository that stores and manipulates individuals' successful experience and knowledge. The belief space can then be used to convey this experience to all other individuals which helps enhancing the global search ability and minimizing the computation cost.

The final component of the CA model is the Interaction Protocol that describes how successful knowledge is extracted from the population space to the belief space, through the "accept" function, and how this knowledge influences the evolution of population individuals,

through the "influence" function. In this way, the population space and belief space interact and support each other. As such, CA can be considered as a dual inheritance system in which both population individuals and their beliefs evolve over time in parallel. Experiences of individuals chosen through the "accept" function are used to update knowledge (Beliefs) of the Belief Space via the "update" function. The other three functions in the CA model, "obj", "select", and "generate", are the functions used in natural evolutionary systems (like the GA) to evaluate population individuals and produce next generation.

We propose the use of the cultural algorithm framework with its belief space because it has been applied successfully as evolutionary programming improvement framework [112] [113] [114]. It is general framework that can capture all previously discussed techniques into its implementation. For example, using the historical knowledge of the cultural algorithm belief space, we can implement the same improvements presented in [69], [107] and [106]. Moreover, using the interaction protocol with a fuzzy implementation of the belief space, we can implement same improvements presented in [105]. Domain knowledge in the belief space can be used to implement the same concept presented in [108].

8.3.2 Automating Reasoning and Learning Initial Configuration

The problem of choosing the initial parameter settings of the reasoning and learning process manually is that it is impossible to choose one group of settings that is suitable for all targeted problems. Moreover, the current manual configuration of the iGA parameters violates the autonomy objective of CNs. Thus, there is a need for a learning process that can learn successful configurations and maintain a configuration knowledge base. This knowledge base can be used later to automate the iGA parameter settings according to the perceived environment conditions and the problem under concern. We propose the use of reinforcement learning techniques [115] to help solving this problem by continuously monitoring the effectiveness of the reasoning and learning processes and tuning their initial settings for next runs accordingly. The responsibility of this task will be implemented as one of the cogni-

tive controller tasks, namely reasoning and learning controller, with the necessary historical knowledge maintained in the reasoning and learning configuration knowledge base.

Fig. 8.2 shows our final proposal for the cognitive node architecture that incorporates two main learning processes to solve the slow convergence and manual configuration problems. The first learning process is coupled with the reasoning process inside a cultural algorithm-based reasoning and learning process, namely the belief space, and the other learning process resides inside the cognitive controller as a subtask of the reasoning and learning controller process. The belief space is responsible of learning and maintaining problem solving knowledge generated from elite individuals' experience. Then, this knowledge can be used to guide the evolutionary operators in producing next generations to enhance the global search ability and accelerate the search convergence. The reasoning and learning controller process inside the cognitive controller are responsible for setting the initial configuration of the distributed reasoning and learning process. This also includes learning successful configurations and storing this knowledge in the reasoning and learning knowledge base. Then, this knowledge can be used later for configuring the distributed reasoning and learning process to improve its results, for example by increasing the number of procedure iterations when a better solution is required. Two candidate algorithms for implementing the belief space are the previously applied techniques for the cultural algorithm: Fuzzy Logic [116] and Version Spaces [117]. Two candidates for implementing the learning of initial reasoning and learning configuration are Reinforcement Neural Networks [118] and Case-based Learning [107]. Further work is required to choose and apply learning algorithms to CNs.

8.4 Final Statement

Cognitive Network is one of the multidisciplinary research fields that forces a unique opportunity for researchers from different areas to cooperate to form a concrete solution. Contributions from many colleagues have facilitated the integration of different pieces of work

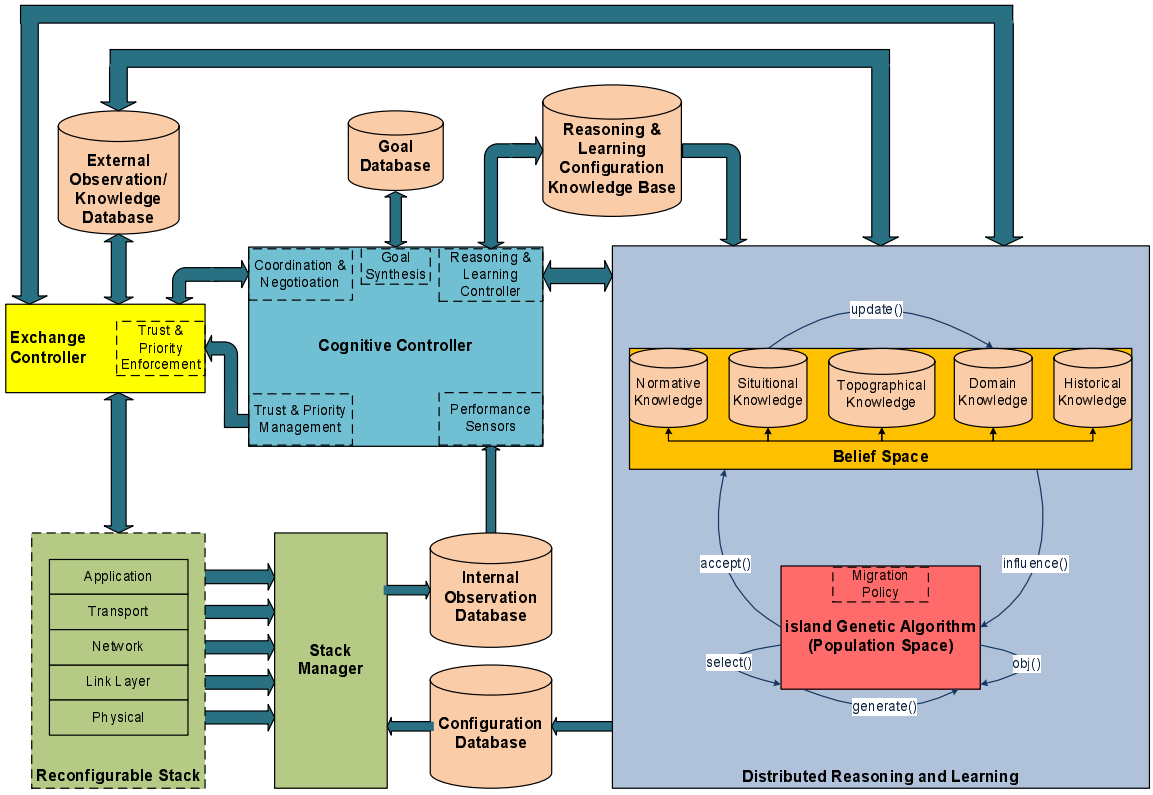

Figure 8.2: Cultural Algorithm-based Cognitive Node Architecture

presented in this dissertation. We have proposed an architecture for a cognitive node in a cognitive network that is able to solve current and future networking problems. We have also proposed and evaluated the performance of the island genetic algorithm as a distributed reasoning algorithm that fits our architecture. We have developed different localized variations of the island genetic algorithms that better suits a larger class of communication and networking problems and scales the standard island genetic algorithm to large networks. We have presented some potential short and long term extensions to our work. There are though more open problems in the proposed architecture that have not been investigated yet. Cognitive Network is a relatively new research field and there is much work still needed to have a complete solution.

Appendix A

Acronyms

ACO Ant Colony Optimization

AI Artificial Intelligence

AODV Ad-hoc On-demand Distance Vector

BFP Bandwidth-Footprint-Product

CA Cultural Algorithm

CBR case-based reasoning

CN Cognitive Network

COTS Commercial off-the-shelf

CR Cognitive Radio

CRM Cognitive Resource Manager

CRN Cognitive Radio Network

CSMA carrier sense multiple access

DSA Dynamic Spectrum Access

FFT Fast Fourier Transform

FPGA field-programmable gate array

GA Genetic Algorithm

GCP Global Control Plane

GMSK Gaussian Minimum Shift Keying

GUI Graphical User Interface

iGA island Genetic Algorithm

IP Internet Protocol

LiGA Localized island Genetic Algorithm

MAC Medium Access Control

MANET Mobile Ad-hoc Network

MDP Markovian Decision Process

MPR MultiPoint Relay

OLSR Optimized Link State Routing

PSD Power Spectral Density

PSO Particle Swarm Optimization

QoS Quality of Service

RF Radio Frequency

RTT Round Trip Time

SA Simulated Annealing

SDR Software Defined Radio

SINR Signal-to-Interference plus Noise Ratio

TS Tabu Search

UDP User Datagram Protocol

USB Universal Serial Bus

USRP Universal Software Radio Peripheral

WSGA Wireless System Genetic Algorithm

Bibliography

[1] G. Staple and K. Werbach, "The end of spectrum scarcity," *IEEE Spectrum*, vol. 41, no. 3, pp. 48–52, Mar. 2004.

[2] J. Mitola III and G. Q. Maguire, "Cognitive radio: Making software radios more personal," *IEEE Personal Communications*, vol. 6, no. 4, pp. 13–18, Aug. 1999.

[3] R. W. Thomas, L. A. DaSilva, and A. B. MacKenzie, "Cognitive networks," in *First IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN'05)*, (Baltimore, Maryland), pp. 352–360, IEEE, 8-11 Nov. 2005.

[4] D. H. Friend, *Cognitive Networs: Foundations to Applications*. PhD thesis, Electrical and Computer Engineering Department, Virginia Tech University, Blacksburg, Virginia, 2009.

[5] J. Mitola III, "Software radios-survey, critical evaluation and future directions," in *Proc. IEEE NTC'92*, pp. 15–23, 19–20 May 1992.

[6] R. J. Lackey and D. W. Upmal, "Speakeasy: the military software radio," *IEEE Communications Magazine*, vol. 33, no. 5, pp. 56–61, May 1995.

[7] S. Haykin, "Cognitive radio: Brain-empowered wireless communications," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 2, pp. 201–220, Feb. 2005.

[8] T. W. Rondeau, C. W. Bostian, D. M. Maldonado, A. Ferguson, S. Ball, B. Le, and S. Midkiff, "Cognitive radios in public safety and spectrum management," in *33rd Research Conference on Communication, Information and Internet Policy*, Sept. 2005.

[9] D. Scaperoth, B. Le, T. Rondeau, D. Maldonado, C. W. Bostian, and S. Harrison, "Cognitive radio platform development for interoperability," in *Proc. IEEE MILCOM'06*, (Washington, D.C.), pp. 1–6, 23–25 Oct. 2006.

[10] 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.

[11] G. Ganesan and Y. G. Li, "Cooperative spectrum sensing in cognitive radio networks," in *Proc. IEEE DySPAN'05*, (Baltimore, MD), pp. 137–143, 8–11 Nov. 2005.

[12] R. Chen, J.-M. Park, and K. Bian, "Robust distributed spectrum sensing in cognitive radio networks," in *Proc. IEEE INFOCOM'08*, (Phoenix, AZ), pp. 1876–1884, 13–18 April 2008.

[13] S. M. Mishra, A. Sahai, and R. W. Brodersen, "Cooperative sensing among cognitive radios," in *Proc. IEEE ICC'06*, (Istanbul, Turkey), pp. 1658 – 1663, IEEE, 11-15 June 2006.

[14] I. F. Akyildiz, W.-Y. Lee, M. C. Vuran, and S. Mohanty, "Next generation/dynamic spectrum access/cognitive radio wireless networks: A survey," *Computer Networks*, vol. 50, no. 13, pp. 2127–2159, 15 Sep. 2006.

[15] D. Maldonado, B. Le, A. Hugine, T. W. Rondeau, and C. W. Bostian, "Cognitive radio applications to dynamic spectrum allocation: a discussion and an illustrative example," in *Proc. IEEE DySPAN'05*, (Baltimore, MD), pp. 597–600, 8–11 Nov. 2005.

[16] S. M. Hasan, P. Balister, K. Lee, J. H. Reed, and S. W. Ellingson, "A low cost multi-band/multi-mode radio for public safety," in *SDR Forum Technical Conference*, (Orlando, FL), 13-17 Nov. 2006.

[17] P. Mähönen, M. Petrova, J. Riihijärvi, and M. Wellens, "Cognitive wireless networks: Your network just became a teenager," in *Poster in IEEE INFOCOM'06*, (Barcelona, Spain), IEEE, April 2006.

[18] Y. Wu and I. Niemegeers, "A cognitive architecture for personal networks," in *1st International IFIP TC6 Conference on Autonomic Networking (AN'06)*, (Paris, France), pp. 12 – 24, Springer-Verlag, Germany, 27-29 Sept. 2006.

[19] P. Demestichas, V. Stavroulaki, D. Boscovic, A. Lee, and J. Strassner, "m@angel: Autonomic management platform for seamless cognitive connectivity to the mobile internet," *IEEE Commun. Mag.*, vol. 44, no. 6, pp. 118–127, June 2006.

[20] P. D. Sutton, L. E. Doyle, and K. E. Nolan, "A reconfigurable platform for cognitive networks," in *Proc. of the 1st International Conference on Cognitive Radio Oriented Wireless Networks and Communications (CROWNCOM 2006)*, (Mykonos Island, Greece), pp. 1–5, IEEE, 8–10 June 2006.

[21] T. W. Rondeau, C. J. Rieser, B. Le, and C. W. Bostian, "Cognitive radios with genetic algorithms: Intelligent control of software defined radios," in *SDR Forum Technical Conference*, (Phoenix, AZ), pp. C–3–C–8, 15-18 Nov. 2004.

[22] K. E. Nolan, P. D. Sutton, and L. E. Doyle, "A framework for implementing cognitive functionality," in *SDR Forum Technical Conference*, (Orlando, FL), 13-17 Nov. 2006.

[23] D. Raychaudhuri, N. B. Mandayam, J. B. Evans, B. J. Ewy, S. Seshan, and P. Steenkiste, "Cognet: an architectural foundation for experimental cognitive radio networks within the future internet," in *Proceedings of first ACM/IEEE international workshop on Mobility in the evolving internet architecture (MobiArch'06)*, (San Francisco, California), pp. 11–16, ACM Press, Dec. 2006.

[24] P. Pawelczak, R. V. Prasad, L. Xia, and I. Niemegeers, "Cognitive radio emergency networks requirements and design," in *First IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN 2005)*, (Baltimore, Maryland), pp. 601 – 606, IEEE, 8-11 Nov. 2005.

[25] A. Peddemors, I. Niemegeers, H. Eertink, and J. D. Heer, "A system perspective on cognition for autonomic computing and communication," in *Proceedings Sixteenth International Workshop on Database and Expert Systems Applications (DEXA 2005)*, (Copenhagen, Denmark), pp. 181–185, IEEE Comput. Soc., 22-26 Aug. 2005.

[26] C. Fortuna and M. Mohorcic, "Trends in the development of communication networks: Cognitive networks," *Computer Networks*, vol. 53, no. 9, pp. 1354 – 1376, 2009.

[27] A. MacKenzie, J. Reed, P. Athanas, C. Bostian, R. Buehrer, L. DaSilva, S. Ellingson, Y. Hou, M. Hsiao, J.-M. Park, C. Patterson, S. Raman, and C. da Silva, "Cognitive radio and networking research at virginia tech," *Proceedings of the IEEE*, vol. 97, no. 4, pp. 660–688, Apr. 2009.

[28] L. P. Kaelbling, M. L. Littman, and A. P. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[29] 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.

[30] R. W. Thomas, *Cognitive Networks*. PhD thesis, Electrical and Computer Engineering Department, Virginia Tech University, Blacksburg, Virginia, 2007.

[31] D. H. Friend, M. Y. ElNainay, Y. Shi, and A. B. MacKenzie, "Architecture and performance of an island genetic algorithm-based cognitive network," in *Proc. IEEE CCNC'08*, (Las Vegas, NV), pp. 993–997, 10–12 Jan. 2008.

[32] E. Alba, *Parallel Metaheuristics: A New Class of Algorithms*. Hoboken, NJ: Wiley-Interscience, Sep. 2005.

[33] R. L. Haupt and S. E. Haupt, *Practical Genetic Algorithms*. Hoboken, NJ: Wiley-Interscience, second ed., May 2004.

[34] E. Alba and J. M. Troya, "A survey of parallel distributed genetic algorithms," *Complexity*, vol. 4, no. 4, pp. 31–52, 1999.

[35] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, May 1983.

[36] E. Onbasoglu and L. Ozdamar, "Parallel simulated annealing algorithms in global optimization," *Journal of Global Optimization*, vol. 19, no. 1, pp. 27–50, Jan. 2001.

[37] M. E. Aydin and T. C. Fogarty, "A distributed evolutionary simulated annealing algorithm for combinatorial optimisation problems," *Journal of Heuristics*, vol. 10, no. 3, pp. 269–292, May 2004.

[38] S.-Y. Lee and K. G. Lee, "Synchronous and asynchronous parallel simulated annealing with multiple markov chains," *IEEE Transactions on Parallel and Distributed Systems*, vol. 7, no. 10, pp. 993–1008, Oct. 1996.

[39] F. Glover, "Tabu search: A tutorial," *Interfaces*, vol. 20, no. 4, pp. 74–94, Jul. 1990.

[40] J. Dabrowski, "Parallelization techniques for tabu search," in *Applied Parallel Computing. State of the Art in Scientific Computing*, pp. 1126–1135, Springer, 2008.

[41] T. G. Crainic, "Parallel computation, co-operation, tabu search," in *Metaheuristic Optimization via Memory and Evolution*, pp. 283–302, Springer, 2005.

[42] M. Dorigo, *Optimization, Learning and Natural Algorithms*. PhD thesis, Politecnico di Milano, Italy, 1992.

[43] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 28–39, Nov. 2006.

[44] G. D. Caro, F. Ducatelle, and L. Gambardella, "AntHocNet: an adaptive nature-inspired algorithm for routing in mobile ad hoc networks," *European Transactions on Telecommunications*, vol. 16, no. 5, pp. 443–455, Sep.-Oct. 2005.

[45] K. M. Sim and W. H. Sun, "Ant colony optimization for routing and load-balancing: survey and new directions," *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, vol. 33, no. 5, pp. 560–572, Sep. 2003.

[46] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE International Conference on Neural Networks*, vol. 4, (Perth, WA, Australia), pp. 1942–1948, 27 Nov.-1 Dec. 1995.

[47] A. Banks, J. Vincent, and C. Anyakoha, "A review of particle swarm optimization. part i: background and development," *Natural Computing*, vol. 6, no. 4, pp. 467–484, 2007.

[48] J. F. Schutte, J. A. Reinbolt, B. J. Fregly, R. T. Haftka, and A. D. George, "Parallel global optimization with the particle swarm algorithm," *Journal of Numerical Methods in Engineering*, vol. 61, no. 13, pp. 2296–2315, 2003.

[49] B. il Koh, A. D. George, R. T. Haftka, and B. J. Fregly, "Parallel asynchronous particle swarm optimization," *Journal of Numerical Methods in Engineering*, vol. 67, no. 4, pp. 578–595, 2006.

[50] C. C. Ribeiro, S. L. Martins, and I. Rosseti, "Metaheuristics for optimization problems in computer communications," *Computer Communications*, vol. 30, no. 4, pp. 656–669, 2007.

[51] V.-D. Cung, S. L. Martins, C. C. Ribeiro, and C. Roucairol, "Strategies for the parallel implementation of metaheuristics," *Essays and Surveys in Metaheuristics*, pp. 263–308, 2002.

[52] T. Crainic and M. Toulouse, "Parallel strategies for meta-heuristics," in *Handbook of Metaheuristics*, pp. 475–513, Springer, Apr. 2003.

[53] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, April 1997.

[54] F. G. Lobo and D. E. Goldberg, "The parameter-less genetic algorithm in practice," *Information Sciences*, vol. 167, no. 1-4, pp. 217 – 232, 2004.

[55] I. Katzela and M. Naghshineh, "Channel assignment schemes for cellular mobile telecommunication systems: A comprehensive survey," *IEEE Personal Communications*, vol. 3, no. 3, pp. 10–31, June 1996.

[56] A. Mishra, S. Banerjee, and W. Arbaugh, "Weighted coloring based channel assignment for wlans," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 9, no. 3, pp. 19–31, 2005.

[57] Q. Zhao and B. Sadler, "A survey of dynamic spectrum access," *IEEE Signal Processing Magazine*, vol. 24, no. 3, pp. 79–89, May 2007.

[58] W. Wang and X. Liu, "List-coloring based channel allocation for open-spectrum wireless networks," in *Proc. IEEE VTC'05-Fall*, (Dallas, Texas), pp. 690–694, 25–28 Sept. 2005.

[59] H. Zheng and C. Peng, "Collaboration and fairness in opportunistic spectrum access," in *Proc. IEEE ICC'05*, (Seoul, Korea), pp. 3132–3136 Vol. 5, 16–20 May 2005.

[60] C. Li and C. Li, "Dynamic channel selection algorithm for cognitive radios," in *Proc. IEEE ICCSC'08*, (Shanghai, China), pp. 275–278, 26–28 May 2008.

[61] C. L. Barrett, G. Istrate, V. Kumar, M. V. Marathe, S. Thite, and S. Thulasidasan, "Strong edge coloring for channel assignment in wireless radio networks," in *Proc. IEEE PERCOMW'06*, (Pisa, Italy), pp. 110–114, 13–17 Mar. 2006.

[62] M. M. Halldórsson, J. Y. Halpern, L. E. Li, and V. S. Mirrokni, "On spectrum sharing games," in *Proc. ACM PODC'04*, (Newfoundland, Canada), pp. 107–114, ACM, 25–28 July 2004.

[63] N. Nie and C. Comaniciu, "Adaptive channel allocation spectrum etiquette for cognitive radio networks," *Mobile Networks and Applications*, vol. 11, no. 6, pp. 779–797, Dec. 2006.

[64] G. Chakraborty and B. Chakraborty, "A genetic algorithm approach to solve channel assignment problem in cellular radio networks," in *Proc. IEEE SMCia'99*, (Kuusamo, Finland), pp. 34–39, 16–18 June 1999.

[65] X. Fu, A. G. Bourgeois, P. Fan, and Y. Pan, "Using a genetic algorithm approach to solve the dynamic channel-assignment problem," *International Journal of Mobile Communications*, vol. 4, no. 3, pp. 333–353, 2006.

[66] S. Matsui, I. Watanabe, and K.-I. Tokoro, "Application of the parameter-free genetic algorithm to the fixed channel assignment problem," *Systems and Computers in Japan*, vol. 36, no. 4, pp. 71–81, 2005.

[67] P. Gupta and P. Kumar, "The capacity of wireless networks," *IEEE Transactions on Information Theory*, vol. 46, no. 2, pp. 388–404, Mar. 2000.

[68] J. Balakrishnan, A. Batra, and A. Dabak, "A multi-band ofdm system for uwb communication," in *Proc. IEEE Conf. on Ultra Wideband Systems and Technologies'03*, (Reston, VA), pp. 354–358, 16–19 Nov. 2003.

[69] K. Rasheed and H. Hirsh, "Using case-based learning to improve genetic-algorithm-based design optimization," in *Proc. of the 1997 International Conference on Genetic Algorithms (ICGA'97)*, (East Lansing, MI), pp. 513–520, 19–23 July 1997.

[70] M. Y. ElNainay, F. A. Ge, Y. Wang, A. Hilal, Y. S. Shi, A. MacKenzie, and C. Bostian, "Channel allocation for dynamic spectrum access cognitive networks using localized island genetic algorithm," in *Proc. TRIDENTCOM'09*, (Washington, D.C.), pp. 1–3, 6–8 Apr. 2009.

[71] GNU Radio. [Online]. Available: http://www.gnu.org/software/gnuradio/index.html.

[72] Ettus Research, LLC, "Universal Software Radio Peripheral." [Online]. Available: http://www.ettus.com.

[73] Q. Chen, Y. Wang, and C. W. Bostian, "Universal classifier synchronizer demodulator," in *Proc. IEEE IPCCC'08*, (Austin, TX), pp. 366–371, 7–9 Dec. 2008.

[74] T. Clausen, P. J. (editors), C. Adjih, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum, and L.Viennot, "Optimized link state routing protocol (OLSR)." RFC 3626, Oct. 2003. Network Working Group.

[75] A. Qayyum, L. Viennot, and A. Laouiti, "Multipoint relaying for flooding broadcast messages in mobile wireless networks," in *Proc. IEEE HICSS'02*, (Island of Hawaii), pp. 3866–3875, 7–10 Jan. 2002.

[76] M. Y. ElNainay, D. H. Friend, and A. B. MacKenzie, "Channel allocation & power control for dynamic spectrum cognitive networks using a localized island genetic algorithm," in *Proc. IEEE DySPAN'08*, (Chicago, IL), pp. 1–5, 14–17 Oct. 2008.

[77] M. Y. ElNainay and A. MacKenzie, "Effect of non-cooperation on dynamic spectrum cognitive networks," in *Proc. IWCMC'09*, (Leipzig, Germany), 21–24 Jun. 2009.

[78] M. Y. ElNainay, D. H. Friend, and A. B. MacKenzie, "Reasoning in a cultural algorithm-based cognitive network," *IEEE Transactions on Wireless Communications, Under Review*, 2009.

[79] A. T. Hoang and Y.-C. Liang, "Maximizing spectrum utilization of cognitive radio networks using channel allocation and power control," in *Proc. IEEE VTC'06-Fall*, (Montréal, Canada), pp. 1–5, 25–28 Sept. 2006.

[80] A. T. Hoang and Y.-C. Liang, "A two-phase channel and power allocation scheme for cognitive radio networks," in *Proc. IEEE PIMRC'06*, (Helsinki, Finland), pp. 1–5, 11–14 Sep. 2006.

[81] F. F. Digham, "Joint power and channel allocation for cognitive radios," in *Proc. IEEE WCNC'08*, (Las Vegas, NV), pp. 882–887, 31 Mar. – 3 Apr. 2008.

[82] M. Ma and D. H. Tsang, "Efficient spectrum sharing and power control in cognitive radio networks," in *Proc. ACM CWNets'07*, (Vancouver, British Columbia, Canada), 14 Aug. 2007.

[83] A. Behzard and I. Rubin, "Impact of power control on the performance of ad hoc wireless networks," in *Proc. IEEE INFOCOM'05*, (Miami, FL), pp. 102–113, 13–17 Mar. 2005.

[84] W. Huang and K. B. Letaief, "Cross-layer scheduling and power control combined with adaptive modulation for wireless ad hoc networks," in *Proc. IEEE GLOBECOM'05*, (St. Louis, MO), pp. 3180–3184, 28 Nov.–2 Dec. 2005.

[85] W. Huang and K. B. Letaief, "Cross-layer scheduling and power control combined with adaptive modulation for wireless ad hoc networks," *IEEE Transactions on Communications*, vol. 55, no. 4, pp. 728–739, Apr. 2007.

[86] C. M. Chin, M. L. Sim, and S. Olafsson, "A dynamic channel assignment strategy via power control for ad-hoc network systems," in *Proc. IEEE VTC'07-Spring*, (Dublin, Ireland), pp. 3006–3010, 22–25 Apr. 2007.

[87] Y. Shi and Y. T. Hou, "Optimal power control for multi-hop software defined radio networks," in *Proc. IEEE INFOCOM'07*, (Anchorage, AL), pp. 1694–1702, 6–12 May 2007.

[88] R. S. Komali, A. B. MacKenzie, and R. P. Gilles, "Effect of selfish node behavior on efficient topology design," *IEEE Transactions on Mobile Computing*, vol. 7, no. 9, pp. 1057–1070, Sept. 2008.

[89] R. S. Komali, *Game-Theoretic Analysis of Topology Control*. PhD thesis, Electrical and Computer Engineering Department, Virginia Tech University, Blacksburg, Virginia, 2008.

[90] S. Yokoyama, Y. Nakane, O. Takahashi, and E. Miyamoto, "Evaluation of the impact of selfish nodes in ad hoc networks and detection and countermeasure methods," in *Proc. IEEE MDM'06*, (Nara, Japan), pp. 95–100, 9–13 May 2006.

[91] M. Y. ElNainay and A. B. MacKenzie, "Joint flow routing, channel allocation, and power control: A cognitive network approach using k-hop island genetic algorithm," *IEEE Transactions on Mobile Computing, Under Review*, 2009.

[92] Y. Shi, Y. Hou, and S. Kompella, "A cross-layer approach to multi-hop networking with cognitive radios," in *Proc. IEEE MILCOM'08*, (San Diego, CA), pp. 1–7, 17–19 Nov. 2008.

[93] M. Ma and D. Tsang, "Joint spectrum sharing and fair routing in cognitive radio networks," in *Proc. IEEE CCNC'08*, (Las Vegas, NV), pp. 978–982, 10–12 Jan. 2008.

[94] G. Cheng, W. Liu, Y. Li, and W. Cheng, "Spectrum aware on-demand routing in cognitive radio networks," in *Proc. IEEE DySPAN'07*, (Dublin, Ireland), pp. 571–574, 17–20 Apr. 2007.

[95] G. Cheng, W. Liu, Y. Li, and W. Cheng, "Joint on-demand routing and spectrum assignment in cognitive radio networks," in *Proc. IEEE ICC'07*, (Glasgow, Scotland), pp. 6499–6503, 24–28 June 2007.

[96] H. Ma, L. Zheng, X. Ma, and Y. luo, "Spectrum aware routing for multi-hop cognitive radio networks with a single transceiver," in *Proc. CrownCom'08*, (Singapore), pp. 1–6, 15–17 May 2008.

[97] Z. Yang, G. Cheng, W. Liu, W. Yuan, and W. Cheng, "Local coordination based routing and spectrum assignment in multi-hop cognitive radio networks," *Mobile Networks and Applications*, vol. 13, no. 1, pp. 67–81, Apr. 2008.

[98] B. Vaidya, D.-Y. Choi, J. Park, and S. Han, "Multipath routing scheme for wireless multihop network," in *Lecture Notes in Computer Science*, (Germany), pp. 433–445, Springer-Verlag, 2008.

[99] X. Li and L. Cuthbert, "Stable node-disjoint multipath routing with low overhead in mobile ad hoc networks," in *Proc. IEEE MASCOTS'04*, (Volendam, The Netherlands), pp. 184–191, 4–8 Oct. 2004.

[100] D. Bertsekas and R. Gallager, *Data Networks*. Prentice Hall, second ed., 1992.

[101] X. Liu and W. Wang, "On the characteristics of spectrum-agile communication networks," in *Proc. IEEE DySPAN'05*, (Baltimore, MD), pp. 214–223, 8–11 Nov. 2005.

[102] R. G. Reynolds and B. Peng, "Cultural algorithms: Modeling of how cultures learn to solve problems," in *Proc. IEEE ICTAI'04*, (Boca Raton, FL), pp. 166–172, 15–17 Nov. 2004.

[103] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar, "An integrated experimental environment for distributed systems and networks," in *OSDI02*, (Boston, MA), pp. 255–270, USENIXASSOC, Dec. 2002.

[104] ORBIT. [Online]. Available: http://www.orbit-lab.org/.

[105] F. Herrera, E. Herrera-Viedma, M. Lozano, and J. Verdegay, "Fuzzy tools to improve genetic algorithms," in *Proc. Second European Congress on Intelligent Techniques and Soft Computing*, (Aachen, Germany), pp. 1532–1539, 20–23 Sep. 1994.

[106] J. Cooper and C. J. Hinde, "Improving genetic algorithms' efficiency using intelligent fitness functions," in *IEA/AIE'03*, (Loughborough, UK), pp. 636–643, 23–26 June 2003.

[107] T. W. Rondeau, *Application of Artificial Intelligence to Wireless Communications*. PhD thesis, Electrical and Computer Engineering Department, Virginia Tech University, Blacksburg, Virginia, 2007.

[108] J. Collins and D. Joslin, "Improving genetic algorithm performance with intelligent mappings from chromosomes to solutions," in *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, (Seattle, Washington, USA), pp. 1407–1408, 2006.

[109] A.-H. A. Liou, T.-H. Chi, and I.-J. Yu, "Improving genetic algorithms with solution space partitioning and evolution refinements," in *Proc. IEEE ICNC'07*, vol. 4, (Haikou, China), pp. 238–242, 24–27 Aug. 2007.

[110] R. G. Reynolds, "An introduction to cultural algorithms," in *Proc. of the 3rd Annual Conference on Evolutionary Programming*, (River Edge, NJ), pp. 131–139, 1994.

[111] C. A. C. Coello and R. L. Becerra, "A cultural algorithm for constrained optimization," in *MICAI '02: Proceedings of the Second Mexican International Conference on Artificial Intelligence*, (London, UK), pp. 98–107, Springer-Verlag, 2002.

[112] C. A. C. Coello and R. L. Becerra, "Adding knowledge and efficient data structures to evolutionary programming: A cultural algorithm for constrained optimization," in *Proc. GECCO'02*, pp. 201–209, 2002.

[113] F. Gao, G. Cui, and H. Liu, "Integration of genetic algorithm and cultural algorithms for constrained optimization," in *Lecture Notes in Computer Science*, (Germany), pp. 817–825, Springer-Verlag, 2006.

[114] B. Ping and R. G. Reynolds, "Cultural algorithms: Knowledge learning in dynamic environment," in *Proc. IEEE CEC2004*, vol. 2, (San Diego, CA), pp. 1751–1758, 19–23 June 2004.

[115] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. The MIT Press, Mar. 1998.

[116] R. G. Reynolds and S. Zhu, "Knowledge-based function optimization using fuzzy cultural algorithms with evolutionary programming," *IEEE Transactions on Systems, Man, And Cybernetics, Part B*, vol. 31, no. 1, pp. 1–18, Feb. 2001.

[117] T. Mitchell, *Version Spaces: An Approach to Concept Learning*. PhD thesis, Computer Science Department, Stanford University, Stanford, California, 1978.

[118] N. Baldo and M. Zorzi, "Learning and adaptation in cognitive radios using neural networks," in *Proc. IEEE CCNC'08*, (Las Vegas, NV), pp. 998–1003, 10–12 Jan. 2008.