OPTIMAL SPECTRUM ALLOCATION TO SUPPORT TACTICAL MOBILE AD-HOC NETWORKS

by

Paul J. Nicholas
A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
In Partial Fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Systems Engineering and Operations Research

Committee:

_____	Dr. Karla L. Hoffman, Dissertation Director
_____	Dr. Ariela Sofer, Committee Member
_____	Dr. Andrew G. Loerch, Committee Member
_____	Dr. Robert P. Simon, Committee Member
_____	Dr. Ariela Sofer, Department Chair
_____	Dr. Stephen G. Nash, Dean, Volgenau School of Engineering

Date: _____ Fall Semester 2016
George Mason University
Fairfax, VA

Optimal Spectrum Allocation to Support Tactical Mobile Ad-hoc Networks

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

By

Paul J. Nicholas
Master of Science
Naval Postgraduate School, 2009
Bachelor of Science
United States Naval Academy, 2003

Director: Dr. Karla L. Hoffman, Professor
Department of Systems Engineering and Operations Research

Fall Semester 2016
George Mason University
Fairfax, VA

Dedication

To Melissa, Abigail, and Elizabeth.

Acknowledgments

I owe a debt of gratitude to Professor Karla Hoffman for her patient mentorship over the last few years. Learning about and then trying (poorly) to mimic your approach to problem-solving will have a lasting effect on the rest of my career. In fact, "What would Karla do?" (WWKD) has become the first question I ask myself when faced with a new analytic challenge. Thank you for your help in developing this new ability to think.

My dissertation committee, including Professors Ariela Sofer, Andy Loerch, and Bob Simon, provided support and guidance during this process. Thank you very much for your time and input. Professors Rajesh Ganesan, Sanjeev Setia, and Fei Li all generously gave their time to help me. Dr. Don Wagner at the Office of Naval Research provided crucial support of this research, including support for time, tuition, travel, and computational resources. Thank you for your confidence in us.

I am deeply indebted to my good friend and mentor, Professor David Alderson at the Naval Postgraduate School. Beginning almost ten years ago, it was your kind encouragement and frank advice that sent me and kept me on a path to deepen my knowledge of operations research. I hope to follow your example as a guide and role model in the professional development of future OR practitioners.

The exceptionally competent and friendly staff at George Mason University have made this task much easier to complete. I am indebted to Angel Manzo, Josefine Wiecks, Jonathan Goldman, Sally Evans, and numerous individuals in the library and Registrar's Office.

My friends and colleagues at Operations Analysis Directorate have played key roles in enabling me to complete this work. Al Sawyers and Mike Bailey encouraged me to begin the program, and provided the freedom and latitude to do so. George Akst provided support via the Marine Corps Academic Degree Tuition Assistance Program. Jeff Tkacheff and Lorri Flint provided technical support during early computational experiments. Thanks for providing such a supportive work environment.

This research is a direct result of the MAGTF Wideband Spectrum Requirement and Allocation Study, with which I am honored to have been involved. Josh Pepper, Max Hipsher, and Pete Bulanow all played key roles in developing the scenarios that provided the testbed for this research. John Pico, Bill Kloth, Dennis Murphy, Mike Bell, Art DeLeon, and Jerome Foreman provided invaluable input as subject matter experts.

Over the last few years I've benefited greatly from friends and colleagues who have shown (or artfully feigned) interest in my work and have provided support ranging from

tips and words of encouragement, to detailed and constructive feedback. A necessarily incomplete list includes Rob Dell, Jerry Brown, Matt Carlyle, Cynthia Irvine, Wade Huntley, Matt Aylward, Wayne Breakfield, Joe Monahan, Chris Fitzpatrick, Shane Price, Steve Charbonneau, Ray Trechter, and Scott Laprise.

I would be remiss if I didn't mention my faithful stainless steel friend, a Gaggia Classic home espresso machine. Your caffeine-infused nectar has powered me through many otherwise soporific and unproductive evenings. May you spout velvety lattes for years to come.

My mom, Margaret, is perhaps my most indefatigable supporter. Thank you for your love and encouragement. You knew I could.

My wife Melissa and daughters Abigail and Elizabeth have shouldered additional burdens while I've neglected many of my duties as husband and father in order to pursue this degree. You bore this without complaint, taking care of each other and taking care of me over the last three years. You kept me fed, clothed, and hydrated when I chained myself to my laptop computer for days on end; your smiles and squirrelly antics held my spirits high when I needed to push through. I've accomplished this only because of you, but it has come at the cost of so many missed bedtime stories, playtime without me, date nights untaken and vacations interrupted, and walks through the park that we never took. Thank you for your love, patience, and sacrifice. I'm all yours.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ANW2 ..... Adaptive Networking Wideband Waveform
C2 ..... command and control
CALMA ..... Combinatorial Algorithms for Military Applications
CAP ..... channel assignment problem
CP ..... constraint programming
CPU ..... central processing unit
dB ..... decibel
DSA ..... dynamic spectrum access
EM ..... electromagnetic
EUCLID ..... European Cooperation on the Long Term in Defense
FCC ..... Federal Communications Commission
FSF ..... full standard formulation
GAMS ..... General Algebraic Modeling System
GB ..... gigabyte
GHz ..... gigahertz
Hz ..... Hertz
kHz ..... kilohertz
km ..... kilometer
IP ..... integer programming
ITM ..... Irregular Terrain Model
MAGTF ..... Marine Air-Ground Task Force
MANET ..... mobile ad-hoc network
MB ..... megabyte
MC-CAP-T ..... minimum-cost channel assignment problem over time
MEB ..... Marine Expeditionary Brigade
MEF ..... Marine Expeditionary Force
MEU ..... Marine Expeditionary Unit
MHz ..... megahertz
MI-CAP ..... minimum-interference channel assignment problem
MIP ..... mixed integer program
MO-CAP ..... minimum-order channel assignment problem
OPL ..... Optimization Programming Language
RAM ..... random access memory
RSF ..... restricted standard formulation
RSS ..... received signal strength
SATCOM ..... satellite communications
SCR ..... single-channel radio
SIR ..... signal-to-interference ratio
SPEED ..... Systems Planning, Engineering, and Evaluation Device
STK ..... Systems Toolkit
TIREM ..... Terrain Integrated Rough Earth Model
USMC ..... United States Marine Corps
WLAN ..... wireless local area network

Abstract

OPTIMAL SPECTRUM ALLOCATION TO SUPPORT TACTICAL MOBILE AD-HOC NETWORKS

Paul J. Nicholas, PhD

George Mason University, 2016

Dissertation Director: Dr. Karla L. Hoffman

Current military forces are developing, purchasing, and fielding tactical wideband radios capable of connecting wirelessly to each other to form a mobile ad-hoc network (MANET), an autonomous communications system where each wideband radio serves as a mobile node. Wideband MANET radios offer tremendous new capabilities, including high data rates and automatic traffic relay, but have large electromagnetic spectrum requirements. Meanwhile, wireless traffic from civilian, joint, and coalition networks will increasingly clutter the electromagnetic spectrum, and militaries will continue to operate in environments with restrictions on spectrum use. Efficient allocation of available spectrum is required to ensure military forces are able to fully utilize new MANET radios, yet current methods of allocation are woefully inadequate.

We consider three challenges faced by a military spectrum manager in supporting MANET communications for mobile units operating on rough terrain. First, prior to the commencement of a military operation, the spectrum manager must determine the minimum number of channels required to support communications with an acceptable level of interference. Second, during ongoing operations, the spectrum manager may have only a fixed and insufficient amount of spectrum available, and must now assign channels to minimize total received interference. Finally, having solved either of these two problems to determine channel assignments at discrete moments of time, the spectrum manager must also consider the number of channel changes over time, as each radio requires manual channel assignment.

Efficient channel allocation schemes leverage channel reuse, and the ability to reuse a channel is dependent on co-channel interference, i.e., interference occurring between radios using the same channel but not communicating on the same network. A simplified version of this problem is the graph-coloring problem, which restricts any two adjacent radios from being assigned the same channel. This seemingly straightforward problem is NP-complete, and yet realistic interference constraints — needed to most efficiently use available spectrum — are much more complex. Namely, one must consider the cumulative effect of multiple sources of interference, rather than just interference between pairs of radios. This greatly increases the computational difficulty of quickly finding good channel allocation schemes, and thus the vast majority of research considers only interference between pairs of radios.

We use heuristic, integer optimization, and constraint programming approaches to develop more efficient methods of channel allocation considering cumulative co-channel interference. We formulate and solve several integer optimization and constraint programming problems to minimize the number of required channels, minimize total received interference, and minimize the number of channel changes over time. We test our methods using radio performance data generated by high-fidelity simulation in the context of realistic, large-scale U.S. Marine Corps combat scenarios.

Our research provides fast methods of calculating realistic and efficient channel allocation schemes that reduce the number of required channels, reduce cumulative co-channel interference, and reduce the total number of required channel changes over time. Our methodologies are applicable to any type of radio or electromagnetic transmission device requiring discrete assignments from a fixed pool of available channels, including radios, radars, jamming devices, and sensors. To our knowledge, we are the first to use integer optimization and constraint programming methods to model and solve full-size instances of the channel assignment problem to global or near-global optimality, while also using a realistic interference model and considering cumulative interference constraints.

Chapter 1: Introduction

1.1 Background

The United States military fields many different types of radios and other wireless systems that require vast swathes of electromagnetic (EM) spectrum, including traditional point-to-point single channel radios (SCRs), wideband radios, radars, jammers, satellite communications (SATCOM) radios, and control and data links for unmanned aerial vehicles. These wireless systems offer tremendous capabilities, including high data transmission rates (in the case of communications devices) and high-fidelity portrayals of the operating environment (in the case of radars and other sensors). However, in general, the larger the amount of transmitted information, the more EM spectrum (i.e., *bandwidth*) is required.

The U.S. military is currently developing, purchasing, and fielding *tactical wideband radios* capable of connecting highly mobile units operating in rugged terrain over long distances with relatively low-power radios (Goulding 2009). These wideband radios connect wirelessly to each other to form a *mobile ad-hoc network* (*MANET*), an autonomous communications system where each wideband radio serves as a *mobile node*. These nodes may move and connect in wireless, dynamic, multi-hop topologies, and exhibit self-learning, self-healing behavior (Corson and Macker 1998, Aggelou 2004), i.e., individual radios may automatically connect and disconnect from a MANET without any user interaction. A MANET comprises physical radios and the associated networking protocols, waveforms, and modulation schemes required to route traffic. Each wideband radio in a MANET is a terminal device for voice or digital communications, and may concurrently serve as a relay device for other radios in the network. In this way, MANETs are similar to client-mesh *wireless mesh networks* (*WMNs*) (Zhang et al. 2006), where client devices perform routing functions.

Wideband MANET radios offer tremendous new capabilities, including high data rates and automatic traffic relay, but have large electromagnetic spectrum requirements. Technological limits constrain the number of wideband radios that can be assigned the same channel (i.e., a contiguous portion of spectrum), and the channels used by wideband radios are larger than that used by legacy *narrowband* radios. For instance, 1.2 MHz wideband channels occupy 48 times more spectrum than 25 kHz narrowband channels used for voice-only communications. The introduction of this new wideband capability will challenge the status quo for spectrum allocation, and future communications planning must balance the requirements of new wideband networks and greater data capabilities with less-capable, legacy narrowband networks that require less spectrum.

Meanwhile, the U.S. military will continue to operate in environments with increasing restrictions on spectrum use, both in the U.S. and abroad. Wireless communications traffic from civilian, joint, and coalition networks will increasingly clutter the EM spectrum, and the Federal Communications Commission (FCC) is moving the military to different bands to share spectrum with the private sector (Goldstein 2013, Selyukh 2013). Efficient allocation of available spectrum is required to ensure military forces are able to fully utilize new tactical wideband radio assets (Stine and Portigal 2004), yet current methods of allocation are woefully inadequate. Indeed, in a major study the U.S. Marine Corps (USMC) finds that with current allocation methods, Marine Air-Ground Task Forces (MAGTFs) will not have enough spectrum available to support the use of wideband MANET radios in major combat operations (Nicholas et al. 2013a).

Various forms of the *channel assignment problem* (*CAP*) aim to optimally allocate spectrum in a given moment of time. To solve realistic problem instances, one must consider not only the scarcity of available spectrum, but also the technological limitations of the radios being supported. Specifically, the performance of a wideband MANET radio (i.e., the data rate) depends greatly on the amount of *interference* it receives (Gupta and Kumar 2000). The interference can be naturally occurring (such as solar radiation), intentional (such as jamming), or unintentional (such as from other nearby radios operating on the

same, adjacent, or harmonically-adjacent frequencies). Other technological limitations include transmission power and antenna and processing gains at each radio. Environmental factors include *free space path loss* (i.e., the loss in EM energy due to geometric spreading over distance) and absorption due to terrain and the atmosphere.

Efficient channels allocation schemes leverage *channel reuse*. The ability to reuse a channel is dependent on (among other things) *co-channel interference*, i.e., interference occurring between two radios using the same channel but not communicating on the same network. A simplified version of this problem is the *vertex* or *graph-coloring problem*, which restricts any two adjacent nodes (i.e., radios) from being assigned the same color (i.e., channel). This seemingly simple problem is *NP-complete* (Skiena 1990, Cuppini 1994), and yet realistic interference constraints are much more complex. Namely, they must consider the cumulative effect of multiple sources of interference, rather than just interference between pairs of radios. This greatly increases the computational difficulty of quickly finding good channel allocations.

Several characteristics of tactical military data communications make the CAP more difficult to solve than for typical civilian applications. For example, in radio or television broadcast there are relatively few transmission towers and many nodes functioning only as receivers, whereas MANET radios function as both transmitters and receivers. Also, MANET radios may be on the move. They cannot benefit from specially-tuned transmission antennae, and instead use omnidirectional antennae that reduce their ability to project power in desired directions and increase their production of and susceptibility to interference. Though mobile phone applications consider mobility, the physical laydown of military radios may be denser relative to the transmission power of each radio. The radios we consider transmit from five to 50 watts, whereas most mobile phone handsets are limited to three watts (Muller 2003) and cellular transmission towers to an effective five to ten watts (Federal Communications Commission 2014). Further, the wideband radios we consider occupy large bandwidths (each channel occupies 1.2 to 5 MHz). These factors decrease the ability to reuse channels, even if the associated CAP is solved to optimality.

The CAP becomes even more complicated when one considers the cost over time of manually changing channels on each radio. *Cognitive radio technology* provides many benefits, but in general its ability to dynamically allocate spectrum is premised on the assumption that channels can be changed automatically and hence without cost (namely, manual configuration time). Many military EM systems, including single-channel radio (SCR), radar, jammers, and the tactical MANETs we consider, are not interconnected because of security concerns and costs of additional complexity and communications overhead. These systems thus cannot automatically sense current spectrum utilization, nor change channels dynamically. They require centralized channel assignments from a *spectrum manager* (who must follow specific procedures based on overall operations and policies imposed by friendly and host nations), and must be manually configured to use a particular channel by a human operator, thus incurring a time cost.

Current software tools to assist in channel allocation, including the Systems Planning, Engineering, and Evaluation Device (SPEED) (Lamar 2013) and Spectrum XXI (Defense Information Systems Agency 2013), provide radio coverage analysis reports, and the latter tool provides a database to deconflict assignments across a given operating area. However, neither consider interference among a large number of mobile transmitters over multiple time periods, nor do they provide a rigorous method for minimizing the number of required channels.

1.2 Problem Statement

We consider the problem of a military spectrum manager who must determine an efficient channel allocation scheme to support radio communications during a certain period of time for mobile units operating on rough terrain. The spectrum manager knows the capabilities of each radio and their starting locations, has a rough understanding of their future locations within the geographic *operating area*, and has access to the underlying terrain elevation data.

Our spectrum manager faces three primary challenges. First, prior to the commencement of operations, the spectrum manager must determine the minimum number of channels required to support communications with an acceptable level of co-channel interference. Solving the associated *minimum-order channel assignment problem* (*MO-CAP*) and identifying this requirement – specific to the local military force and operating area – informs larger spectrum management operations involving other military forces and adjacent operating areas. This may be particularly useful prior to the start of operations when a higher headquarters is allocating available spectrum and needs to know requirements within each operating area.

The second challenge faced by our spectrum manager occurs during operations when the allocated spectrum available for assignment to each MANET is less than that indicated by the solution to the MO-CAP. That is, the spectrum manager must now make do with the available spectrum. Following the seminal work of Gupta and Kumar (2000), we use interference as a proxy for overall wireless network performance, and so we wish to minimize total received interference in order to maximize network performance. We model this via the *minimum-interference channel assignment problem* (*MI-CAP*).

Finally, having solved either the MO-CAP or MI-CAP to determine channel assignments at discrete moments (or *time steps*) within the operation, the spectrum manager must also consider the number of channel changes over time because each radio requires manual channel assignment and cannot change channels automatically. We model this via our *minimum-cost channel assignment problem over time* (*MC-CAP-T*). A *myopic* solution considers channel assignments only during a particular moment in time. Such a solution may needlessly "flip-flop" channel assignments over time, and may be particularly fragile to changes in network physical topologies. By leveraging available information on the future locations of radios, we can quickly provide a more far-sighted solution that aims to reduce the number of required channel changes over time. This decreases the time used by operators to manually adjust radio configurations, and the time needed by the spectrum

manager to de-conflict unexpected interference, thus improving overall network availability and performance.

In all cases, we assume the spectrum manager has a few hours to perhaps several days to determine the best allocation using local computing resources (including one or more computers with multiple cores). Once the allocation is determined using this centralized approach, the channel assignments are distributed to each radio operator for manual configuration.

We consider a specific type of radio system (i.e., multiple, independent MANETs) operating within a particular EM band, i.e., we do not consider the use of other radio transmission systems such as radar or other communications radios in the same band. However, our approach applies to any type of EM transmission system requiring a channel assignment within a particular band. This includes traditional point-to-point single-channel radios, point-to-point wideband transmission systems, radar systems, and jamming equipment. In reality, EM bands will be segmented for use by different systems in this way, so this approach is realistic.

1.3 Research Objective and Approach

The objective of this research is to identify and develop formulations and methodologies for solving realistic, full-sized instances of the mobility-aware channel assignment problem in a reasonable amount of time to provide a more efficient method of allocating channels for military communications.

We use heuristic, integer optimization, and constraint programming methods to develop more efficient methods of military channel allocation. We formulate several *integer programs* (*IPs*) and *constraint programming* (*CP*) problems to minimize the number of required channels subject to cumulative co=channel interference constraints, minimize total received interference, and minimize the number of channel changes over time.

We examine the cumulative interference constraints and consider methods of addressing their computational difficulties. We explore the use of heuristics to preprocess input data

and reduce the complexity of each problem, and the use of exact methods to solve each IP and CP problem to global or near-global optimality. Crucially, we consider the far-sighted assignment of channels over time and not just multiple, independent "snap-shots" in time.

We use realistic radio performance data from high-fidelity simulations of U.S. Marine Corps combat scenarios. The radio propagation engine, the Terrain Integrated Rough Earth Model (TIREM) (Alion Science and Technology Corporation 2016), is the de facto standard government model for calculating radio propagation over terrain and through the atmosphere. Our scenarios model both steady-state operations (i.e., irregular warfare or peacekeeping operations) and major combat operations (i.e., large amphibious assaults). The data sets range in complexity from just a few dozen radios to nearly 2,000 radios. Our computational experiments consider only the largest scenario, since (as we demonstrate) the others are trivial.

1.4 Research Contribution

Our research provides fast methods of calculating realistic and efficient channel allocation schemes that reduce the number of required channels, reduce total co-channel interference, and/or reduce the total number of required channel changes over time for tactical military wideband radio systems. This methodology is applicable to any type of radio or EM transmission device requiring discrete assignments from a fixed pool of available channels, including radios, radars, jamming devices, and certain types of sensors.

The vast majority of previous research on exactly solving the channel assignment problem ignores cumulative co-channel interference, instead modeling only pairwise interference. Most cognitive radio research considers cumulative interference but assumes channel changes are automated and essentially costless (in our case, a time cost is incurred). The only paper that considers the use of *temporal graphs* to minimize the number of required channels over time considers only pairwise interference (Yu et al. 2013). Such simplifications greatly

reduce computational load and may increase model tractability, but come at the price of reduced model fidelity.

Further, most research on solving realistic instances of the channel assignment problem use heuristics, which may quickly provide feasible solutions but in general fail to provide any sort of *certificate of optimality*. We use exact optimization and constraint programming techniques to provide bounds on the goodness of a solution.

To our knowledge, there has been no research that uses integer optimization and constraint programming methods to model and solve full-size instances of the channel assignment problem to global or near-global optimality, while also using a realistic interference model and considering cumulative interference constraints.

1.5 Document Structure

This document is structured as follows. Chapter 2 comprises a literature review. Chapter 3 describes our model of MANET communications, and provides detail on our datasets. The next three chapters each describe a particular type of channel assignment problem of interest to our spectrum manager. Each chapter includes various formulations and descriptions of their computational challenges, and our solution methods and results. Chapter 4 considers the minimum-order CAP, which is relevant to a spectrum manager conducting planning in advance of an operation and who wishes to determine the minimum number of required channels. Chapter 5 considers the minimum-interference problem, which is relevant to a spectrum manager immediately before and during an operation, when a fixed number of channels have been allocated and the manager must make do with available spectrum. Chapter 6 considers the minimum-cost CAP, wherein the spectrum manager uses the results from either MO-CAP or MI-CAP and attempts to reduce the total number of required channel changes over time. Chapter 7 provides our conclusions and recommendations for future research.

Chapter 2: Literature Review

2.1 The Channel Assignment Problem (CAP)

The CAP is a well-researched problem, and interest has been growing rapidly with the spread of wireless telephony (including both voice and data networks) and satellite communications (Aardal et al. 2007). Hale (1980) wrote a landmark paper on the *frequency assignment problem*. He differentiates the frequency assignment problem (where assigned frequencies may be non-contiguous) from the channel assignment problem (where assigned frequencies are in a contiguous block). Note this terminology is not consistent in the literature, and these terms are often interchangeable. (In the present research, we consider only the channel assignment problem.) Hale (1980) recognizes two possible figures of merit for this family of problems: *span* (the total range of frequencies assigned) and *order* (the total number of channels), which we consider.

Metzger (1970) is usually credited with first observing the possibility of using optimization techniques for solving channel assignment problems. He describes several heuristic methods to make sequential channel assignments. A *frequency exhaustive* method attempts to assign the lowest available frequency. A *uniform* method attempts to use that frequency which has been used the least. A *requirement exhaustive* method attempts to use each frequency in order.

Metzger (1970) compares the CAP to the *vertex* or *graph coloring problem* (Gould 1988), where any two adjacent vertices (i.e., radios) may not be colored the same color (i.e., channel). In the CAP, this is analogous to ensuring two particular radios are not assigned the same channel in order to prevent interference. The problem is easy to explain, yet it is proven to be *NP-complete* (Skiena 1990, Cuppini 1994). The cumulative co-channel

interference problem which we consider is more complex because we disallow certain n-tuples of radios from being assigned the same channel, potentially of much higher order than just pairs.

Murphey et al. (1999) observe that though there is extensive research into the channel assignment problem, it remains a notoriously difficult problem to solve. They state that due to the complexity of the problem, real-world practitioners often rely on *sequential* methods that assign a channel to one radio or network at a time. They contrast these methods to the exact methods based on the graph coloring problem.

Aardal et al. (2007) provide the seminal survey of contemporary research into the models and solution methods for the CAP, focused primarily on the practical aspects of mathematical optimization. They differentiate between *dynamic* channel assignment problems (where channel assignments may vary over time) and the *fixed* channel assignment problems; they consider only fixed channel assignment. They provide a basic formulation for a CAP (which we build upon), including an objective, assignment constraints, and interference constraints. They state interference constraints are usually represented as an *interference graph*, where an arc represents an unallowable combination. With only pairwise constraints, this reduces to a *binary constraint satisfaction problem*. They describe several different types of objective functions, including *maximum service* (assign as many channels as possible to each node), *minimum blocking* (minimize the number of blocked calls in a phone network), *minimum span* (minimize the total range of spectrum needed to support operations), *minimum interference*, and *minimum order* (i.e., minimize the total number of required channels), the latter two of which we consider.

Aardal et al. (2007) describe four different types of CAP constraints. *Co-cell separation* constraints ensure channels being used by the same antenna must differ in frequency by a given amount. *Co-site separation* constraints ensure channels used at the same physical site must differ by a given amount. *Interference* constraints (which we consider) ensure radios using the same or spectrally-adjacent channels do not provide unacceptable interference.

Hand-over separation constraints ensure that when a mobile radio moves from one service cell to another, the channels must differ by a given *frequency distance*.

Aardal et al. (2007) describe several CAP test sets, including Philadelphia (Anderson 1973), the COST 259 project (COoperation Européenne le Domaine de la Recherche Scientifique et Technique) (Correia 2001), the EUCLID CALMA project (see Aardal et al. (2002) for overview), and the CELAR instances which consider multiple interference (used by Palpant et al. (2008), Sarzeaud and Berny (2003), Dupont et al. (2005)). Each of these datasets provides pre-calculated interference constraints, whereas we must discover our interference constraints from raw data generated by our combat simulations.

We differentiate our approach from the classic fixed CAP of Aardal et al. (2007) and the dynamic CAP of Katzela and Naghshineh (1996) as follows. Unlike the fixed CAP, we consider channel changes over time and look at these changes holistically, instead of just successive "snapshots" in time. Unlike the dynamic CAP, we do not assume there is a fixed pool of available channels that are used to meet changing demand, but rather try to find this total minimal number over the time period of interest.

2.2 Computational Challenges of Cumulative Interference

The vast majority of exact optimization work on the CAP considers only pairwise interference constraints (Aardal et al. 2007). This is due to the computational challenges of explicitly representing cumulative interference, and the ease with which the problem can be represented as a graph coloring problem when considering only pairwise constraints (Berry 1990, Wang and Rappaport 1989, Dunkin et al. 1998, Nicholas and Hoffman 2015).

Dunkin et al. (1998) describe the computational challenge of using cumulative interference constraints, and instead use simple binary and tertiary constraints (e.g., groups of three interfering radios) using a constraint satisfaction approach. Daniels et al. (2004) formulate an integer minimum-order CAP that considers cumulative interference (unlike the related work of Murphey et al. (1999)) and establish the NP-hardness of the problem.

Their centralized dynamic channel assignment heuristic provides solution values within 3% of those obtained via CPLEX. They note the impact of cumulative interference in their artificial test data is quite small, unlike the cumulative interference we observe in our realistic test data.

Fischetti et al. (2000) use pre-processing and *branch-and-cut* to solve their cumulative interference CAP. They use the *Big M* technique to avoid nonlinearity in their integer formulation, and tune their Big M value to improve convergence performance of the integrality-relaxed problem. They solve a number of real-world problem instances in a reasonable amount of time, but their problem sizes are much smaller than ours and consider relatively few sources of interference.

Palpant et al. (2008), Sarzeaud and Berny (2003), and Dupont et al. (2005) all consider cumulative interference using *integer programming* formulations to solve the minimum interference and minimum span problems. Palpant et al. (2008) note their IP formulations perform badly because of the huge number of variables and the symmetry of the problem, problems which we also detect in our test data. They show that using cumulative interference constraints provides a much larger feasible region than simply replacing all cumulative constraints with binary constraints.

Other papers that consider cumulative interference include Alouf et al. (2005), who use heuristic and exact optimization techniques to allocate spectrum for satellites, and Garcia Villegas et al. (2005), who use a distributed heuristic to minimize interference for WiFi networks.

2.3 CAP Solution Methods

In the following sections, we categorize previous research for solving CAPs into two broad groups, *exact methods* and *heuristic methods*, provide a brief overview of related *dynamic spectrum access* research, and then describe previous research using two approaches (specifically, temporal graphs and parallel and distributed computation) that we leverage to solve our large-scale, mobility-aware CAPs.

Exact methods use mathematical optimization techniques to find optimal solutions, or solutions whose goodness (i.e., distance to optimality) can be calculated. These methods are useful when there is sufficient time and computational power available to find an optimal or near-optimal solution, e.g., for small communications networks, when designing fixed communications infrastructures (like cellular phone and television towers), and during deliberate military planning. *Heuristic methods* are used when such resources are not available, or exact solutions are not required, e.g., when using radio systems that can automatically change their own channels based on environmental conditions. In general, heuristics provide solutions quickly but with no certification of the *optimality gap* of any particular solution. Heuristics are the most common method used in real-world channel allocation algorithms, and can be useful for pre-processing input data prior to solving using exact methods.

2.3.1 Exact Solution Methods

Exact solution methods are the most relevant to our minimum-order CAP because we assume that the spectrum manager has sufficient time and computational resources to find certifiably-good solutions, and has incentive to do so because of the spectrum scarcity and the time cost of changing channels. However, as Garcia Villegas et al. (2005) note, great computational power is required to solve real-world CAP problems to optimality. We explore the use of exact optimization and constraint programming techniques to provide a bound to the goodness of our solutions.

The CAP naturally lends itself to an integer programming (IP) formulation, and the vast majority of the literature on exact solution methods for CAP use an IP formulation, including Aardal et al. (2007), Fischetti et al. (2000), Mannino and Sassano (2003), Daniels et al. (2004), and Palpant et al. (2008). Our IP formulation follows that of Aardal et al. (2007); other formulations include *column generation* (see, e.g., Mehrotra and Trick (1996) and Jaumard et al. (2002)) and the *orientation* formulation of Borndörfer et al. (1998). The most common exact solution methods are variations of *combinatorial tree search*, including *branch-and-bound*, *branch-and-cut*, and *implicit enumeration* (see, e.g., Aardal et al. (1996,

2007), Fischetti et al. (2000), Mannino and Sassano (2003), Chen et al. (2010)), along with heuristics to bound the optimal solution.

In general, these exact methods explore the solution tree by selecting variables to fix, solving the associated sub-problem, and using the result to update upper and lower bounds in order to *fathom* provably suboptimal portions of the tree. Solving sub-problems is generally done via *linear programming* (*LP*) *relaxation*, i.e., relaxing the integer constraints and solving using a variation of the *simplex method* or other LP solution method (Grötschel and Lovász 1995, Wolsey and Nemhauser 2014, Hoffman and Ralphs 2013).

Aardal et al. (2007) note the particular challenge posed by the minimum-interference CAP due to its weak linear programming relaxation, and also note the relative dearth of literature on the topic. Hassan and Chickadel (2011) provide a brief overview of graph coloring methods to minimize interference in wireless networks. Ahmadi and Pan (2011) use IP to solve the minimum-interference CAP, but consider only pairwise interference and use much smaller problem instances (12 nodes). Sridhar et al. (2009) use Lagrangian relaxation with their IP to solve the MI-CAP and provide a lower bound to the solutions they obtain using a heuristic. They too consider only pairwise interference, and use problem instances of 60 nodes or less.

Tiourine et al. (1995) are the first to work on bounding the MI-CAP. Fishburn et al. (1998) establish a lower bound for the number of interfering edges when coloring a d-regular graph. Montemanni et al. (2001, 2004) refine the work of Koster (1999) to establish lower bounds for the number of interfering pairs of transmitters within cliques. Montemanni et al. (2001) also develop a closed-form equation for bounding the number of interfering pairs of transmitters within cliques, which we use to bound the goodness of our MI-CAP. Subramanian et al. (2008) use an exact IP technique based in part on Montemanni et al. (2001) to provide a lower bound to their MI-CAP, and they use this bound to gauge the performance of their tabu search algorithm. However, their network instances are much smaller than ours (50 nodes).

The CAP can also be expressed as a *constraint satisfaction problem* (*CSP*) or *constraint program* (*CP*) as first suggested by Dunkin and Jeavons (1997). CSPs determine if there exists a consistent assignment of variables that satisfies a system of logical constraints. Related *weighted constraint satisfaction* or *optimal soft arc consistency* problems aim to find a solution which minimizes penalties associated with violating these logical constraints (Rossi et al. 2006, Cooper et al. 2007). We reformulate our MO-CAP as a CP to aid in determining lower bounds, and use an optimal soft arc consistency approach to solve the MI-CAP.

Dunkin et al. (1998) model their CAP and solve the problem using custom CSP code, but they consider only groups of seven or fewer transmitters for their dataset of 37 transmitters. Our datasets (and the number of associated logical clauses) are much larger and may be beyond the ability of current constraint satisfaction solvers when considered *en masse*. Palpant et al. (2008) solve their cumulative interference CAP using a hybrid of constraint programming and heuristic methods, and provide comparable or better performance than heuristic methods (specifically Sarzeaud and Berny (2003) and Dupont et al. (2005)) using a dataset from a military application. Hu (2012) considers the same dataset, and uses constraint satisfaction to identify *irreducible infeasible subsets*, which are useful in finding geographic areas where a given number of available channels may be insufficient, i.e., the subproblem is infeasible.

Constraint satisfaction may also be used within a *Benders decomposition framework* (see, e.g., Hooker (2011), Hooker and Ottosson (2003), Chu and Xia (2004)). We use constraint programming, integer optimization, and decomposition techniques to solve various subsproblems within a larger CAP framework. Another approach worth investigation is solving certain geographic areas of the problem at a time (see, e.g., Ding et al. (2010)).

2.3.2 Heuristic Solution Methods

Due to the computational difficulties of exactly solving the CAP, heuristics are often used to solve the problem (Aardal et al. 2007, Mannino and Sassano 2003).

Heuristic methods that have been used to consider cumulative interference CAPs include *neighborhood search* (Palpant et al. 2008, Voudouris and Tsang 1998); *simulated annealing* (Sarzeaud and Berny 2003, Smith et al. 2001), *tabu search* (Dupont et al. 2005, Smith et al. 2001, Capone and Trubian 1999, Vlasak and Vasquez 2003); *ant colony optimization* (Montemanni et al. 2002), *greedy heuristics* (Gomes et al. 2001, Daniels et al. 2004, Babadi and Tarokh 2010, Yu et al. 2013, Nicholas 2016), and a combination of greedy and exact methods (Palpant et al. 2008, Alouf et al. 2005).

Skalli et al. (2007) survey channel assignment methods for wireless mesh networks, and Katzela and Naghshineh (1996) do the same for cellular systems. Both categorize techniques as fixed (i.e., not changing over time), dynamic, or hybrid. Skalli et al. (2007) describe the *ripple effect* which often affects heuristics, where an already-assigned node is repeatedly revisited. This increases time to convergence and/or the complexity of the algorithm. They develop a new centralized algorithm with fixed channel assignment, i.e., without considering changes over time. Katzela and Naghshineh (1996) describe schemes that set aside a portion of channels in a common pool, to be dynamically assigned as needed.

The problem of assigning units to channels naturally lends itself to a *clustering* interpretation. Xu et al. (2005) provide a well-referenced survey of clustering algorithms. Abbasi and Younis (2007) and Boyinbode et al. (2011) both provide surveys of clustering algorithms specifically for *wireless sensor networks*, which share some important common features with the MANETs we consider. The use of clusters to "bin" radios has much in common with packing problems (see, e.g., Dowsland and Dowsland (1992) and Sung and Wong (1997)), which we consider when handling our cumulative co-channel interference constraints.

While heuristics can often provide useful solutions in reasonable amounts of time, in general they do not provide certificates of optimality for any particular solution, i.e., the distance to the global optimum is unknown. We feel these bounds are important for understanding the goodness of a particular solution, especially since spectrum is increasingly crowded and scarce.

2.3.3 Dynamic Spectrum Access

Dynamic spectrum access (*DSA*) is a broad term that refers to dynamic (rather than fixed) allocation of spectrum. Spectrum may be assigned in a centralized or distributed fashion, and is reassigned based on the current state of the environment, including changing radio locations, interference, traffic patterns, and even market conditions (Zhao and Sadler 2007). In general, DSA technology assumes channels can be changed dynamically by each radio at little or no cost (Akyildiz et al. 2008).

As previously noted, for our application there is a cost (namely, configuration time) associated with changing channels. Another difference of our approach is that we do not instantly react to new environmental states as they occur: to do so may require many frequent channel reassignments. Rather, we leverage available information regarding future radio locations to determine channel allocations that both efficiently use spectrum and minimize the number of required channel changes over a given planning horizon. Our methods can be re-run as often as needed with new information on the status of the operating environment, but our aim is to provide a degree of temporal stability to channel allocation. We provide a brief overview of DSA research to identify some similarities and differences of our work and DSA.

The term *NeXt Generation* (*xG*) network is sometimes used synonymously with DSA (Akyildiz et al. 2008). However, we follow Zhao and Sadler (2007) and do not use the term *cognitive radio* synonymously with DSA. Cognitive radio refers to devices that can change their configurations in a dynamic and intelligent manner based on current conditions (Federal Communications Commission 2003). Cognitive radio technology includes DSA capabilities but also includes dynamic power allocation, antenna reconfiguration, and differing signal encoding and modulation schemes.

Akyildiz et al. (2006, 2008) and Zhao and Sadler (2007) provide overviews and surveys of DSA technology. Following Zhao and Sadler (2007), DSA can be divided into three categories of models. *Dynamic exclusive use models* assign spectrum to licensed users for exclusive use. Under this model, licensees may sell and trade their spectrum rights

using market mechanisms, or they may use *dynamic spectrum allocation* to assign spectrum based on current environmental conditions. Under the *open sharing model*, all users share spectrum in a peer relationship. The most common example of this is WiFi. Under the *hierarchical access model*, primary users are licensed to use spectrum at their convenience; secondary users are allowed to use spectrum when primary users are not, or at transmission powers that are below the noise floor of primary users. Technology categorized within this model would not apply to the tactical MANET radios which we consider, which are *constant key* devices and continually transmit, regardless of current traffic levels.

Other relevant papers on DSA include Ding et al. (2010), who describe a distributed, localized algorithm where each radio makes real-time decisions based on locally-collected information. They consider cumulative interference at each node in a *signal-to-interference ratio* (*SIR*) format (very similar to our approach). They also bound upper and lower transmission power based on performance and noise thresholds. Zhao et al. (2005) used a distributed, coordinated method to dynamically assign spectrum, but use a simply binary interference model. Riihijärvi et al. (2005) relate their distributed, dynamic channel allocation method for *wireless local area networks* (*WLANs*) to graph coloring techniques, but do not consider cumulative interference. Garcia Villegas et al. (2005) also using a distributed, dynamic model for WLANs. They use a minimum interference objective function, as this makes sense for WiFi applications (where the number of available channels is fixed), and scan and react to cumulative interference.

2.3.4 Mobility-aware Methods and Temporal Graphs

Most of the methods described thus far are generally applied to fixed CAPs, where assignments are permanent or not expected to change quickly. Dynamic CAPs consider frequent channel changes, but most of these methods simply repeatedly apply fixed CAP methodologies (usually heuristics), or employ schemes for borrowing channels between radios, without consideration of reducing reassignments over time. See Katzela and Naghshineh (1996) for a survey on dynamic CAPs.

A seldom-researched challenge of the mobility-aware CAP is channel allocations changing over time, and not just at certain points in time, i.e., a *myopic solution*. Such a solution may needlessly flip-flop channel assignments, and may be particularly fragile to changes in physical network topologies. The movement of radios in a military environment is far from arbitrary (Zhou et al. 2004, Nicholas et al. 2013b); by leveraging available information on the future locations of radios and considering the effects of network perturbations (such as degraded signal quality), one can provide a more far-sighted and robust solution to reduce the number of required channel changes over time. This decreases the time used by operators to manually adjust radio configurations, and the time needed by the spectrum manager to de-conflict unexpected interference. Changes over time make the challenges we consider that much more difficult, as now we must compute possible channel assignments over multiple time steps.

We assume our spectrum manager has some information available regarding the future positions of radios. In a combat environment, this information may be incomplete or later proved to be entirely inaccurate, but we assume there is at least some utility in considering this information in allocating channels. Tseng et al. (2002) present (apparently for the first time) a location-aware method for dynamically allocating channels to MANETs. Their methods are similar to those used to support GSM cellular service, including channel borrowing. The tactical MANET radios we consider have GPS-enabled position-location information (PLI) available for use, and new technology such as DARPA's RadioMap will allow each radio to sense the interference environment in real time (Defense Advanced Research Projects Agency 2015). However, we still require the ability to plan channel assignments in advance, as channels will not be automatically assigned and configured. Kostakos (2009) and Whitbeck et al. (2012) develop the concept of *temporal graphs* (or *evolving graphs*), which Yu et al. (2013) use to consider channel assignment over time, rather than just a series of snapshots.

Casteigts et al. (2012) provide an overarching framework of time-varying graphs in pursuit of general properties, and mention that very little work on algorithms and protocols

has been done in this area. In a seminal and highly-referenced work, Ferreira (2004) provides several useful definitions of terms relevant to evolving graphs, and describes the use of such graphs to consider time-varying MANETs. This work was extended by Monteiro et al. (2006), who use evolving graphs to model dynamic MANET communications, and by Ferreira et al. (2010), who demonstrate the use of evolving graphs to analyze MANET protocol performance.

Scellato et al. (2013) present for the first time a measure of temporal robustness for time-varying mobile networks (an area of research that surely is applicable to military problems), but do not provide any measures specific to channel allocation.

One of the most relevant papers to our research is that of Yu et al. (2013), who present a unique methodology for channel assignment using temporal graphs. They develop several heuristics to solve their multi-objective optimization problem to minimize the number of required channels, while also considering co-channel interference and the cost of changing channels over time. They evaluate several different algorithms, including *SNAP*, which assigns colors for each "snapshot" in time independently (without regard to channel reassignment), and *SMASH*, which "smashes" together all of the snapshots into a single temporal graph and assigns channels considering channel reassignments. They state their work is the first to apply a temporal graph methodology to the channel assignment problem that considers the cost of reassignment.

Our work builds upon Yu et al. (2013). They use a "protocol model" for interference calculations, whereas we use a much more realistic SIR model. They assume random unit mobility, whereas we base future locations upon a military *concept of operations* (specific to each scenario). They use only greedy heuristics and pairwise interference constraints, whereas we use exact optimization techniques (which allow us to provide a measure of goodness for a given solution) and cumulative interference, which is more realistic for military MANET operations.

2.3.5 Parallel and Distributed Methods

We assume our spectrum manager will be solving the problem from a central location and will have multiple computers and/or cores available. As described, the CAP can be unwieldy for realistic problem sizes. We assume our spectrum manager will leverage distributed and parallel computation to quickly obtain useful solutions.

Computing technology has advanced significantly since much of the work on CAPs in the late 2000s. Most computers and even smartphones have multiple cores, yet most algorithms specifically developed for solving CAPs are serial and do not take advantage of parallel and distributed computation. The problem has structure that seems to naturally lend itself to decomposition (e.g., into physical neighborhoods of radios, or by separate time steps), increasing the desirability of applying parallel and distributed techniques. New versions of both CPLEX and Gurobi enable distributed implementations, and there are several free computing packages to support distributed programming (see, e.g., the Python `dispy` library (Pemmasani 2016)).

Crainic et al. (2006) provide a seminal survey paper focusing on parallel implementations of the branch-and-bound algorithm. Drummond et al. (2006) use a distributed branch-and-bound approach to solve the *Steiner tree problem* (see, e.g., Hwang et al. (1992)), and consider the effects of unreliable communications links between processes (which may affect our spectrum manager if he/she is using computing resources distributed across a tactical network). Modi et al. (2005) and Yeoh et al. (2008) respectively create and further develop a distributed constraint optimization framework, but they consider only binary constraints. Their work was extended by Pecora et al. (2006) to look at n-ary constraints, which would be necessary if we were to apply this framework to consider our cumulative co-channel interference problem.

The use of parallel and distributed methods to specifically solve the channel assignment problem include Zhao et al. (2005), who design their distributed algorithm around the assumption that a central control station may not be accessible by all nodes (which also

applies in our problem). They model only binary interference. Riihijärvi et al. (2005) and Garcia Villegas et al. (2005) each use a distributed, dynamic graph-coloring approach to assign channels to WLANs, but both require the existence of a *backhaul network* to enable a channel coordination mechanism among access points. Garcia Villegas et al. (2005) take dynamic readings of local background interference in order to inform channel assignment. Ding et al. (2010) look at simultaneously solving routing, power, and spectrum allocation decisions. They use a distributed, localized algorithm where each radio makes real-time decisions based on locally-collected information. They consider cumulative interference at each node using signal-to-interference ratio information, and provide computational results using network simulation.

2.3.6 Current Real-world Solution Methods

Military spectrum managers have several tools to assist in allocating spectrum. One is the Systems Planning, Engineering, and Evaluation Device (SPEED) (Lamar 2013). Another is Spectrum XXI (Defense Information Systems Agency 2013). Both tools use the high-fidelity Terrain Integrated Rough Earth Model (TIREM) (Alion Science and Technology Corporation 2016) (which we also use) to provide radio coverage analysis reports and interference calculations. Spectrum XXI provides a database to deconflict assignments across a given operating area. Neither of these software tools consider cumulative co-channel interference among a large number of mobile transmitters over multiple time periods, nor do they provide an automated method for minimizing the number of required channels. Rather, the spectrum manager simply tries out different solutions and attempts to manually reduce total spectrum requirements.

2.4 Relationship to the Literature

We use an integer programming formulation to model the CAP similar to Aardal et al. (2007), but unlike nearly all research on the CAP using IP, we consider cumulative co-channel interference. Our research builds upon the method of Yu et al. (2013), who consider co-channel interference but only solve small problem instances with heuristic methods. We develop a method for bounding the performance of the MI-CAP, and present a new method for minimizing the number of channel changes over time. To our knowledge, we are the first to use exact optimization methods to solve realistic, full-size instances of the cumulative interference MO-CAP and MC-CAP-T to global or near-global optimality.

Chapter 3: Model of MANET Communications

This chapter describes how we model and simulate MANET communications, and provides details on our datasets.

3.1 Preliminaries

We create a network model to simulate key aspects of a MANET formed by tactical wideband radios at a given moment in time (i.e., time step). We specifically model variants of the Harris PRC-117G Multiband Networking Manpack Radio (Harris Corporation 2016) and the *Adaptive Networking Wideband Waveform* (*ANW2*), but our technique is applicable to any type of EM transceiver system requiring a channel assignment.

Let $r \in R$ (alias s) represent each *radio*. Each radio is permanently assigned to a MANET *unit* $u \in U$, indicated by the set of *logical arcs* $(r, u) \in L$. A unit may represent a tactical organization such as an infantry company or reconnaissance team. Let the set of *nodes* N (indexed by n) consist of both radios R and units U, i.e., $n \in N = R \cup U$. Let a channel $c \in C$ be a contiguous range of EM frequencies, where C is the set of available orthogonal (i.e., non-interfering) channels. Each unit u and the radios $r \in R$ assigned to it require a channel assignment.

Let $(r, s) \in W$ indicate the set of arcs representing wireless transmissions between all radios $r, s \in R$. These arcs represent both intentional EM transmissions between radios assigned to the same unit, and unwanted interference from all other radios assigned to the same channel $c \in C$. These arcs exist in both directions, and each radio can receive transmissions from any other radio, so $|W| = |R|\,(|R| - 1)$.

We do not explicitly model communications within a MANET formed by a unit, but we must simulate this communication prior to solving our channel assignment problem in order to calculate the maximum allowable interference and provide that information as

input to the formulations. A unit $u \in U$ forms a MANET among its assigned radios using the available wireless arcs $(r, s) \in W : (r, u) \in L, (s, u) \in L$. Each MANET enables the exchange of communications traffic between all radios and a *network control radio*, such as the infantry company commander or reconnaissance team leader. This bi-directional connectivity to a single radio ensures that radios within each unit are *strongly connected* (i.e., a directed path exists between each pair of radios) (Ahuja et al. 1993). Technological limits of the radios constrain the number of radios that can be assigned to the same unit; we assume a limit of 30 radios.

Figure 3.1 shows two separate units (indicated in blue and green) and their assigned radios. The solid lines indicate bidirectional wireless arcs $(r, s) \in W$ between radios. Any radio (e.g., radio r in Figure 3.1) communicates with its network control radio (e.g., radio s) via these arcs (a radio may route through other radios in the same unit to reach the network control radio). All radios are subject to co-channel interference from any other radios assigned to different units but operating on the same channel, indicated by dashed gray arrows directed to r (other lines withheld for clarity). In our scenarios, there are no connections between units; that is, disparate MANETs are not connected via a *backhaul network*. In practice, connectivity between units (if any) is provided by satellite, fiber optic cable, or other backhaul network.

Using its assigned channel, each independent MANET uses *orthogonal frequency division multiplexing* (*OFDM*) to enable connectivity between assigned radios, though other multiplexing techniques may be used without altering our formulation.

3.2 Calculating Received Signal Strength

To calculate both co-channel interference and the strength of desired wireless transmissions between intra-unit radios, we calculate the *received signal strength* (*RSS*) ρ_{rs} along all wireless arcs $(r, s,) \in W$ in dBm (decibel-milliwatts) using the standard *link budget* formula (Olexa 2004):

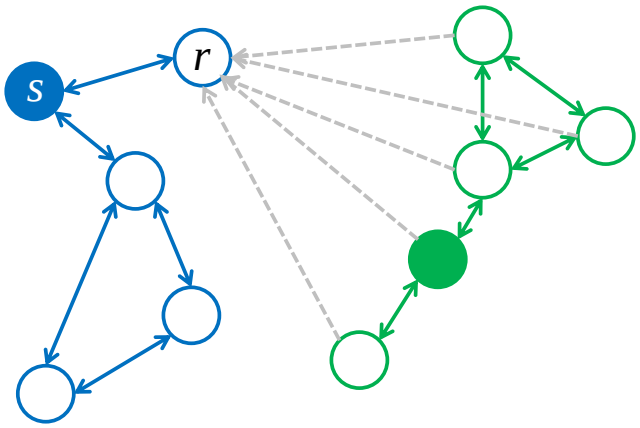


Figure 3.1: Simple example of two units (indicated in blue and green) with network control radios (solid circles) and other radios (open circles). Wireless arcs are indicated by arrows. The radios within each unit must be capable of bi-directional communication with their network control radio via direct communication or routing through other radios in the same unit. All radios are subject to co-channel interference (dashed arrows) from other radios assigned to different units but operating on the same channel.

$$\rho_{rs} = power_r + g_r - l_r - l_{path} - l_{misc} + g_s - l_s \qquad \forall\,(r, s,) \in W \tag{3.1}$$

where $power_r$ is transmitted power in dBm, g_r and g_s are respectively the gains of the radios r and s in dB, l_r and l_s are respectively the losses (i.e., from cables, connectors, etc.) of the radios in dB, l_{path} is total path loss in dB, and l_{misc} is miscellaneous loss or fade margin in dB. All of the terms are input data, determined by the equipment and environment, except for the total path loss l_{path}, which depends on the physical position of radios r and s and the intervening terrain.

Our formulation allows the use of any method for computing l_{path}, including the Irregular Terrain Model (ITM) (Longley and Rice 1968) and Hata-COST 231 (Cichon and Kürner 1993). We instantiate our scenarios in Systems Toolkit (STK) (Analytical Graphics, Inc. 2016) and then use Python and the Terrain Integrated Rough Earth Model (TIREM) of Alion Science and Technology Corporation (2016) to calculate l_{path}. We use STK to consider

the movement of various types of platforms (e.g., ground troops, vehicles, aircraft, etc.) in a three-dimensional environment, defining realistic locations and velocities according to the scenario concept of operations. TIREM samples terrain elevation to compute path loss, and considers the effects of free space loss, diffraction around obstacles, and atmospheric absorption and reflection.

We can use STK to run TIREM to calculate l_{path}, but STK is very graphics-intensive and calculates many other quantities with which we are not particularly concerned (e.g., Doppler shift and the effects of different signal coding schemes). This computational overhead causes very slow simulation runtimes: simulating just a single time step of our largest scenario in STK takes over one week. To reduce this runtime, we export the locations of each radio from STK to a text file and then use Python to iteratively calculate l_{path} via TIREM (as a dynamic-link library) for each radio pair at each time step. Using this method, we are able to reduce simulation runtimes to less than one day using a laptop computer. While TIREM is computationally more expensive than simpler models, it provides fairly accurate results. For line-of-sight propagation in commonly-used frequency ranges, Eppink and Kuebler (1994) compare TIREM predictions and actual measurements. They find a difference with a mean of -2.8 dB and a standard deviation of 8.9 dB, which is very accurate considering the speed and relative simplicity of the model. Nicholas and Alderson (2012) use TIREM to simulate WiFi propagation, and find the computed predictions to be very comparable to that obtained during real-world field testing.

In general, higher frequency signals will propagate farther than lower frequencies. We are able to use TIREM to calculate propagation at any frequency band within the operating specifications of our modeled radios. However, Nicholas et al. (2013a) and Nicholas (2016) use a similar modeling approach and find that the cumulative effects of channels at various frequencies tend to essentially cancel out. Specifically, higher frequencies propagate less far and produce less interference, but are more sensitive to interference because their intra-unit signal strengths are weaker. Conversely, lower frequencies propagate farther and produce more interference, but they are less sensitive to interference. We thus make a simplifying

assumption that each channel will perform roughly the same in our scenarios, though our formulations allow for the general case where channel performance varies.

3.3 Calculating Connectivity and Interference

To calculate the strength of connectivity between each radio and its network control radio, we use *Dijkstra's algorithm* (Dijkstra 1959) to calculate the *shortest path* from each radio to its assigned network control radio. Arc cost is defined to be inversely proportional to the RSS ρ_{rs} between radios. This methodology favors paths that have both fewer links and higher received signal strengths, and is similar to the *Open Shortest Path First* (*OSPF*) routing algorithm (Moy 1998, Coltun et al. 2008). We assume a radio will be disconnected from its assigned network control radio if it is unable to communicate along this shortest path, as all other paths will be more costly (i.e., consist of more links and/or links of weaker signal strength). Along each path and at each radio $s \in R$, we follow Aardal et al. (2007) and pre-calculate the maximum allowable interference in watts $max\_interference_s^c$. This calculation is based on the RSS ρ_{rs} between radios and each particular radio's required signal-to-interference ratio (SIR), a measure of signal quality (Poisel 2011). Any co-channel interference above this level severs the shortest path and thus disconnects the radio from its assigned network control radio. (Unless otherwise noted, throughout this work we assume a minimum required SIR of 10 dB.)

Among radios not assigned to the same unit but operating on the same channel, the RSS ρ_{rs} represents co-channel interference. The magnitude of co-channel interference along all arcs $(r, s) \in W$ for each available channel $c \in C$ is pre-calculated in watts and is indicated by $interference_{rs}^c$. (We simulate transmissions between all radios, though in practice some arcs may represent negligible or zero interference.)

Based on Marine Corps wideband spectrum allocation practices and following Nicholas et al. (2013a), we assume spectrum is pre-divided into channels with sufficient *white space* to prevent adjacent-channel or other *harmonic interference* between channels. Hence, we

need only consider co-channel interference. While other factors (such as signal modulation schemes and routing protocols) will affect the ability of two radios to communicate, in the scenarios we consider (with mobile radios operating over rough terrain), propagation loss and signal interference are by far the two strongest determinants of radio performance (Molisch 2011, Katzela and Naghshineh 1996, Nicholas et al. 2013b).

The following pseudo-code describes our algorithm for calculating connectivity and interference in our MANET model. Throughout this document, the arrow notation $x \leftarrow y$ indicates the assignment of value y to variable x.

Algorithm *Calculate Connectivity*

Input: Radio technical specifications and locations; terrain and atmospheric data; required SIR (in dB)

Output: $max\_interference_s^c \quad \forall s \in R, c \in C$

begin
  Calculate path loss l_{path} along all wireless arcs $(r, s) \in W$
  Calculate ρ_{rs} along all wireless arcs $(r, s) \in W$
  for $u \in U$
    for $r, s \in u$
      $minSignal_s \leftarrow \infty$
      $arcCost_{rs} \leftarrow \dfrac{1}{\rho_{rs}}$
    next;
    for $r \in u$
      $path_{ru} \leftarrow$ Shortest path from r to network control radio for unit u
    next;
    for each $path_{ru}$
      for $s \in path_{ru}$ // For each radio in $path_{ru}$
        $signal_s \leftarrow \min(\rho_{s+1,s}, \rho_{s-1,s})$
        if $signal_s < minSignal_s$
          $minSignal_s \leftarrow signal_s$ // Set minimum received at s
        endif;
      next;
    next;
    for $s \in u, c \in C$
      $max\_interference_s^c \leftarrow minSignal_s - requiredSIR$
    next;
  next;
end;

3.4 Description of Datasets

We use realistic datasets depicting particular time steps within high-fidelity simulations of U.S. Marine Corps combat operations. We use Systems Toolkit (STK) (Analytical Graphics, Inc. 2016) to develop our scenarios, i.e., to position radios in time and space according

to the scenario *concept of operations*, and then use Python and TIREM to calculate radio propagation between all radios at each time step. We consider three tactical Marine Air-Ground Task Force (MAGTF) scenarios, each with different network topologies. The first scenario, based on Major Combat Operation 1 (Department of Defense 2007) involves a Marine Expeditionary Unit (MEU) conducting an amphibious assault on an island. The second scenario, based on combat operations in Helmand Province, Afghanistan circa January 2010, is a Marine Expeditionary Brigade (MEB) conducting irregular warfare (IW) operations in a desert environment. Our final scenario, based on Integrated Security Construct B (Department of Defense 2013), is a Marine Expeditionary Force (MEF) conducting a major amphibious assault. Each of these scenarios include classified details; we make inconsequential adjustments to the scenarios to keep this research unclassified and to be able to provide the datasets to the research community.

A description of these scenarios by number of Marines, units, and radios is displayed in Table 3.1. We find the largest scenario to be the most computationally interesting, and so we generate separate datasets at 20 different time steps within that scenario (each with 118 units comprising 1887 total radios). Nicholas et al. (2013a) provide full details on our scenarios.

Table 3.1: Description of datasets, depicting the number of Marines, units, and radios represented in each combat scenario.

Scenario	**Marines**	**Units**	**Radios**
MEU	2000	6	131
MEB	15,000	24	641
MEF	60,000	118	1887

In general, we observe that the MANET radios assigned to a particular unit are located relatively close to one another due to the limitations of transmission distance, as is evident in Figure 3.2, a Google Earth (2016) image of all radios in the MEF scenario during the first time step. The distance between units depends on the particular scenario and associated

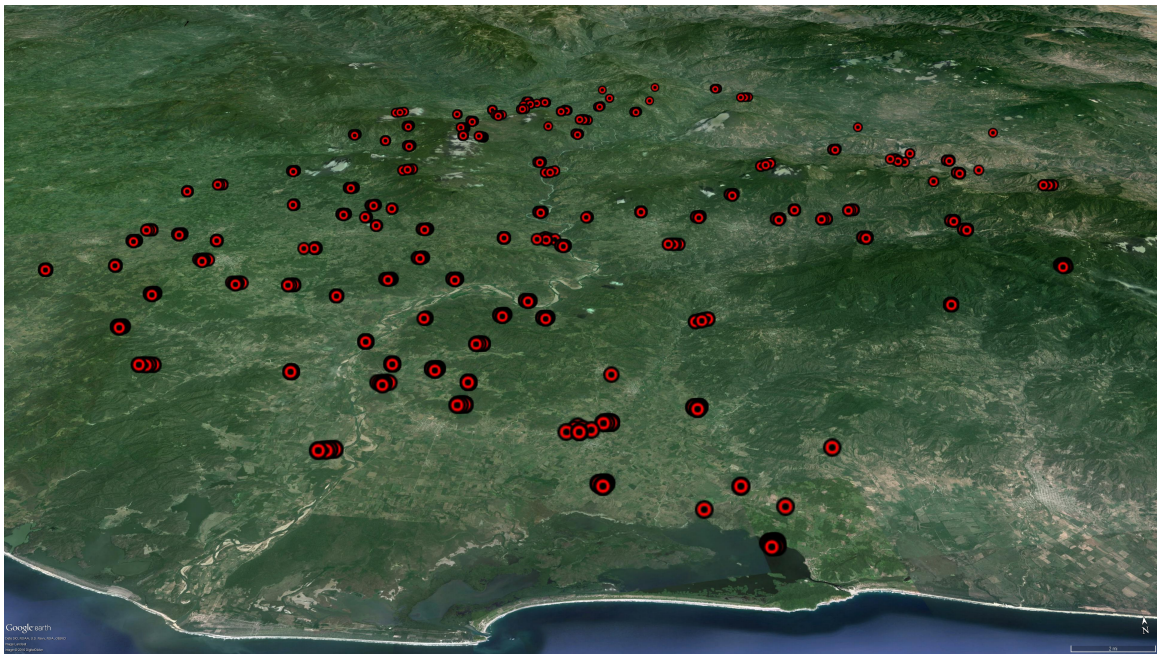


Figure 3.2: Locations of 1887 radios at the first time step within the MEF scenario. [Image courtesy of Google Earth Pro (2016) and Digital Globe (2016)].

concept of operations. For example, units may be located close to each other when building combat power ashore during an amphibious assault, but thereafter may be relatively dispersed as forces push farther inland.

Figure 3.3 displays the location of each radio in the MEF scenario at each of 20 time steps (arranged from upper left to lower right), where each grid line represents one degree of latitude and longitude (approximately 69 miles or 111 kilometers). During the first time step, the units have just arrived ashore during the amphibious assault and are located relatively close to one another. As time progresses, the units gradually disperse and move northward.

In Table 3.2, we provide descriptive statistics of the MEF scenario by time step. "Number of pairwise constraints" is the number of units that cannot be assigned the same channel at the same time without excessive co-channel interference (see Section 4.1). The graph

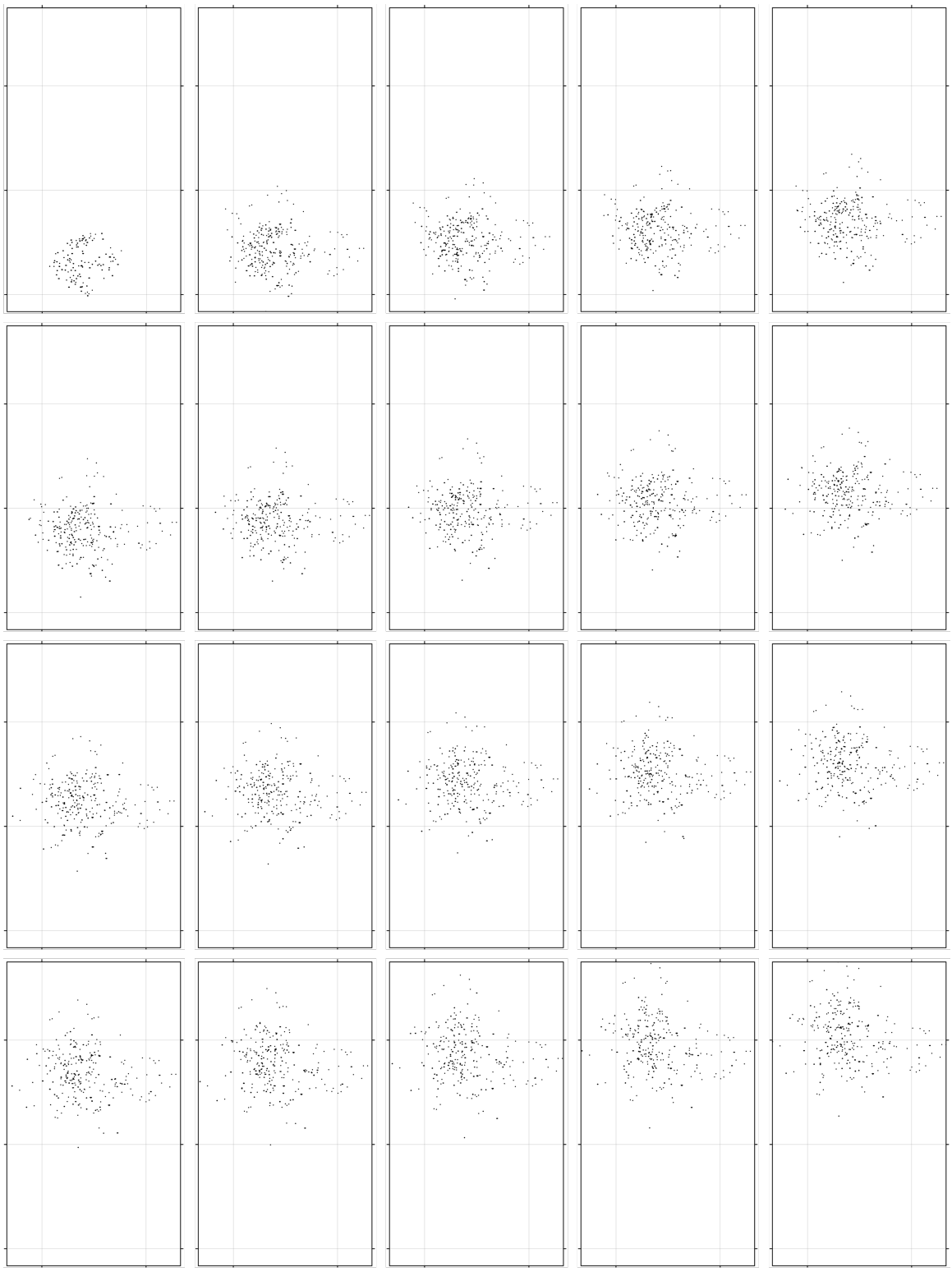


Figure 3.3: Locations of radios within the MEF scenario, by time step (from upper-left to lower-right).

formed by these constraints is an interference graph, and we calculate the density and average degree for this graph. We provide several measures of geographic dispersion, including "Geographic diameter" (the longest great circle distance between any two radios), and the average and standard deviation of distances between each radio, and between each network control radio (NCR). In general, the MEF formation becomes more dispersed as time progresses. Not surprisingly, the highest graph density occurs when the geographic dispersion is the smallest (i.e., time step one).

Table 3.2: Descriptive statistics of the MEF scenario, by time step.

Time Step	Number of Pairwise Constraints	Interference Graph Density	Interference Graph Average Degree	Geographic Diameter (km)	Average Distance Between Radios (km)	σ of Distance Between Radios (km)	Average Distance Between NCRs	σ of Distance Between NCRs
1	4407	0.6384	74.69	75.31	29.09	14.31	28.84	14.19
2	3892	0.5638	65.97	147.36	36.55	20.20	29.62	14.53
3	3945	0.5715	66.86	143.75	36.71	20.02	30.10	14.86
4	3823	0.5538	64.80	147.23	37.50	20.41	30.67	15.12
5	3762	0.5450	63.76	151.67	38.32	21.10	31.65	15.60
6	3904	0.5656	66.17	150.74	38.37	21.53	31.51	15.79
7	3884	0.5627	65.83	157.50	39.09	21.93	32.14	16.20
8	3538	0.5125	59.97	151.44	39.73	22.07	32.65	16.31
9	3398	0.4922	57.59	151.54	40.55	22.31	33.38	16.62
10	3541	0.5130	60.02	156.47	41.15	22.77	34.16	17.46
11	3449	0.4996	58.46	165.48	42.17	23.24	34.96	17.88
12	3367	0.4878	57.07	165.79	42.47	23.79	35.35	18.20
13	3550	0.5143	60.17	163.73	42.90	23.73	36.16	18.48
14	3301	0.4782	55.95	165.30	42.86	24.17	36.24	18.79
15	3666	0.5311	62.14	168.04	43.43	24.38	36.84	18.91
16	3749	0.5431	63.54	167.17	43.24	24.35	36.85	18.96
17	3721	0.5390	63.07	171.44	44.59	25.13	38.07	19.76
18	3214	0.4656	54.47	173.29	45.01	25.29	38.88	20.06
19	3282	0.4754	55.63	175.19	45.10	25.63	38.94	20.38
20	3660	0.5302	62.03	176.35	45.81	25.46	39.48	20.35
Aver:	**3652.65**	**0.5291**	**61.91**	**156.24**	**40.73**	**22.59**	**34.33**	**17.42**

While our scenarios are derived from official U.S. Defense Planning Scenarios, our methods apply to any type of input dataset, including randomly-generated datasets. Unlike the publically available test sets often used for CAP research (e.g., CELAR or CALMA), our

approach assumes that interference constraints must be discovered from raw radio propagation data, i.e., they are not already provided in the form of preprocessed interference constraints.

3.5 Computational Resources

Unless otherwise indicated, all results are obtained using a Dell Mobile Precision 6800 laptop with 32 GB of RAM and an Intel Core i7-4940MX processor running at 3.1 GHz. We use IBM ILOG CPLEX version 12.6.2 and Python 2.7.

Chapter 4: Minimum-Order Channel Assignment Problem

This chapter describes the minimum-order channel assignment problem (MO-CAP), which aims to minimize the total number of channels required to support MANET communications at a given time step, subject to cumulative interference constraints. We provide several formulation variations, and include our solution techniques and results. To our knowledge, we are the first to solve this problem to global or near-global optimality for a realistic, full-size dataset.

4.1 MO-CAP Full Standard Formulation

The minimum-order channel assignment problem (MO-CAP) aims to minimize the total number of channels required to support MANET operations at a given moment in time. Let the binary variable X_n^c indicate whether node n (either a radio or a unit) is using channel c:

$$X_n^c = \begin{cases} 1, & \text{if node } n \text{ uses channel } c \\ 0, & \text{otherwise} \end{cases} \qquad \forall n \in N, c \in C. \tag{4.1}$$

Each radio is assigned the same channel as its associated unit, so

$$X_r^c = X_u^c \quad \forall c \in C, (r,u) \in L. \tag{4.2}$$

To ensure each unit u is assigned one and only one channel, the problem contains the constraint:

$$\sum_{c \in C} X_u^c = 1 \quad \forall u \in U. \tag{4.3}$$

Let the binary variable Y^c indicate whether channel c is being used:

$$Y^c = \begin{cases} 1, & \text{if channel } c \text{ is used} \\ 0, & \text{otherwise} \end{cases} \qquad \forall c \in C. \tag{4.4}$$

Since the goal is to minimize the total number of required channels, our objective function is:

$$\min \sum_{c \in C} Y^c. \tag{4.5}$$

Two radios from different units are subject to interference if they are both assigned to the same channel, so one possible constraint is:

$$interference^c_{rs} X^c_r X^c_s \leq max\_interference^c_s \quad \forall (r,s) \in W, c \in C. \tag{4.6}$$

That is, a radio $s \in R$ may be assigned a particular channel $c \in C$ only if the interference from any other single radio is at or below the pre-calculated $max\_interference^c_s$ threshold. Following Katzela and Naghshineh (1996) and Ståhlberg (2000), we assume the cumulative effects of jamming sources on the same channel are additive (in watts) at each receiver. That is, a radio $s \in R$ may be unable to use a channel $c \in C$ because the total sum of interference exceeds the threshold $max\_interference^c_s$, even if the interference received from any single radio is less than the threshold. Summing along all arcs yields:

$$\sum_{r:(r,s) \in W} interference^c_{rs} X^c_r X^c_s \leq max\_interference^c_s \quad \forall s \in R, c \in C. \tag{4.7}$$

Figure 4.1 provides a graphical representation of several possible interference conditions between a receiver r and transmitters s and t, where the colored blobs represent radio propagation. In Figure 4.1a, r receives an acceptable amount of interference from transmitter t and thus these two radios may be assigned the same channel. In Figure 4.1b, r receives

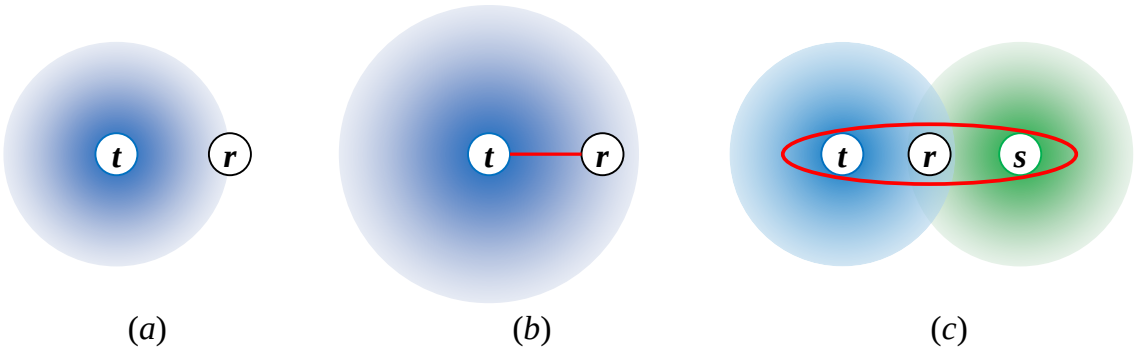


Figure 4.1: Several possible interference conditions between a receiver r and transmitters t and s, where a red arc or hyper-arc connects radios that cannot be assigned the same channel.

unacceptable interference from t, and these two may not share a channel, i.e., one of the constraints (4.6) is violated. This is depicted as a red arc between the radios. Note this constraint exists if either or both radios receive unacceptable interference from the other. In Figure 4.1c, r receives acceptable interference from both transmitters t and s separately, but unacceptable interference when all three are assigned the same channel, i.e., at least one of the constraints (4.7) is violated. This is depicted as a red *hyper-edge* or *hyper-arc* among the three radios. The set of all pairwise interference constraints (e.g., the arc in Figure 4.1b) is an interference graph; the set of all interference constraints (pairwise and higher-order) is an *interference hypergraph*.

To linearize constraints (4.7), we introduce the binary variable Z^c_{rs} where:

$$Z^c_{rs} = \begin{cases} 1, & \text{if } X^c_r = X^c_s = 1 \\ 0, & \text{otherwise} \end{cases} \qquad \forall\,(r,s) \in W, c \in C \tag{4.8}$$

which is enforced via:

$$Z_{rs}^{c} \geq X_{r}^{c} + X_{s}^{c} - 1 \qquad \forall\,(r,s) \in W, c \in C \tag{4.9}$$

$$Z_{rs}^{c} \leq X_{r}^{c} \qquad \forall\,(r,s) \in W, c \in C \tag{4.10}$$

$$Z_{rs}^{c} \leq X_{s}^{c} \qquad \forall\,(r,s) \in W, c \in C. \tag{4.11}$$

We thus obtain our cumulative co-channel interference constraints:

$$\sum_{r:(r,s)\in W} interference_{rs}^{c} Z_{rs}^{c} \leq max\_interference_{s}^{c} \quad \forall s \in R, c \in C. \tag{4.12}$$

Given the results of radio propagation simulation in a combat scenario, we pre-calculate the $max\_interference_{s}^{c}$ values (using the method described in Chapter 3), and fix the assignment of radios to their respective units (indicated by arcs $(r,u) \in L$). We summarize our *Full Standard Formulation* (*FSF*) of the MO-CAP as follows:

MO-CAP Full Standard Formulation

Index and Set Use

$n \in N$	node (either radio or unit)
$r \in R \subset N$	radio (*alias s*)
$u \in U \subset N$	unit
$c \in C$	channel
$(r,u) \in L$	arc indicating logical assignment of radio $r \in R$ to unit $u \in U$
$(r,s) \in W$	arc indicating wireless interference between radios r and $s \in R$ where r and s are not in the same unit, i.e., $r,s \notin u, \forall u \in U$

Input Data

$interference^c_{rs}$	interference on $c \in C$ along arc $(r,s) \in W$ [watts]
$max\_interference^c_s$	max allowable interference $s \in R$ and $c \in C$ [watts]

Decision Variables

X^c_n	binary variable indicating whether n is using c
Y^c	binary variable indicating whether channel c is being used
Z^c_{rs}	binary variable indicating whether r and s are both using c

Formulation

$$\min_{X,Y} \quad \sum_{c \in C} Y^c \tag{F0}$$

$$s.t. \quad X^c_u \leq Y^c \qquad \forall u \in U, c \in C \tag{F1}$$

$$\sum_{c \in C} X^c_u = 1 \qquad \forall u \in U \tag{F2}$$

$$X^c_r = X^c_u \qquad \forall c \in C, (r,u) \in L \tag{F3}$$

$$\sum_{r:(r,s) \in W} interference^c_{rs} Z^c_{rs} \leq max\_interference^c_s \qquad \forall s \in R, c \in C \tag{F4}$$

$$Z^c_{rs} \geq X^c_r + X^c_s - 1 \qquad \forall (r,s) \in W, c \in C \tag{F5}$$

$$Z^c_{rs} \leq X^c_r \qquad \forall (r,s) \in W, c \in C \tag{F6}$$

$$Z^c_{rs} \leq X^c_s \qquad \forall (r,s) \in W, c \in C \tag{F7}$$

$$X^c_n \in \{0,1\} \qquad \forall n \in N, c \in C \tag{F8}$$

$$Y^c \in \{0,1\} \qquad \forall c \in C \tag{F9}$$

$$Z^c_{rs} \in \{0,1\} \qquad \forall (r,s) \in W, c \in C \tag{F10}$$

The MO-CAP FSF is a pure 0-1 integer program. The objective function (F0) minimizes the sum of assigned channels. Constraints (F1) ensure that each channel utilized by a unit is counted toward the objective function. Constraints (F2) require the assignment of one

channel to each unit. Constraints (F3) require that each radio uses the same channel as its assigned unit. Constraints (F4) ensure that the sum total of co-channel interference at each radio is below the maximum threshold. Constraints (F5)-(F7) enforce the definition of Z^c_{rs}.

4.1.1 Computational Challenges of the MO-CAP FSF

The Full Standard Formulation is relatively easy to understand and describe. However, it suffers from several serious computational difficulties when the full problem is simply "thrown" at a commercial solver (e.g., CPLEX or Gurobi) with our realistic datasets. Following Nicholas and Hoffman (2015, 2016), we describe and provide evidence of these problems, and provide preliminary results that demonstrate the challenges.

First, commercial solvers may be sensitive to vast differences in input parameters. Our interference values may range from extremely small to quite large, depending on the distance and terrain between the given radios. In our simulated datasets, these values vary by 24 orders of magnitude, and are generally quite small (see an example from the MEF scenario in Figure 4.2). The CPLEX solver may experience difficulties when the objective function and constraint coefficients vary by six or more orders of magnitude (IBM 2013b). Also, non-integral input data may result in highly *fractionalized* LP solutions, as the solver will attempt to "pack" the most units (including fractions of units) onto the same channel. These fractional solutions must then undergo a computationally-costly *repair* process to become integer-feasible.

Another computational problem (also observed by Palpant et al. (2008)) is that of *symmetry*, which occurs when channel assignments may be changed among units with no corresponding change in the objective function value (Margot 2010). While there are performance differences between channels on different frequencies (i.e., lower frequency channels generally propagate farther than higher frequencies), these differences may be very slight or even indistinguishable (given computer floating-point precision) between proximate channels. When conducting a tree search over problems exhibiting near symmetry, solvers may waste time examining different solutions that provide essentially identical utility. The very

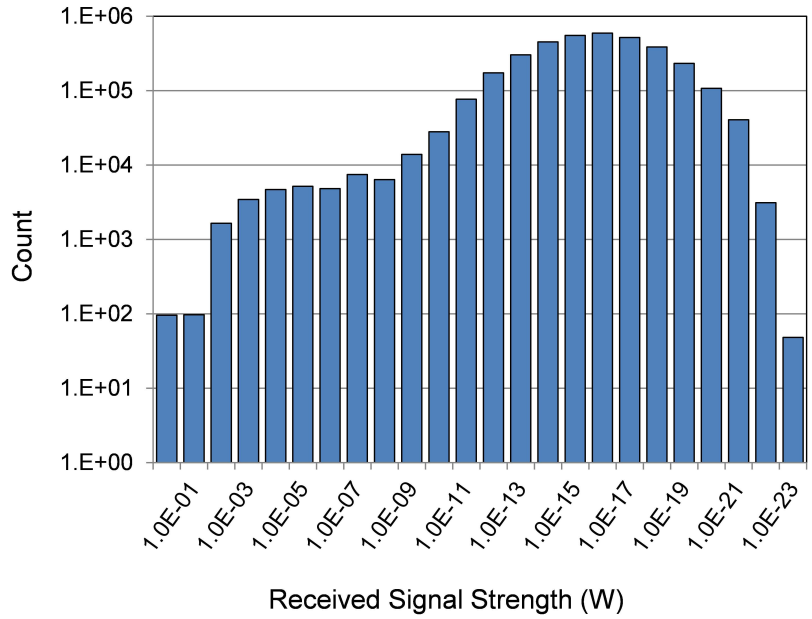


Figure 4.2: Distribution of received signal strengths between all radios in the first time step of the MEF scenario. The values vary by 24 orders of magnitude, and are in general quite small. Such numeric properties are known to cause computational difficulties with commercial solvers.

near symmetry that is characteristic of our datasets (as opposed to *exact* symmetry) is especially difficult for solvers to detect and mitigate (Barnhart et al. 1998, Ostrowski et al. 2011, Margot 2002).

Most commercial LP solvers leverage the *sparse* nature of a problem by considering only subsets of variables at a time. However, in our cumulative interference constraints (F4), a row may contain hundreds of nonzero coefficients. That is, a given radio s on channel c may experience interference from dozens or hundreds of other radios assigned to the same channel. Thus the overall constraint matrix is much more dense than if we considered only pairwise interference constraints, i.e.,

$$interference^{c}_{rs} Z^{c}_{rs} \leq max\_interference^{c}_{s} \quad \forall\, (r,s) \in W, c \in C. \tag{4.13}$$

The system of linear equations formed by these constraints would be very sparse, i.e., each row may contain only one nonzero coefficient (representing two radios from different units assigned to the channel); all other column entries would be zero. These pairwise interference constraints can be handled very efficiently by IP and constraint satisfaction solvers.

Unfortunately, these pairwise constraints alone do not adequately represent the real-world problem. Specifically, the assignment of a radio to a particular channel may not result in excessive interference from any other single radio assigned to a different unit, but that radio may very well likely receive excessive cumulative interference from all the other radios assigned to different units but on the same channel (also observed by Garcia Villegas et al. (2005)).

To illustrate this in the context of the MEF scenario (our largest dataset), for each of the roughly 1800 radios we sum the total interference received from all other radios not assigned to the same unit. We then calculate the total percentage of interference that is captured by the single largest source of interference, i.e., that interference that would be avoided via a pairwise constraint. Ideally this is a large percentage, indicating that we can use pairwise constraints to reasonably represent co-channel interference. Figure 4.3 presents the results for each radio, where the vertical axis displays the percentage of total interference. On average, the single largest source of interference (blue line) accounts for 73.4% of total interference received by each radio. However, for about 34% of radios, this single source only accounts for half or less of total received interference. By considering the strongest ten sources (black line in Figure 4.3), on average 95.1% of interference is captured, and for less than four percent of radios would these ten sources capture less than 50% of total interference. This illustrates that pairwise interference constraints may fail to capture a large portion of the total interference received by most radios, and thus if used alone, may inadequately represent the real-world problem.

We find in our scenarios that considering only pairwise interference constraints will cause at least a few radios to be disconnected from their respective MANETs. "Repairing" these disconnections, i.e., ensuring all radios in each unit are connected, is what makes

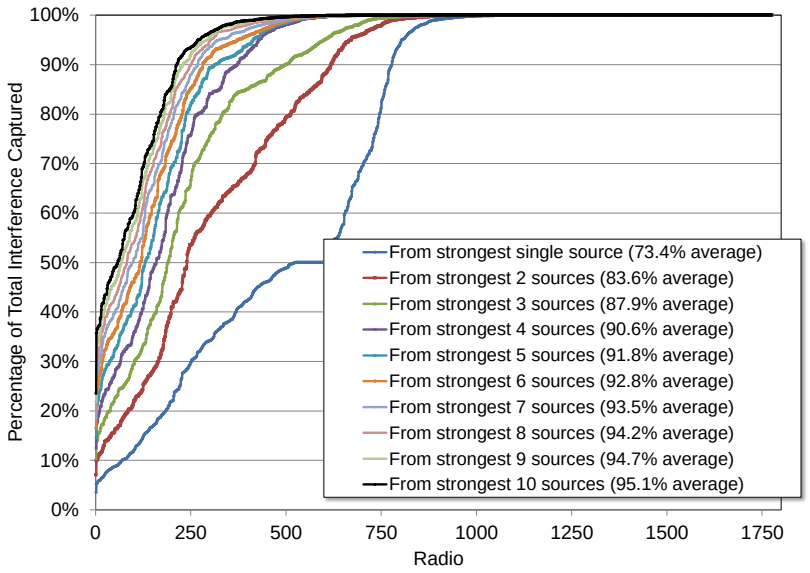


Figure 4.3: Percentage of total interference captured by considering the strongest sources of interference (for one to ten sources), for each radio in the full MEF scenario (time step one).

this problem particularly challenging. Figure 4.4 provides a visualization of the received interference and interference constraints for the first time step of the MEF scenario, solved using CPLEX and considering only pairwise constraints. Each black dot indicates the received interference at each radio, where the vertical axis indicates signal strength in dBm, and the horizontal axis follows the rank-ordered list of radios by interference (i.e., the radio receiving the least interference is on the extreme left). The red line (actually, collection of points) immediately above each radio indicates the $max\_interference_s^c$ threshold; a point above this line indicates a radio receives too much interference. There are nine radios that receive excessive interference and are thus unable to communicate, visible in the lower-left corner.

One can imagine trying to "push" these points under the line in Figure 4.4 by reassigning channels. The empty space under the red line in the upper-right corner might seem to indicate slack in the constraints, i.e., that reassigning the violated radios should be relatively

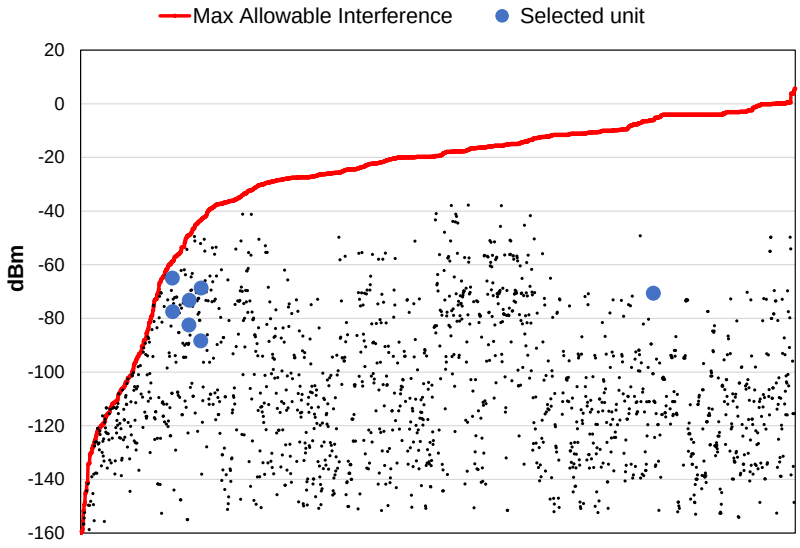


Figure 4.4: Depiction of the received interference (dots) and interference threshold (red line) for each of the 1887 radios in the MEF scenario (time step one), after being solved by CPLEX with only pairwise constraints. Nine radio fall above the line (in the lower-left corner), indicating constraint violations. Large blue dots represent seven radios assigned to one particular unit.

easy, given that some radios receive interference far below their respective thresholds. In practice, this is very computationally challenging, in part because radios within a particular unit are often spread across this diagram, i.e., the radios receive greatly different interference. To illustrate, the highlighted blue dots indicate radios from a single unit. While one radio (on the right) has considerable slack, several radios are very close to their respective thresholds, and thus cannot easily be reassigned to a different channel with other radios operating concurrently.

Despite these challenges, we find that even a simple "brute force" IP method (i.e., using CPLEX to solve the full problem as-is, without providing any initial solution or conducting preprocessing) is sufficient to solve the smaller two scenarios (i.e., MEU and MEB) to optimality (see Section 4.1.3). However, this approach fails to obtain useful answers to the MEF scenario, even after 60 hours of computation on a cluster of 14 high-performance

desktop computers. We use a variation of the Full Standard Formulation to address the computational problems imposed by the cumulative interference constraints (F4), including the vast differences in input parameter values and symmetry.

4.1.2 Relationships to Other Problems

We observe our MO-CAP is structurally similar to other NP-hard problems. These observations may be useful when exploring the use of heuristics or approximate algorithms to solve the problem or sub-problems.

Hypergraph Coloring. Binary interference constraints are often represented using an interference graph, where an edge connects two radios that may not be assigned the same channel. Our cumulative co-channel interference constraints can be represented using a hypergraph (Berge 1984), where a hyper-edge or hyper-arc may connect more than two radios (as opposed to two radios defining a traditional edge or arc). Thus our problem may be represented as a generalized form of *hypergraph coloring* (Phelps and Rödl 1984, Brown 1996), which seeks the minimal numbers of colors such that no hyper-edge is *monochromatic*. In our problem, we have an additional complication in that different colors (i.e., channels) may perform differently because the associated radio frequencies may have different propagation properties, e.g., lower frequencies generally propagate farther than higher frequencies. In any case, hypergraph coloring is known to be a notoriously difficult problem, but this field may be worth exploring further to determine if there are algorithms or approaches that may benefit our research.

Set Covering. The MO-CAP is similar to the *set covering problem* in that we aim to select the minimal number of sets (i.e., channels) that cover all elements (i.e., units).

Bin Packing Problem. Our problem bears a superficial resemblance to the *bin-packing problem*, which attempts to pack items (i.e., units) into as few bins (i.e., channels) as possible.

However, our problem is different and more complex in that capacities are associated with each unit, not a channel, and the "space" occupied by a unit (i.e., the amount of interference it provides) depends on the other units assigned the same channel. This property also differentiates our problem from the classic *knapsack problem*.

4.1.3 MO-CAP FSF Preliminary Solution Method and Results

We pre-process all $interference^{c}_{rs}$ and $max\_interference^{c}_{s}$ values, as described in Section 3.3. We use Python 2.7 and Pyomo (Hart et al. 2011, 2012) to create a problem instance, and initially attempt to solve it using CPLEX without any further processing, i.e., a "brute force" approach. We obtain the following results:

- MEU scenario (6 units, 131 radios): Solves to optimality in less than two seconds.
- MEB scenario (24 units, 641 radios): On certain time steps, solves to optimality in five seconds. On one time step, finds feasible solution but fails to converge after 24 hour of computation.
- MEF scenario (118 units, 1887 radios). Fails to find a feasible solution, even after two weeks of processing on a 14-computer cluster of high-performance desktops running a distributed version of CPLEX.

These preliminary results clearly indicate we need a better method than just "throwing" the full, original problem at CPLEX. For the rest of this analysis, we focus solely on the MEF scenario (at each of 20 time steps), as it presents the most interesting computational challenge.

4.1.4 MO-CAP Greedy Heuristic Solution Method

In an attempt to improve the solution process, we create a simple greedy heuristic to find and provide an initial feasible solution to the solver. The heuristic iteratively "packs" units onto channels until the channel is full, and then starts with the next channel. This constructive

heuristic guarantees a feasible solution (as long as the number of available channels is at least as large as the number of units, i.e., $|C| \geq |U|$), but provides no *certificate of optimality*. The following pseudo-code describes this heuristic:

Algorithm *Pack Channels*

Input: Number of units requiring channels U, $max\_interference_s^c, \forall s \in R, c \in C$

Output: $X_u^c, \forall u \in U, c \in C; Y^c, \forall c \in C$

begin

$currentChannel \leftarrow 0$

$numberAssignedUnits \leftarrow 0$

for $r \in R, c \in C$

$interferenceMargin_r^c \leftarrow max\_interference_r^c$

next;

for $i = 1, 2, \ldots, |U|$

Assign individual channel to any unit that cannot share channels

$numberAssignedUnits \leftarrow numberAssignedUnits + 1$

next;

while ($numberAssignedUnits < |U|$) **do**

$currentChannel \leftarrow$ next available channel

if ($|U| - numberAssignedUnits > 2$)

$nextUnit \leftarrow$ unassigned unit that receives least interference from all other unassigned units

else

$nextUnit \leftarrow$ the first remaining unit

endif;

$X_{nextUnit}^{currentChannel} \leftarrow 1$

$eligibleUnits \leftarrow$ ***Calculate Eligible Units*** ($currentChannel$)

while ($|eligibleUnits| > 0$) **do**

$weakestUnit \leftarrow$ the unit already assigned to $currentChannel$ with smallest remaining $interferenceMargin_r^{currentChannel}$

$leastInterferer \leftarrow$ the $eligibleUnit$ that least interferes with $weakestUnit$

$eligibleUnits \leftarrow eligibleUnits \backslash leastInterferer$

$X_{leastInterferer}^{currentChannel} \leftarrow 1$

$numberAssignedUnits \leftarrow numberAssignedUnits + 1$

Update $interferenceMargin_r^{currentChannel}$

$eligibleUnits \leftarrow$ ***Calculate Eligible Units*** ($currentChannel$)

end;

$Y^{currentChannel} \leftarrow 1$ // No more eligible units; $currentChannel$ is "packed"

end;

end;

The following function supports **Algorithm *Pack Channels*** by determining the units that are eligible to be assigned to the given channel, considering interference constraints:

Function *Calculate Eligible Units* (*givenChannel*)

```
// Calculate and return eligibleUnits (the set of units eligible for assignment) for the
givenChannel
begin
    eligibleUnits ← { }
        for each unassigned unit u
            if u can be assigned to givenChannel and not cause unacceptable co-channel
                interference
                eligibleUnits ← eligibleUnits ∪ u
            endif;
        next;
end;
return eligibleUnits;
```

4.1.5 MO-CAP Greedy Heuristic Results

We use our heuristic to solve each time step of the MEF scenario; the results are displayed in Table 4.1. The heuristic runs quickly, but it provides no indication of the goodness of each solution, as it does not provide a lower bound or measure of the optimality gap.

In a further attempt to reduce the computational load on CPLEX, we make some additional assumptions. First, we assume we can ignore any source of interference that is less than 1/50th of the maximum allowable by a given receiver. That is, we assume that even if all other units operating at or below this 1/50th level are communicating on the same channel as a given radio, that radio would not be significantly affected. This assumption reduces the size of the input data file from 1.42 GB to 152 MB. (However, in general this assumption is not valid, as we find solutions using our other techniques where this level of interference affects the results.) We also assume that all channels provide the same level of performance, regardless of frequency band. In practice we have found this to make only a negligible difference in the quality of the results. This assumption furthers reduces the input file size to 1.2 MB. We then use the initial feasible solution provided by our heuristic

Table 4.1: Performance results of the MO-CAP greedy heuristic by time step.

Time step	**Solution value**	**Runtime (s)**
1	51	292.66
2	48	350.75
3	46	340.53
4	47	379.69
5	43	339.25
6	51	333.76
7	49	358.94
8	42	342.33
9	43	371.08
10	49	348.66
11	45	354.01
12	43	298.93
13	43	311.49
14	43	321.6
15	49	409.84
16	47	358.38
17	49	328.1
18	40	324.81
19	40	297.56
20	49	345.82
Average:	**45.9**	**340.41**

and attempt to solve the MEF problem using a 14-computer cluster of high-performance desktops.

We find that after 60 hours of runtime on a single time step, CPLEX improves upon the initial feasible solution (providing a reduction of over 18% in the number of channels), but the solution has an optimality gap of 77%. This indicates that our heuristic may not be providing very good solutions (as in this instance it could be improved by over 18%), and that we require more sophisticated methods if we are going to solve realistic instances of this problem over multiple time steps.

4.2 MO-CAP Restricted Standard Formulation

We next describe a *Restricted Standard Formulation* (*RSF*) to enable us to preprocess the cumulative interference constraints (F4) and alleviate the numerical issues associated with the $interference^c_{rs}$ and $max\_interference^c_s$ values.

Given a dataset, we preprocess the interference constraints to create simplified and more computationally tractable *packing constraints*. For example, suppose two specific nodes r and s (not assigned to the same unit) are not both allowed to be assigned to channel c because to do so would violate the associated interference constraint (i.e., Figure 4.1b). This may be represented as:

$$X^c_r + X^c_s \leq 1. \tag{4.14}$$

We use Python and the `mpmath` library (Johannson et al. 2013), which allows the use of arbitrary-precision floating point mathematics, to identify unacceptable pairs of radios and handle the extremely small interference values present in our realistic data sets. Figure 4.5 (created using Gephi (Gephi Consortium 2016, Bastian et al. 2009)) displays all the pairwise interference constraints, i.e., the interference graph, for the first time step of the MEF scenario.

Among these pairwise interference constraints, we identify and constrain the *maximum clique*, which is the largest *maximal clique* (i.e., *complete sub-graph*) formed from among the pairwise interference constraints (i.e., the interference graph). We use the `NetworkX` Python library (Hagberg et al. 2008) to find the maximum clique, which relies on the algorithm of Bron and Kerbosch (1973) as adapted by Tomita et al. (2006). Figure 4.6 depicts in red the maximum clique among the pairwise constraints for the first time step of the MEF scenario. Let $M \subset U$ be the subset of units in the maximum clique. The maximum clique is constrained by:

$$\sum_{u \in M} X^c_u \leq 1 \qquad \forall c \in C. \tag{4.15}$$

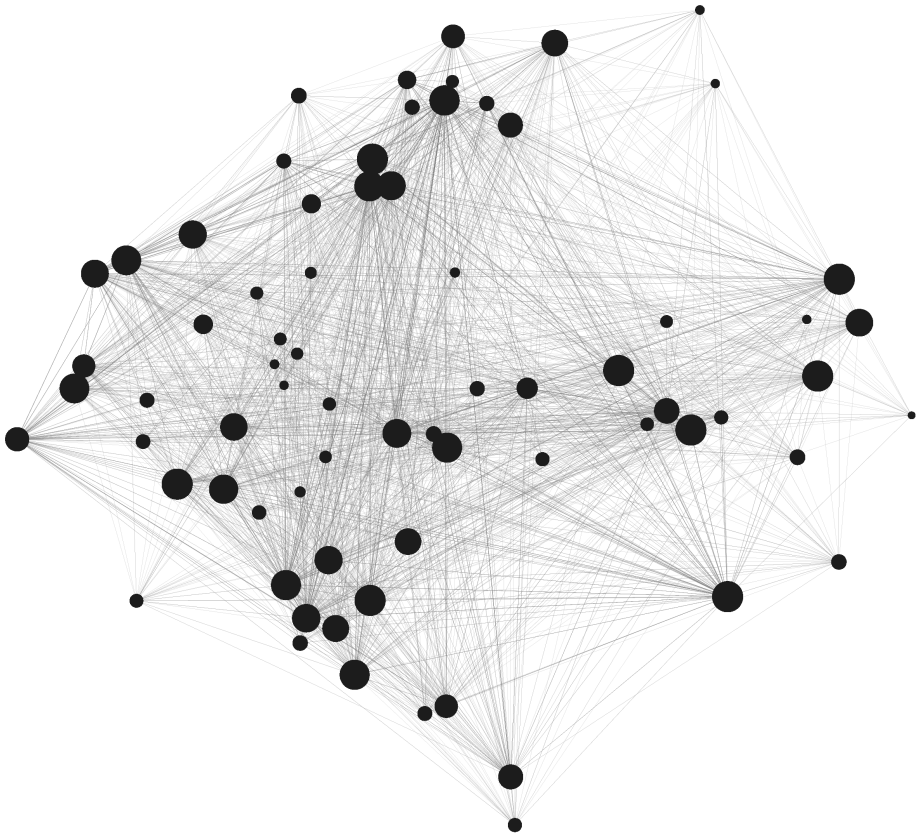


Figure 4.5: Depiction of pairwise constraints in MEF time step one, where an arc indicates that the two associated units cannot be assigned the same channel at the same time, and the size of each node is relative to the degree of the node.

That is, only one unit in the clique may be assigned any given channel. Cutting off such a clique more efficiently constrains the problem than a series of pairwise constraints, as a larger portion of the solution space can be excluded and the cut face will be closer in proximity to the integer optimal solution. After we add the maximum clique, we then add to the list of constraints all remaining pairwise constraints, i.e., those pairs that are not included in the clique.

To generalize for larger n-tuples of units above pairs (triplets, quadruplets, etc.) (e.g., Figure 4.1c), let $S \subset U$ be a subset of units that cannot all be assigned to the same channel

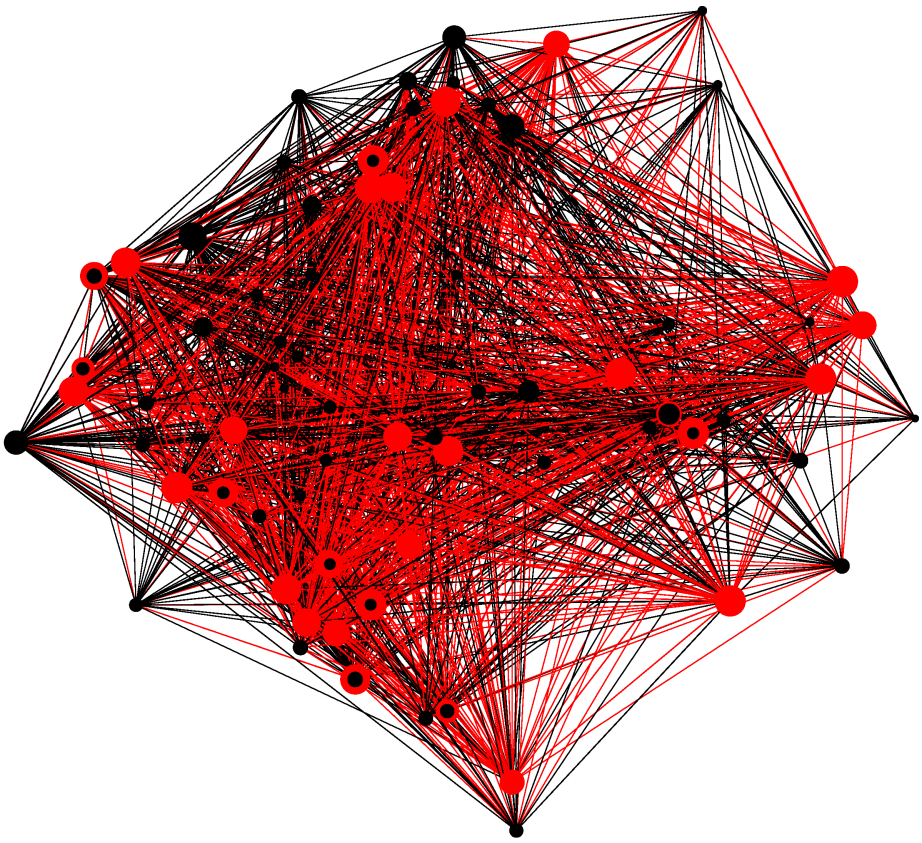


Figure 4.6: Depiction of pairwise constraints in MEF time step one, with the maximum clique (comprising 46 units) depicted in red.

c. We can represent such a restriction of assignments as

$$\sum_{u \in S} X_u^c \leq |S| - 1 \qquad \forall c \in C. \tag{4.16}$$

Preprocessing all such unacceptable combinations and adding them as constraints would effectively replace the cumulative co-channel interference constraints (F4). However, identifying all unacceptable combinations would be very computationally costly (as they grow

exponentially in number with both the number of units and available channels) and unnecessary, as many combinations will be redundant and/or represent negligible levels of co-channel interference.

Instead, we dynamically add these higher-order constraints to the formulation only as needed via *lazy constraints*, which are constraints which are feasible in the full version of the problem, but are only checked on an as-needed basis (IBM 2013a). When the solver obtains a feasible solution, it will check the feasibility of the solution in the full problem, and if one or more infeasibilities exist, it will add constraints to further constrain the problem.

This approach avoids the problem of very small numbers in CPLEX, as we can process the constraints outside of the solver (e.g., in Python), and then add the much-simplified packing constraints (4.16) dynamically. Also, since the solver is no longer required to calculate cumulative interference, the formulation no longer requires the index $r \in R$ or variables Z^c_{rs}, greatly reducing the number of decision variables in the problem. Our MO-CAP Restricted Standard Formulation (RSF) is summarized as follows:

MO-CAP Restricted Standard Formulation

Index and Set Use

$u \in U$	unit
$c \in C$	channel
$M \subset U$	maximum clique formed from pairwise interference constraints
$S \subset U, S \in P$	a dynamically-generated subset of units that cannot be assigned the same channel c, for all such generated subsets P

Decision Variables

X_u^c	binary variable indicating whether unit u is using c
Y^c	binary variable indicating whether channel c is being used

Formulation

$$\min_{X,Y} \quad \sum_{c \in C} Y^c \tag{R0}$$

$$s.t. \quad X_u^c \leq Y^c \qquad \forall u \in U, c \in C \tag{R1}$$

$$\sum_{c \in C} X_u^c = 1 \qquad \forall u \in U \tag{R2}$$

$$\sum_{u \in M} X_u^c \leq 1 \qquad \forall c \in C \tag{R3}$$

$$\sum_{u \in S} X_u^c \leq |S| - 1 \qquad \forall S \in P, c \in C \tag{R4}$$

$$X_u^c \in \{0, 1\} \qquad \forall u \in U, c \in C \tag{R5}$$

$$Y^c \in \{0, 1\} \qquad \forall c \in C \tag{R6}$$

The MO-CAP RSF is a pure 0-1 integer program. The objective function (R0) minimizes the sum of assigned channels. Constraints (R1) ensure that each channel assigned to a unit is counted toward the objective function. Constraints (R2) require the assignment of one channel to each unit. Constraints (R3) enforce the maximum clique cut formed among the pairwise interference constraints. Constraints (R4) are packing constraints for dynamically-generated subsets S of units that cannot co-occupy a channel, for all such subsets P.

4.2.1 MO-CAP RSF Solution Method

After building an initial problem instance with the maximum clique and pairwise constraints with Python and Pyomo, we send the problem to CPLEX via the Python API and indicate

to the solver that we wish to initiate lazy constraints *callbacks*. Upon finding a solution that is feasible (with the current constraints), the solver runs our lazy constraint callback code (written in Python). The code checks the feasibility of the current solution in the full problem; this can be calculated in polynomial time, specifically $\mathcal{O}\left(|R|^2|C|\right)$. If infeasibility exists, we add the lowest-order packing constraints (R4) to prevent the same units from being assigned the same channel again. CPLEX then continues the search process with these new constraints added into the formulation. The process repeats until optimality is achieved or a time limit is reached.

The following pseudo-code describes our algorithm for solving the MO-CAP Restricted Standard Formulation using lazy and maximum clique constraints:

Algorithm *MO-CAP RSF*

Input: MO-CAP problem; MO-CAP initial feasible solution (if desired); *max_time*

Output: MO-CAP solution, objective value, and optimality gap

begin
    Preprocess and identify all pairwise interference constraints
    Initialize CPLEX problem instance
    Calculate maximum clique and add clique constraints to CPLEX problem instance
    Add remaining pairwise constraints to CPLEX problem instance
    Add initial feasible solution to CPLEX problem instance (if desired)
    while *time* < *max_time* **do**
        Run CPLEX
        if CPLEX finds new feasible RSF solution
            Check feasibility of current solution in MO-CAP FSF
            if current solution is not feasible in MO-CAP FSF
                Identify *violations* of co-channel interference constraint(s) (F4)
                packingConstraints ← ***Calculate Packing Constraints*** (*violations*)
                Add *packingConstraints* to CPLEX problem instance
            endif;
        endif;
        Continue CPLEX search process
    end;
end;

The following function supports **Algorithm *MO-CAP RSF*** by finding the lowest-order packing constraints, given violation(s) in the original MO-CAP FSF (i.e., radios that receive excessive co-channel interference):

Function *Calculate Packing Constraints* (*violations*)

```
// Calculate and return packingConstraints (the lowest-order packing constraints), given
violations indicating radios that receive excessive co-channel interference
begin
    unitList ← { }
    packingConstraints ← { }
        for each violation
            Add that radio's associated unit to unitList
        next;
        for unit ∈ unitList
            tempList ← { }
            channel ← channel assignment of unit
            for u ∈ U, u ≠ unit
                if channel assignment of u = channel
                    tempList ← tempList ∪ u
                    unitList ← unitList\u
                endif;
            next;
            tupleSize ← 3
            while True
                for each combination in tempList of size tupleSize:
                    if combination is an interference violation (F4)
                        newConstraint ← {combination_1 + combination_2 + ··· +
                            combination_|tupleSize|} ≤ |tupleSize| − 1
                        packingConstraints ← packingConstraints ∪ newConstraint
                        break;
                    endif;
                    tupleSize ← tupleSize + 1
                next;
            end;
        next;
return packingConstraints;
```

We provide partial programming code in Python to solve the MO-CAP RSF using Pyomo and CPLEX in Appendix B. The following sections describe our results using lazy constraints, the use of an initial feasible solution (provided via our heuristic), and the use of lazy constraints and the maximum clique constraints.

4.2.2 MO-CAP RSF Lazy Constraint Results

We use the restricted standard formulation of the MO-CAP to add to the problem all pairwise interference constraints, and then dynamically add interference constraints using lazy constraints (without providing an initial solution). Table 4.2 displays results for each time step in the MEF scenario, including the number of pairwise constraints, the number of lazy constraints (and the order of the highest-order lazy constraint), and solution results. Each time step is run for 9,000 seconds, or until optimality is obtained. The times in Table 4.2 indicate solver time when the displayed solution value and optimality gap is obtained; those time steps with a non-zero optimality gap fail to converge within 9,000 seconds.

Table 4.2: MO-CAP results by time step in the MEF scenario using pairwise and lazy constraints, without an initial feasible solution. "Time" indicates the time at which the displayed solution and optimality gap is obtained, during a total runtime of 9,000 seconds.

Time Step	Number pairwise constraints	Number lazy constraints	Highest-order lazy constraint	Solution value	Lower Bound	Gap	Time (s)	Improvement over heuristic
1	4407	49	5	46	45	2.17%	1356.53	9.80%
2	3892	25	5	37	37	0%	1333.92	22.92%
3	3945	87	6	36	34	5.56%	4432.53	21.74%
4	3823	62	5	34	32	5.88%	7828.09	27.66%
5	3762	9	5	33	33	0%	678.23	23.26%
6	3904	104	6	36	35	2.78%	4086.04	29.41%
7	3884	67	5	37	37	0%	1737.45	24.49%
8	3538	57	5	31	29	6.45%	8614.79	26.19%
9	3398	21	8	32	32	0%	271.19	25.58%
10	3541	0	0	34	34	0%	248.16	30.61%
11	3449	121	11	33	32	3.03%	5997.82	26.67%
12	3367	29	5	36	35	2.78%	927.38	16.28%
13	3550	104	6	32	31	3.12%	2510.22	25.58%
14	3301	69	6	31	30	3.23%	1780.48	27.91%
15	3666	147	6	38	37	2.63%	1669.23	22.45%
16	3749	119	8	36	34	5.56%	4194.86	23.40%
17	3721	128	6	37	36	2.70%	3092.38	24.49%
18	3214	8	4	31	31	0%	245.20	22.50%
19	3282	99	5	30	29	3.33%	1673.56	25.00%
20	3660	47	5	37	37	0%	1268.30	24.49%
Aver:	**3652.7**	**67.6**	**5.6**	**34.9**	**34.0**	**2.46%**	**2697.32**	**24.02%**

The lazy constraint approach to solving the MO-CAP yields results far superior to our previous methods. The solutions are on average 24% lower than the heuristic, and each solution has an associated optimality gap. Bolded solution values indicate better solution values than that provided via the heuristic, which is the case in every time step. On seven time steps, optimality is achieved. While generally slower than the greedy heuristic, this method finds solutions within one or two channels of optimality within an average of about 45 minutes, which is not unreasonable for our projected use case.

We note that the number of required channels is considerably higher in the first time step than in any other time step. This occurs because within the MEF amphibious assault scenario, the units have just reached the beach at the first time step and are relatively close to one another (see Figure 3.3). Thereafter, the units spread apart and are better able to leverage channel reuse.

4.2.3 MO-CAP RSF Results Using an Initial Feasible Solution

Next, we examine whether the use of an initial feasible solution improves the performance of the lazy constraint method. We provide the output of the greedy heuristic as an initial solution to CPLEX; the results are displayed in Table 4.3.

We observe no qualitative difference in the solutions obtained when we provide the solver an initial feasible solution: the solution values are the same, and the runtimes are very similar. This indicates that the solutions found with the heuristic are of little use to CPLEX. For the remainder of this analysis, we do not consider the heuristic solutions.

4.2.4 MO-CAP RSF Lazy Constraints and Maximum Clique Results

In an attempt to further reduce the optimality gap, we build on our lazy constraint method by adding the maximum clique formed among the pairwise interference constraints. We then add all remaining pairwise constraints and dynamically add lazy constraint callbacks. The results are displayed in Table 4.4.

Table 4.3: MO-CAP results by time step in the MEF scenario using pairwise and lazy constraints, with an initial feasible solution. "Time" indicates the time at which the displayed solution and optimality gap is obtained, during a total runtime of 9,000 seconds.

Time Step	Number pairwise constraints	Number lazy constraints	Highest-order lazy constraint	Solution value	Lower Bound	Gap	Time (s)	Improvement over heuristic
1	4407	49	5	46	45	2.17%	1352.19	9.80%
2	3892	25	5	37	37	0%	1362.26	22.92%
3	3945	122	5	36	34	5.56%	4793.59	21.74%
4	3823	62	5	34	32	5.88%	7738.66	27.66%
5	3762	9	5	33	33	0%	670.54	23.26%
6	3904	103	6	36	35	2.78%	4002.05	29.41%
7	3884	68	5	37	37	0%	1723.43	24.49%
8	3538	57	5	31	29	6.45%	8641.12	26.19%
9	3398	21	8	32	32	0%	271.18	25.58%
10	3541	0	0	34	34	0%	248.10	30.61%
11	3449	121	11	33	32	3.03%	5879.65	26.67%
12	3367	48	5	36	35	2.78%	919.39	16.28%
13	3550	105	6	32	31	3.12%	2492.53	25.58%
14	3301	70	6	31	30	3.23%	1737.95	27.91%
15	3666	147	6	38	37	2.63%	1690.00	22.45%
16	3749	52	5	36	34	5.56%	4242.65	23.40%
17	3721	128	6	37	36	2.70%	3075.19	24.49%
18	3214	8	4	31	31	0%	244.39	22.50%
19	3282	116	5	30	29	3.33%	1667.00	25.00%
20	3660	57	5	37	37	0%	1248.50	24.49%
Aver:	**3652.7**	**68.4**	**5.4**	**34.9**	**34.0**	**2.46%**	**2700.019**	**24.02%**

Bolded values indicate an improvement over the previously-described technique. Again, each time step is run for 9,000 seconds, or until optimality is obtained, and "Time" indicates solver time when the displayed solution value and optimality gap is obtained. Overall, inclusion of the maximum clique reduces average runtime to obtain solutions within one channel of optimality. On time step 3, this method obtains a solution that requires one less channel than that identified without the use of the maximum clique. On eight time steps, this method reduces the known optimality gap, and on 12 time steps, it obtains the provably-optimal solution (five more than the previous method). It is interesting to note that the size of the maximum clique (which itself provides a lower bound on the number of required channels) is within one of the best known solution, for each time step. This is

Table 4.4: MO-CAP results by time step in the MEF scenario using pairwise and lazy constraints, and a maximum clique constraint, without an initial feasible solution. "Time" indicates the time at which the displayed solution and optimality gap is obtained, during a total runtime of 9,000 seconds.

Time Step	Max Clique Size	Number pairwise constraints	Number lazy constraints	Highest-order lazy constraint	Sol'n value	Lower Bound	Gap	Time (s)	Improvement over heuristic
1	46	3372	45	6	46	46	**0%**	552.68	9.80%
2	37	3226	4	4	37	37	0%	273.40	22.92%
3	34	3384	143	7	**35**	35	**0%**	4338.21	**23.91%**
4	33	3295	85	6	34	33	**2.94%**	3831.19	27.66%
5	33	3234	10	3	33	33	0%	1010.00	23.26%
6	35	3309	95	6	36	35	2.78%	3128.34	29.41%
7	37	3218	13	6	37	37	0%	266.68	24.49%
8	30	3103	45	5	31	30	**3.23%**	4415.48	26.19%
9	32	2902	2	4	32	32	0%	226.37	25.58%
10	34	2980	6	4	34	34	0%	323.08	30.61%
11	33	2921	42	8	33	33	**0%**	856.69	26.67%
12	35	2772	30	5	36	35	2.78%	1577.96	16.28%
13	31	3085	131	6	32	31	3.12%	3172.95	25.58%
14	30	2866	214	9	31	30	3.23%	2702.16	27.91%
15	38	2963	105	6	38	38	**0%**	1047.00	22.45%
16	35	3154	16	5	36	35	**2.78%**	600.91	23.40%
17	36	3091	89	5	37	36	2.70%	1495.15	24.49%
18	31	2749	13	4	31	31	0%	322.90	22.50%
19	30	2847	74	6	30	30	**0%**	1653.29	25.00%
20	37	2994	33	4	37	37	0%	1387.28	24.49%
Aver:	**34.4**	**3073.3**	**59.8**	**5.5**	**34.8**	**34.4**	**1.18%**	**1659.09**	**24.13%**

indicative of the power of the maximum clique constraint, which very efficiently cuts off a significant portion of the solution tree. It is also interesting to note that CPLEX did not exploit this clique structure until we manually provided it to the solver.

To provide a qualitative sense of the results of the MO-CAP RSF, we generate Figure 4.7 using Gephi (Gephi Consortium 2016, Bastian et al. 2009) and the MO-CAP solution for the first time step of the MEF scenario using the maximum clique and lazy constraints. Color indicates channel assignment. Arcs connect those units assigned the same channel, and the size of each node is relative to the degree of that associated unit.

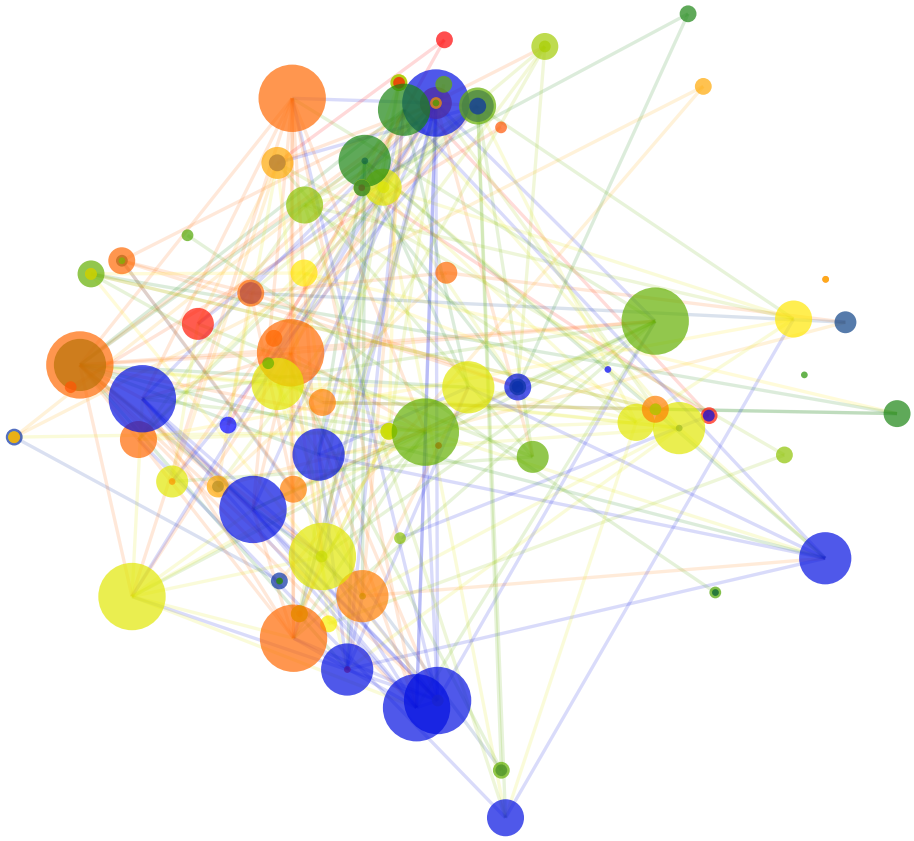


Figure 4.7: Depiction of the MO-CAP RSF solution at time step one. Color indicates channel assignment. Arcs connect those units assigned the same channel, and node size is relative to the degree of the associated unit.

4.3 MO-CAP Constraint Programming Formulation

Constraint programming (CP) is often used to complement IP approaches. We reformulate MO-CAP as a CP problem to attempt to quickly find lower bounds to the problem. We use the Optimization Programming Language (OPL) (Van Hentenryck 1999) to formulate the problem using integer variables, where each variable $W_u \in C$ indicates the channel that unit $u \in U$ is assigned, and the domain of each variable is equal to the number of available channels $|C|$. (We originally formulate this problem using binary variables, but find that the

CP solver is much less efficient in finding solutions using binary variables for this particular problem.)

We add all pairwise constraints to the problem, indicating that two given units u and v are not allowed to be assigned the same channel, for all pairs $(u, v) \in A$:

$$W_u \neq W_v \qquad \forall\,(u, v) \in A. \tag{4.17}$$

We also identify the maximum clique M, and add the constraint using the CP constraint type `allDifferent`, which requires that all variables be assigned pairwise different values (Régin 1994, Puget 1998, Mehlhorn and Thiel 2000):

$$\texttt{allDifferent}\left(\left[W_{u=1}, W_{u=2}, \ldots, W_{u=|M|}\right]\right). \tag{4.18}$$

Our CP formulation of the MO-CAP is summarized as follows:

MO-CAP Constraint Programming Formulation

Index and Set Use			
	$u \in U \subset N$	unit (*alias* v)	
	$c \in C$	channel	
	$(u, v) \in A$	arc indicating u and v cannot occupy the same channel	
	$M \subset U$	maximum clique formed from pairwise interference constraints	
Decision Variables			
	W_u	integer variable indicating the channel assignment of unit u	
Formulation			
s.t.	$W_u \neq W_v$	$\forall\,(u, v) \in A$	(C1)
	$\texttt{allDifferent}\left(\left[W_{u=1}, W_{u=2}, \ldots, W_{u=\lvert M\rvert}\right]\right)$		(C2)
	$W_u \in C$	$\forall u \in U$	(C3)

The MO-CAP CP formulation attempts to find a feasible assignment of integer values for the variables $W_u \in C$. Constraints (C1) represent the pairwise interference constraints, and constraint (C2) represents the maximum clique constraint. Note this is a *relaxation* of the Full Standard Formulation in that only pairwise constraints are considered.

4.3.1 MO-CAP CP Solution Method

To solve the problem, we use the IBM ILOG CPLEX CP Optimizer (IBM 2016). We decrease the number of available channels $|C|$ (i.e., the domain of each W_u) until the solver determines that the problem is infeasible, or until a maximum time limit is reached (i.e., infeasibility is not detected). If this relaxation of the original problem is infeasible with the given number of channels, then we have established that the corresponding Full Standard Formulation problem (with all constraints) is also infeasible. This indicates that at least $|C| + 1$ channels are required, establishing a lower bound. If the lower bound equals the upper bound (obtained using CPLEX and the Restricted Standard Formulation), we have obtained an optimal solution.

We provide partial Python and OPL code for solving the MO-CAP using constraint programming in Appendix C. The following pseudo-code describes our algorithm:

Algorithm *MO-CAP CP*

Input: MO-CAP problem; starting value of $|C|$; *max_time*

Output: *min_channels* (lower bound on $|C|$)

```
begin
    Preprocess and identify all pairwise interference constraints
    Initialize CP problem instance
    Calculate maximum clique and add clique constraints to CP problem instance
    Add remaining pairwise constraints to CP problem instance
    avail_channels ← |C|
    while time < max_time do
        Run CP
        if CP determines feasible
            avail_channels ← avail_channels − 1
        else
            min_channels ← avail_channels + 1
            exit while;
        endif;
        Continue CP search process
    end;
end;
```

4.3.2 MO-CAP CP Results

The results of solving our MO-CAP CP formulation are displayed in Table 4.5, where "Infeasible" indicates the largest value at which CPLEX CP Solver detects infeasibility, i.e., at least one more channel is required for the problem to be feasible. "Optimal solution?" indicates whether the obtained value proves the optimality of a solution (i.e., no gap between this solution and that provided in the previous methods), where bolded values indicate new lower bounds (i.e., not found in the previous analyses).

Table 4.5: MO-CAP results by time step in the MEF scenario using constraint programming.

Time Step	**Infeasible**	**Optimal solution?**
1	45	Yes
2	36	Yes
3	33	
4	32	
5	32	Yes
6	34	
7	36	Yes
8	29	
9	31	Yes
10	33	Yes
11	32	Yes
12	34	
13	31	**Yes**
14	29	
15	37	Yes
16	34	
17	36	**Yes**
18	30	Yes
19	29	Yes
20	36	Yes

While the CP solver does not find the exact lower bound at each time step, it does establish two new exact lower bounds (at time steps 13 and 17). When infeasibility is detected by the solver, it is detected extremely quickly (less than a tenth of a second in each case). This provides great utility when an approximate lower bound is needed quickly.

On the other time steps, we are unable to tighten the lower bound, even after considerable runtimes (over 12 hours) and the addition of *symmetry-breaking constraints*. A high degree of symmetry exists in this problem in that many different solutions (i.e., assignments of channels to units) have the same objective value. For example, the channel assignments of any two units may be swapped without changing the objective. Symmetry-breaking constraints reduce or eliminate this possibility, and may (though with no certainty) provide better CP performance (IBM 2016). Following Ramani et al. (2004), we add such constraints to the MO-CAP CP problem instance, but in each case, we fail to improve the ability of the solver to find a tighter lower bound.

We also try adding all triplet and maximum clique constraints (via `allDifferent` constraints), as well as adding constraints iteratively in a sort of lazy-constraint approach, all to no avail. This "try and see" approach is common in constraint programming, as there are few general guidelines on the types of CP formulations that always provide good performance (Hooker and Ottosson 2003, Hooker 2011). We find in general that this CP approach is very efficient at finding infeasibilities (and thus establishing lower bounds), but may struggle to find a feasible solution close to the lower bound.

4.4 Summary of MO-CAP Results

The summary results from our MO-CAP analysis are displayed in Figures 4.8 and 4.9. Figure 4.8 displays the MO-CAP objective value from the use of the greedy heuristic, CPLEX with lazy constraints, CPLEX with lazy constraints and the maximum clique constraint, the best known lower bound, and the highest known infeasible solution found using constraint programming. Figure 4.9 displays the runtimes for each of the techniques. While the heuristic is overall the fastest technique, our CPLEX techniques provide certifiably-good solutions in reasonable amounts of time. The maximum clique technique allows us to achieve solutions within one channel of optimality for all time steps. In all but one time step (3), the constraint programming technique provides a lower bound within one channel

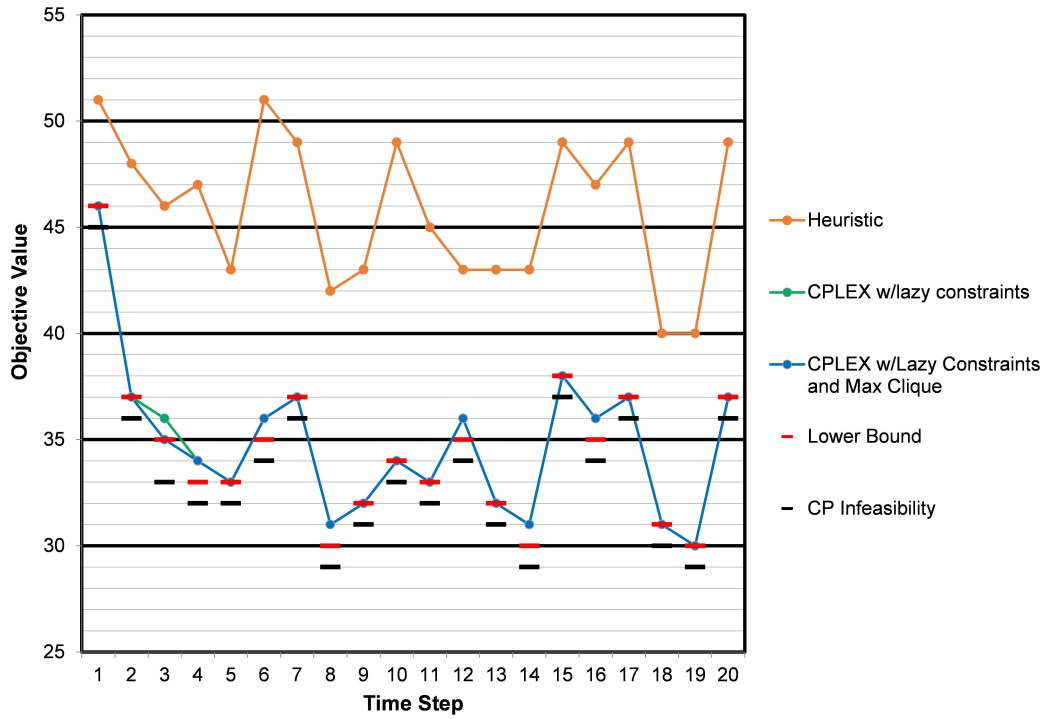


Figure 4.8: MO-CAP objective values and best known lower bound, for each time step in the MEF scenario, using various techniques.

of the best solution found using CPLEX, and does so extremely quickly (less than a tenth of a second).

4.5 MO-CAP Sensitivity Analysis

We next conduct sensitivity analysis on our MO-CAP RSF formulation and solution method to determine its robustness to small perturbations in inputs. Specifically, we randomly perturb our received signal strength values ρ_{rs} by up to $\pm 10\%$ (uniform random distribution), and then re-run our CPLEX method with lazy constraints and the maximum clique constraint, for each time step. In each case, we find that the number of required channels differs by at most one from the control case (i.e., no perturbation in input values) (left side of Table 4.6).

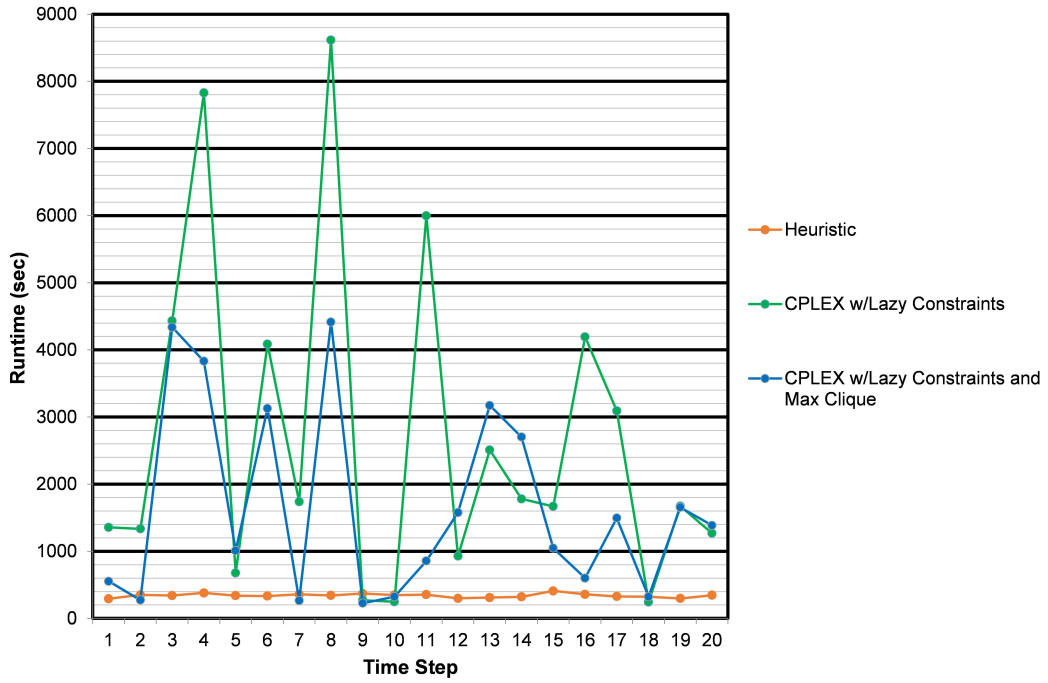


Figure 4.9: MO-CAP runtimes, for each time step in the MEF scenario, using various techniques.

These results indicate that our method is fairly robust to small perturbations in input values. This is not surprising, given the vast range in input values present in our data (see Section 4.1.1). This is encouraging from the perspective of our spectrum manager in that not even the highest-fidelity simulation of the radio environment will necessarily be perfect; this robustness to perturbation provides evidence that the computed solution objective values will not vary tremendously if the simulated values are slightly different than the real world.

We next wish to examine how much the solution itself (i.e., the assignment of channels to units) changes due to these perturbations. Note that we cannot simply penalize the assignment of a different channel number, as a *group* of units may remain assigned together but simply be assigned a different channel number, and the channel number itself is arbitrary. For example, suppose in the unperturbed control case our solution indicates that units i, j,

and k should be assigned channel 1. If we perturb our input values and our solution now indicates that these three units (and only these three units) should be assigned channel 2, we do not wish to penalize this difference.

We calculate the difference in *group membership* using a technique very similar to that described in Section 6.2, and present the results in the right side of Table 4.6, where each entry indicates the percentage of units that must be assigned to a new channel (compared to the unperturbed control case), for each time step and level of perturbation from $\pm 0.5\%$ to $\pm 10\%$. With even small levels of perturbation, we find a large percentage of units will be assigned to different groups. This is not surprising, given the vast symmetry in the problem. That is, it is frequently possible to swap group membership of many units with no effect on the objective value. In Chapter 6, we find this to be the case when moving from time step to time step, and develop a method to minimize the number of required channel changes.

Table 4.6: MO-CAP sensitivity analysis results for each time step, including the number of required channels and the percentage of units that must change channel (compared to the unperturbed control case), for given levels of random perturbation of input values.

	Number of Required Channels					**% Units Changing Groups**			
Time Step	**No Perturbation**	±0.5%	±1.0%	±5%	±10%	±0.5%	±1.0%	±5%	±10%
1	46	46	46	46	46	46.6%	50.0%	43.2%	44.9%
2	37	37	37	37	37	47.5%	43.2%	48.3%	43.2%
3	35	35	35	35	36	44.1%	44.1%	44.9%	45.8%
4	34	34	34	34	33	41.5%	36.4%	35.6%	44.9%
5	33	33	33	33	33	33.1%	44.9%	39.8%	41.5%
6	36	36	36	36	36	35.6%	33.1%	40.7%	46.6%
7	37	37	37	37	37	49.2%	50.0%	44.9%	49.2%
8	31	31	31	31	31	39.0%	48.3%	41.5%	48.3%
9	32	32	32	32	31	52.5%	52.5%	52.5%	50.8%
10	34	34	34	34	34	49.2%	42.4%	47.5%	50.0%
11	33	33	33	33	33	48.3%	49.2%	54.2%	50.0%
12	36	36	36	36	36	48.3%	48.3%	47.5%	50.0%
13	32	32	32	32	32	41.5%	40.7%	47.5%	43.2%
14	31	31	31	31	31	41.5%	40.7%	41.5%	43.2%
15	38	38	38	38	38	50.8%	49.2%	49.2%	48.3%
16	36	36	36	36	36	44.1%	46.6%	53.4%	50.0%
17	37	37	37	37	37	45.8%	44.9%	55.1%	50.0%
18	31	31	31	31	31	49.2%	52.5%	53.4%	53.4%
19	30	30	30	30	30	48.3%	50.8%	46.6%	45.8%
20	37	37	37	37	37	44.1%	44.1%	46.6%	42.4%
Average	**34.8**	**34.8**	**34.8**	**34.8**	**34.75**	**45.0%**	**45.6%**	**46.7%**	**47.1%**

Chapter 5: Minimum-Interference Channel Assignment Problem

This chapter describes the minimum-interference channel assignment problem (MI-CAP), which aims to minimize the total received interference given a fixed number of channels. As Gupta and Kumar (2000) show, minimizing interference is essential in maximizing wireless network performance. This problem reflects the real-world challenge of a spectrum manager being allocated less than the required number of channels identified by a MO-CAP solution. That is, the spectrum manager is now forced to make do with the channels available. We first develop a MI-CAP full standard formulation, and then present clustering, integer optimization, and constraint programming methods and their respective results.

In general, we assume that the number of channels available is less than that required by MO-CAP, i.e., there will necessarily be violations of the cumulative interference constraints (F4) for any time step that we have solved to optimality in the MO-CAP.

5.1 MI-CAP Full Standard Formulation

The MI-CAP Full Standard Formulation (FSF) is similar to the MO-CAP FSF, with the following modifications. We assume all available channels $|C|$ will be used, so the binary variable Y^c is no longer required, nor are constraints (F1) and (F9). To create the MI-CAP objective function, we can essentially relax the cumulative interference constraints from MO-CAP (F4). One possible objective function is:

$$\min \sum_{s \in R} \left(\sum_{c \in C} \sum_{(r,s) \in W} interference_{rs}^{c} Z_{rs}^{c} - max\_interference_{s}^{c} \right). \tag{5.1}$$

This objective function minimizes the difference between the cumulative received interference and maximum allowable interference. However, this function also provides a benefit when a radio receives less interference than $max\_interference_s^c$, which is not our intent and may skew the solver into seeking such solutions. Instead, consider the objective function:

$$\min \sum_{s \in R} \left(\sum_{c \in C} \sum_{(r,s) \in W} interference_{rs}^c Z_{rs}^c - max\_interference_s^c \right)_+ \tag{5.2}$$

where $(\cdot)_+$ denotes projection onto the non-negative real line. We define the inner quantity as *excessive interference*, which we wish to penalize. We introduce the nonnegative real variable E_s to represent excessive interference received by radio s, where:

$$E_s \geq \sum_{(r,s) \in W} interference_{rs}^c Z_{rs}^c - max\_interference_s^c \qquad \forall s \in R, c \in C. \tag{5.3}$$

That is, E_s is positive only when the received interference at s is greater than $max\_interference_s^c$. Our objective function thus becomes:

$$\min \sum_{s \in R} E_s. \tag{5.4}$$

The MI-CAP Full Standard Formulation follows:

MI-CAP Full Standard Formulation

Index and Set Use

$n \in N$	node (either radio or unit)
$r \in R \subset N$	radio (*alias s*)
$u \in U \subset N$	unit
$c \in C$	channel
$(r,u) \in L$	arc indicating logical assignment of radio $r \in R$ to unit $u \in U$
$(r,s) \in W$	arc indicating wireless interference between radios r and $s \in R$ where r and s are not in the same unit, i.e., $r,s \notin u, \forall u \in U$

Input Data

$interference^c_{rs}$	interference on $c \in C$ along arc $(r,s) \in W$ [watts]
$max\_interference^c_s$	max allowable interference $s \in R$ and $c \in C$ [watts]

Decision Variables

X^c_n	binary variable indicating whether n is using c
Z^c_{rs}	binary variable indicating whether r and s are both using c
E_s	nonnegative variable representing excessive interference received at $s \in R$ [watts]

Formulation

$$\min_{E,X,Z} \quad \sum_{s \in R} E_s \tag{M0}$$

$$\sum_{c \in C} X^c_u = 1 \qquad \forall u \in U \tag{M1}$$

$$X^c_r = X^c_u \qquad \forall c \in C, (r,u) \in L \tag{M2}$$

$$E_s \geq \sum_{r:(r,s) \in W} interference^c_{rs} Z^c_{rs} - max\_interference^c_s \qquad \forall s \in R, c \in C \tag{M3}$$

$$Z^c_{rs} \geq X^c_r + X^c_s - 1 \qquad \forall (r,s) \in W, c \in C \tag{M4}$$

$$Z^c_{rs} \leq X^c_r \qquad \forall (r,s) \in W, c \in C \tag{M5}$$

$$Z^c_{rs} \leq X^c_s \qquad \forall (r,s) \in W, c \in C \tag{M6}$$

$$E_s \geq 0 \qquad \forall s \in R \tag{M7}$$

$$X^c_n \in \{0,1\} \qquad \forall n \in N, c \in C \tag{M8}$$

$$Z^c_{rs} \in \{0,1\} \qquad \forall (r,s) \in W, c \in C \tag{M9}$$

The MI-CAP FSF is a *mixed integer program* (*MIP*). The objective function (M0) minimizes the total excessive interference. Constraints (M1) require the assignment of one channel to each unit. Constraints (M2) require that each radio uses the same channel as its assigned

unit. Constraints (M3) enforce the definition of E_s, and constraints (M4)-(M6) enforce the definition of Z^c_{rs}.

5.1.1 Computational Challenges of the MI-CAP FSF

Like the MO-CAP FSF, the MI-CAP FSF suffers from the problems of extremely small input values, which are beyond the precision of the solver to handle correctly. Also, we cannot simply pre-process and handle these numbers in the form of lazy constraints (as we did with the MO-CAP), as the troublesome values are required to calculate penalties in the objective function. Aardal et al. (2007) note that this property of the MI-CAP makes it in general more difficult to solve than other versions of the CAP. Rather than attempting to solve the problem as-is, we re-formulate and develop several methods of solving variations of the problem.

5.1.2 Estimating the Operational Impact of Interference

In general, minimizing co-channel interference is a worthy goal, but simply providing the amount of interference at each radio (e.g., in terms of watts or dBm) may not be enlightening to a decision-maker. In order to provide an estimate of the *operational impact* of excessive interference, we follow Nicholas et al. (2013b, 2016) and calculate *network availability*, defined as the number of radios that are able to communicate with their respective network control radio. This is perhaps the most fundamental metric of network performance, as without simple availability, two radios cannot communicate and few other measures of network performance will be non-zero.

Recall from Section 3.3 that we calculate $max\_interference^c_s$ by calculating the shortest path between a network control radio and each of its assigned radios. The value $max\_interference^c_s$ represents that interference strength that would disconnect radio s from its network control radio. In practice, such a disconnection may also disconnect other radios of the network that are dependent on radio s to reach the network control radio, or these other radios may have alternate paths to the network control radio (i.e., the MANET

may exhibit self-healing behavior). To capture and quantify these repercussions on network availability, we resolve the shortest path problem described in Section 3.3 using each MI-CAP solution.

We note that network availability may be a more complete measure of network performance than simply counting the number of radios that receive greater than $max\_interference_s^c$, since disconnected radios may disconnect other radios. For example, using our CP method (Section 5.4), we find that on average there are about seven times more unavailable radios than those radios exceeding their $max\_interference_s^c$ thresholds, indicating that in general the disconnection of a radio results in considerable impact on other radios within the same unit.

In the following sections, we use network availability as a metric in evaluating and comparing the performance of our clustering, IP, and CP methods.

5.2 MI-CAP Clustering Formulation

Our first formulation and solution method is based on the idea of clustering units into a given number of *groups* (equal to the number of available channels $|C|$), where the clustering metric is based on minimizing cumulative co-channel interference. Such a clustering approach is complicated by the interference relationships. When examining the assignment of any particular unit to a group, one must consider not only the pairwise interference between any two radios, but also the cumulative interference from all other radios in that group and the effect on the ability of a unit to communicate among its assigned radios (based on the $max\_interference_s^c$ value of each radio).

We simplify the problem as follows. We pre-process our dataset to allow us to consider pairwise interactions between units during the clustering process. For a given unit pair $(u, v) \in U$, we first calculate the *total normalized interference* received by each radio $s \in v$

from all radios in unit u, $r \in u$, on a given channel, i.e.,

$$total\_normalized\_interference_{su} \equiv \frac{\sum_{r \in u} interference^{c}_{rs}}{max\_interference^{c}_{s}}. \tag{5.5}$$

This allows us to equitably compare the interference received by different radios, where a value greater than one indicates excessive interference will be received by radio s if it is assigned the same channel as unit u. For each unit v we determine the *most sensitive radio* to u by examining the total normalized interference received at each $s \in v$, i.e., that radio s which most violates (or is closest to violating) its interference threshold if u is assigned the same channel:

$$\begin{aligned} most\_sensitive\_radio_{uv} &\equiv \arg\max_{s \in v} \left\{ \frac{\sum_{r \in u} interference^{c}_{rs}}{max\_interference^{c}_{s}} \right\} \\ &= \arg\max_{s \in v} \left\{ total\_normalized\_interference_{su} \right\}. \end{aligned} \tag{5.6}$$

We then use these resulting sensitivity values as *distances* between units in a clustering paradigm. Note these distance scores are not symmetric, as different radio characteristics and terrain effects will create different $interference^{c}_{rs}$ and $max\_interference^{c}_{s}$ values. To overcome this challenge, we redefine the distance between two units as the sum of the original distances in each direction. Specifically, the distance between units u and v is defined as:

$$distance_{uv} \equiv total\_normalized\_interference_{su} + total\_normalized\_interference_{rv} \tag{5.7}$$

where $s \in v$ is $most\_sensitive\_radio_{uv}$ and $r \in u$ is $most\_sensitive\_radio_{vu}$. We pre-calculate $distance_{uv}$ values for all $(u, v) \in A$ in $\mathcal{O}\left(a|U|a\left(|U|-1\right)\right) = \mathcal{O}\left(a^2|U|^2\right)$ time, where the constant a is the maximum number of radios in a unit (we assume 30).

Note that even with these simplifying assumptions, this is not a *metric space* because the *triangle inequality* does not necessarily hold. This defining property of a metric space

states that (in our case) the combined distance from a unit a to unit b and unit b to unit c must be greater than or equal to the distance between unit a to unit c, i.e.,

$$distance_{ac} \leq distance_{ab} + distance_{bc}. \tag{5.8}$$

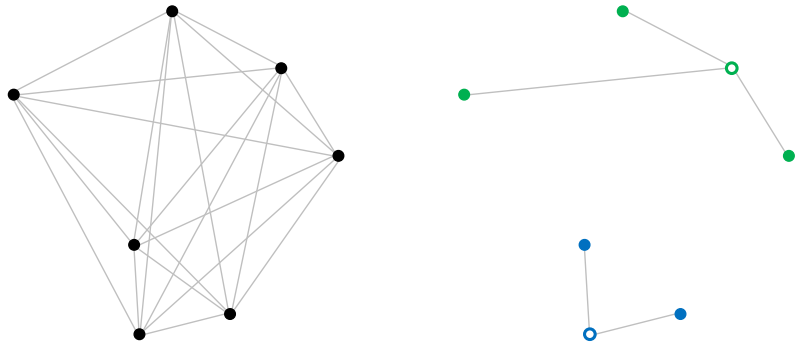
This property does not necessarily hold in the current paradigm because of the effects of rough terrain (e.g., hills and valleys) and different radio specifications on signal propagation. This prevents us from using clustering algorithms that are dependent on a metric space where the triangle inequality holds, such as k-means.

To overcome this computational challenge, we use a k-medoids clustering algorithm, which generates k clusters based on the *dissimilarity* (in our case, $distance_{uv}$) between the k selected *medoids* and the surrounding observations (in our case, units) (Hastie et al. 2001). This method avoids the computation of a *Euclidean centroid* (such as in k-means clustering), which is not defined in this non-metric space.

Ignoring for a moment the non-Euclidean nature of this method, we provide a simple graphic example of our clustering process in Figure 5.1. In Figure 5.1a, the arcs between each unit represent $distance_{uv}$ values, where shorter arc lengths represent smaller $distance_{uv}$. We wish to divide this space into $k = 2$ clusters, i.e., channels. In Figure 5.1b, two medoids are selected (indicated by open circles), and the units are divided into two clusters (indicated by blue and green) to minimize the total distance between each medoid and its assigned units.

We specify our k-medoids clustering formulation building on the notation we use in the MO-CAP, where X_u^c is a binary variable indicating whether unit u is assigned channel c. Let Y_u^c be a binary variable indicating whether unit u is selected as a medoid and is assigned channel c, i.e.,

$$Y_u^c = \begin{cases} 1, & \text{if unit } u \text{ uses channel } c \\ 0, & \text{otherwise} \end{cases} \qquad \forall u \in U, c \in C. \tag{5.9}$$



(*a*) (*b*)

Figure 5.1: Simple example of the clustering algorithm. In (a), the distance between each unit is calculated based on radio sensitivity and interference, as per (5.7). In (b), two clusters are created (indicated by green and blue) by choice of medoids (indicated by open circles) that minimize the total distance from each medoid to each assigned unit.

The total number of medoids is $|C|$, which is constrained via:

$$\sum_{u \in U} Y_u^c = |C|. \tag{5.10}$$

We wish to select medoids and assign units to medoids (via channel assignments) such that the total distances from the selected medoids to all units in that cluster (i.e., on the same channel) are minimized, i.e.,

$$\min \sum_{c \in C} \sum_{(u,v) \in A} distance_{uv} Y_u^c X_v^c. \tag{5.11}$$

Note that minimizing this distance is not equivalent to minimizing the total distances among all pairs within a cluster (i.e., total received interference). We make this simplification in order to avoid the computational difficulties of the latter problem. To linearize this

objective, we use the binary variable Z_{uv}^c where:

$$Z_{uv}^c = \begin{cases} 1, & \text{if } Y_u^c = X_v^c = 1 \\ 0, & \text{otherwise} \end{cases} \qquad \forall\,(u,v) \in A, c \in C \tag{5.12}$$

which is enforced via:

$$Z_{uv}^c \geq Y_u^c + X_v^c - 1 \qquad \forall\,(u,v) \in A, c \in C \tag{5.13}$$

$$Z_{uv}^c \leq Y_u^c \qquad \forall\,(u,v) \in A, c \in C \tag{5.14}$$

$$Z_{uv}^c \leq X_v^c \qquad \forall\,(u,v) \in A, c \in C. \tag{5.15}$$

We thus obtain our objective function:

$$\min \sum_{c \in C} \sum_{(u,v) \in A} distance_{uv} Z_{uv}^c. \tag{5.16}$$

Our MI-CAP Clustering Formulation is summarized as follows:

MI-CAP Clustering Formulation

Index and Set Use

$u \in U$	unit (*alias* v)
$(u, v) \in A$	set of all arcs (u, v)
$c \in C$	channel

Input Data

$distance_{uv}$	total distance (dissimilarity) between u and v [none]

Decision Variables

X_u^c	binary variable indicating whether u is using c
Y_u^c	binary variable indicating whether u is the medoid for c
Z_{uv}^c	binary variable indicating whether u is the medoid for c and unit v is using c

Formulation

$$\min_{X,Y,Z} \sum_{c \in C} \sum_{(u,v) \in A} distance_{uv} Z_{uv}^c \tag{L0}$$

$$\sum_{c \in C} X_u^c = 1 \qquad \forall u \in U \tag{L1}$$

$$\sum_{c \in C} \sum_{u \in U} Y_u^c = |C| \tag{L2}$$

$$Z_{uv}^c \geq Y_u^c + X_v^c - 1 \qquad \forall\, (u, v) \in A, c \in C \tag{L3}$$

$$Z_{uv}^c \leq Y_u^c \qquad \forall\, (u, v) \in A, c \in C \tag{L4}$$

$$Z_{uv}^c \leq X_v^c \qquad \forall\, (u, v) \in A, c \in C \tag{L5}$$

$$X_u^c \in \{0, 1\} \qquad \forall u \in U, c \in C \tag{L6}$$

$$Y_u^c \in \{0, 1\} \qquad \forall u \in U, c \in C \tag{L7}$$

$$Z_{uv}^c \in \{0, 1\} \qquad \forall\, (u, v) \in A, c \in C \tag{L8}$$

The MI-CAP Clustering Formulation is a pure 0-1 integer program. The objective function (L0) minimizes the sum of distances (or dissimilarity) among all units assigned the same channel, i.e., in the same cluster. Constraints (L1) require the assignment of one channel to each unit, and constraint (L2) requires the assignment of $|C|$ medoids among the units $u \in U$. Constraints (L3)-(L5) enforce the definition of Z_{uv}^c.

5.2.1 Relationships to Other Problems

If $distance_{uv}$ is calculated between all $(u, v) \in A$ (thus forming a complete graph), then selecting the $|C|$ clusters that minimize these distances is equivalent to selecting cliques among this complete graph that minimize total clique distances. Our clustering formulation is a simplification in that we consider only the distance from each unit to its assigned medoid, not the clique distance.

5.2.2 Computational Challenges of the MI-CAP Clustering Formulation

The MI-CAP Clustering Formulation suffers from the same computational problem of the MI-CAP Full Standard Formulation: the presence of very small values in the objective function which prevent us from using CPLEX or other exact optimization solvers. Instead, we use a heuristic approach.

5.2.3 MI-CAP Clustering Formulation Solution Method

We use Python to implement a k-medoids clustering method similar to that of Park and Jun (2009), where k is the number of available channels $|C|$, and m is a unit $u \in U$ assigned as a medoid for a cluster of units $g \in G$. The following pseudo-code describes our algorithm for solving the MI-CAP using k-medoid clustering:

Algorithm *MI-CAP Clustering*

Input: $max\_interference_s^c$ and $interference_{rs}^c$ values; number of available channels $|C|$; $max\_iterations$

Output: cluster medoid and cluster assignments

begin
  Calculate $distance_{uv}$ for all units $(u, v) \in A$
  $iteration \leftarrow 0$
  $bestMedoids \leftarrow \{\ \}$
  $bestClusters \leftarrow \{\ \}$
  $bestDistances \leftarrow \{\ \}$
  while $iterations < max\_iterations$ **do**
    Randomly select medoids m
    Assign each unit $u \in U$ to medoid m that minimizes $distance_{um}$ to create clusters $g \in G$
    $oldMedoids \leftarrow$ current medoid assignments m, for all clusters $g \in G$
    $currentMedoids \leftarrow \{\ \}$
    while $currentMedoids \neq oldMedoids$ **do**
      $oldMedoids \leftarrow currentMedoids$
      for $g \in G$
        Find unit u in g that minimizes within-cluster $distance_{um}$
        Set this unit as new medoid m for cluster g
      next;
      for each medoid m
        Assign each unit $u \in U$ to medoid m that minimizes $distance_{um}$
      next;
      $currentMedoids \leftarrow$ current medoid assignments m, for clusters $g \in G$
      $currentClusters \leftarrow$ current cluster assignments
      $currentDistances \leftarrow$ current total distances $\sum_{c \in C} \sum_{(u,v) \in A} distance_{uv} Z_{uv}^c$
    end;
    if $currentDistances < bestDistances$
      $bestMedoids \leftarrow currentMedoids$
      $bestClusters \leftarrow currentClusters$
      $bestDistances \leftarrow currentDistances$
    endif;
    $iteration \leftarrow iteration + 1$
  end;
end;

Algorithm *MI-CAP Clustering* first calculates the distance (i.e., dissimilarity) values (5.7) between all units. During each iteration, the algorithm randomly selects initial medoids, and assigns each unit to the nearest (least dissimilar) medoid. With these randomly-selected medoids as a starting point, the algorithm will next iteratively find the unit in each cluster that minimizes total distance within each cluster and assign that unit as the new medoid. The algorithm will then assign each unit to the closest medoid among these new medoids. This process repeats until a local optimum is reached, i.e., the newly-selected medoids are the same as the previous medoids. Upon discovering this local optimum, the algorithm will then randomly select new initial medoids and repeat the entire process, for a given number iterations. Newly-selected medoids are saved as the incumbents *bestMedoids* if they provide the best solution yet discovered. We provide partial Python code for solving the MI-CAP using our clustering method in Appendix D.

5.2.4 MI-CAP Clustering Formulation Results

We run our clustering algorithm for 200 iterations for each case, which runs in about 12 minutes. Our detailed results our tabulated in Appendix A. Though this algorithm does not explicitly value reducing the number of pairwise constraint violations, we provide this information in Table A.2 for the sake of comparison with our other MI-CAP solution methods (Sections 5.3 and 5.4). Table A.3 displays the number of radios receiving excessive interference, and Table A.4 displays the total excessive interference (i.e., the objective function of the MI-CAP Full Standard Formulation). In Table 5.1, we provide the percentage of network availability (i.e., the percent of radios that are able to communicate with their network control node), for each time step and for varying levels of channel availability.

While the clustering algorithm runs relatively quickly and does not require a commercial solver, it provides only a heuristic solution and is prone to falling into local optima. As we will show in the next section, it also provides poorer performance than our other methods in most cases. Further, this method does not provide a lower bound.

Table 5.1: Percentage network availability (for radios) using the MI-CAP clustering formulation, with varying numbers of available channels and 200 iterations of the algorithm.

	Network Availability (radios), by Channel Availability					
Timestep	**100%**	**90%**	**80%**	**70%**	**60%**	**50%**
1	57.8%	57.0%	50.9%	49.9%	42.8%	37.8%
2	50.7%	43.6%	38.7%	44.6%	37.3%	38.5%
3	51.5%	48.2%	48.1%	45.7%	45.3%	36.9%
4	48.3%	47.9%	51.4%	48.2%	40.2%	41.0%
5	52.0%	41.3%	44.4%	43.1%	42.1%	40.8%
6	61.3%	53.7%	46.2%	48.4%	42.9%	41.5%
7	50.1%	46.7%	47.7%	47.7%	39.8%	34.1%
8	47.4%	46.3%	40.7%	38.0%	37.8%	38.8%
9	54.6%	50.0%	47.7%	42.5%	45.2%	39.6%
10	61.7%	55.9%	54.2%	51.4%	45.7%	41.0%
11	52.6%	43.9%	43.5%	44.5%	42.9%	38.6%
12	63.6%	60.4%	54.7%	52.5%	46.2%	43.1%
13	52.4%	56.2%	48.0%	50.5%	47.1%	46.4%
14	56.2%	52.4%	51.8%	49.9%	46.2%	43.7%
15	54.0%	52.8%	47.5%	42.0%	46.2%	37.3%
16	64.6%	54.4%	58.1%	52.0%	55.9%	47.2%
Average	**54.8%**	**50.8%**	**48.4%**	**47.5%**	**43.7%**	**40.3%**

5.3 MI-CAP Restricted Integer Programming Formulation

We next develop a restricted version of the MI-CAP Full Standard Formulation to which we can apply integer optimization techniques. In this relaxation, we follow Subramanian et al. (2008) and try to minimize the number of pairwise interference constraint violations. Pairwise constraints represent the most critical interference and are more likely to impact the solution than higher-order constraints. Thus by focusing solely on these pairs, we capture the largest portion of the total interference.

Let the binary variable X_u^c indicate whether unit u is assigned to channel c, and let the binary variable Z_{uv}^c indicate whether units u and v are both assigned to channel c, for all arcs $(u, v) \in A$. Let $penalty_{uv}$ indicate the penalty for assigning u and v to the same channel; if non-zero, this represents the violation of a pairwise interference constraint. Our

objective function is thus:

$$\min \sum_{c \in C} \sum_{(u,v) \in A} penalty_{uv} Z^c_{uv}. \tag{5.17}$$

We initially use a unit penalty for all pairwise constraints. This is equivalent to the *maximum constraint satisfaction problem* (*MaxCSP*) (Freuder and Wallace 1992), as minimizing the number of violated constraints is equivalent to maximizing the number of satisfied constraints. We also consider the use of penalties where $penalty_{uv}$ is equal to the number of radios in u and v that will receive excessive interference if the two units are assigned the same channel; we refer to this as *weighted penalties*.

The MI-CAP Restricted Integer Programming Formulation is summarized as follows:

MI-CAP Restricted Integer Programming Formulation

Index and Set Use

$u \in U$	unit (*alias* v)
$(u, v) \in A$	arc representing constraint violation between u and v
$c \in C$	channel

Input Data

$penalty_{uv}$	penalty of assigning u and v to the same channel [number of radios]

Decision Variables

X^c_u	binary variable indicating whether u is using c
Z^c_{uv}	binary variable indicating whether u and v are both using c

Formulation

$$\min_{X,Z} \quad \sum_{c \in C} \sum_{(u,v) \in A} penalty_{uv} Z^c_{uv} \tag{P0}$$

$$\sum_{c \in C} X^c_u = 1 \qquad \forall u \in U \tag{P1}$$

$$Z^c_{uv} \geq X^c_u + X^c_v - 1 \qquad \forall (u, v) \in A, c \in C \tag{P2}$$

$$Z^c_{uv} \leq X^c_u \qquad \forall (u, v) \in A, c \in C \tag{P3}$$

$$Z^c_{uv} \leq X^c_v \qquad \forall (u, v) \in A, c \in C \tag{P4}$$

$$X^c_u \in \{0, 1\} \qquad \forall u \in U, c \in C \tag{P5}$$

$$Z^c_{uv} \in \{0, 1\} \qquad \forall (u, v) \in A, c \in C \tag{P6}$$

The MI-CAP Restricted Integer Programming Formulation is a pure 0-1 integer program. The objective function (P0) minimizes the sum of penalties for violating (preprocessed) pairwise interference constraints. Constraints (P1) require the assignment of one channel to each unit. Constraints (P2)-(P4) enforce the definition of Z_{uv}^c.

5.3.1 Relationships to Other Problems

Subramanian et al. (2008) observe that the pairwise interference MI-CAP formulated as an IP is similar to the *max k-cut problem* (see, e.g., Frieze and Jerrum (1995)). This problem assigns each node within a graph into one of k separate partitions in order to maximize the number of edges whose endpoints are in different partitions. When this problem is applied to our interference graph, this is equivalent to minimizing the number of pairwise interference constraint violations, i.e., $\sum_{(u,v)\in A} Z_{uv}^c$. However, the max k-cut problem is also NP-hard, so this insight does not immediately result in an improvement in optimization efficiency.

5.3.2 Computational Challenges of the MI-CAP Restricted IP Formulation

We note that as long as each unit is assigned a channel (P1), any channel assignment solution will satisfy the constraints. We also observe that this formulation has a *weak LP relaxation*, i.e., there is potential for a very large gap between the LP solution and the integer-feasible solution. Specifically, when the integer constraints are relaxed, the LP optimal solution will comprise very small fractions for X_u^c and Z_{uv}^c that will cumulatively satisfy constraints (P1), but will generate only very small penalties in the objective function (P0). This typically results in an optimality gap between the best integer solution and the LP relaxation of nearly 100%.

5.3.3 Calculating a Lower Bound

In order to find a more useful lower bound for determining the goodness of solutions obtained to the MI-CAP IP formulation, we use the result of Montemanni et al. (2001), who establish a lower bound on the number of *monochromatic arcs* within a clique given k colors. In our problem, a monochromatic arc within the interference graph represents a pairwise interference constraint violation. We apply and develop their result (also used for this purpose by Subramanian et al. (2008)) to the cliques within our interference graph to establish a lower bound on the number of pairwise constraints violations and provide a tighter optimality gap. We consider both the case where all $penalty_{uv} = 1$, and weighted $penalty_{uv}$ values.

5.3.3.1 Unit Penalties

We first consider the case where all $penalty_{uv} = 1$, i.e., unit penalties. We describe our method by slightly modifying the notation of Montemanni et al. (2001). Let $S \subset U$ be a subset of units forming a clique S within the unweighted interference graph. Let $|S|$ be the number of units in the clique, and let $|C|$ be the number of available channels, which may be less than the number identified using MO-CAP (i.e., reduced channel availability). Define:

$$\alpha = \left\lfloor \frac{|S|}{|C|} \right\rfloor \tag{5.18}$$

and

$$\beta = |S| \bmod |C|. \tag{5.19}$$

Montemanni et al. (2001) derive and prove that a lower bound on the number of monochromatic arcs within S is:

$$\tau = \frac{\alpha\beta\,(\alpha+1) + (|C| - \beta)\,\alpha\,(\alpha-1)}{2}. \tag{5.20}$$

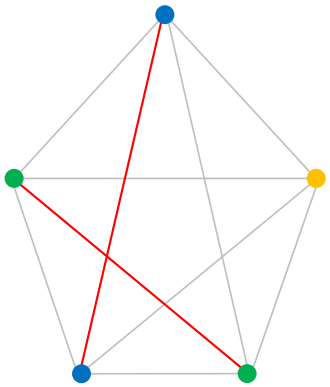


Figure 5.2: A simple example of the lower bound on the number of monochromatic arcs in a clique. With three available colors (i.e., green, blue, and orange), a clique comprising five nodes must have at least $\tau = 2$ monochromatic arcs (indicated by red).

Note that $|C| \geq |S| \implies \tau = 0$, i.e., the lower bound is zero when there are sufficient channels to uniquely assign a channel to each unit $u \in S$.

We provide an illustration of this lower bound in Figure 5.2. Consider the clique formed among the five units in this interference graph. If we wish to color the units in this clique (i.e., assign channels) with $|C| = 3$ colors (indicated by blue, green and orange), there must be at least $\tau = 2$ violations where an arc is monochromatic (indicated by red).

We calculate τ for disjoint cliques within the interference graph, beginning with the maximum clique M and then iteratively considering the maximum clique among those units that have not yet been considered. We define T as the sum of the τ for these disjoint cliques within the interference graph, and use T as a lower bound on our MI-CAP IP objective function, i.e.,

$$\sum_{c \in C} \sum_{(u,v) \in A} penalty_{uv} Z_{uv}^{c} \geq T \tag{5.21}$$

assuming all $penalty_{uv} = 1$. The bound

$$\sum_{c \in C} \sum_{(u,v) \in A} Z_{uv}^{c} \geq T \tag{5.22}$$

is valid for both the unit penalty and weighted penalty cases, and we use this bound when we are unable to calculate a tighter bound to the weighted MI-CAP.

We investigate adding (5.22) as a constraint within our Restricted IP Formulation, but find that it does not improve the efficiency of the search process, and in fact sometimes results in poorer solutions in the same amount of time (unlike the results observed with the much smaller datasets of Subramanian et al. (2008)). However, it does provide utility in providing non-zero lower bounds on the number of pairwise violations, and can be calculated *a priori*.

Further, this result can be used to calculate lower bounds on both the total amount of received excessive interference (i.e., $\sum_{s \in R} E_s$ from the MI-CAP Full Standard Formulation), and on the total number of radios that will receive excessive interference. To calculate each, we find the τ arcs among each clique S that, should they be violated, respectively result in the least total amount of excessive interference and the fewest number of radios receiving excessive interference, i.e., the best case possible. (Note the τ arcs selected by these two operations may be different.) A drawback to this method is that these bounds may be infeasible in the full problem because they do not consider the added cost incurred when two or more arcs are adjacent. However, both of these lower bounds can be calculated *a priori*.

The following pseudo-code describes our method of calculating these lower bounds for a particular time step and given number of available channels in the MI-CAP with unit penalties.

Algorithm *MI-CAP Lower Bounds* (*Unit Penalties*)

Input: pairwise interference constraints among all $(u, v) \in A$; number of available channels $|C|$

Output: T (lower bound on number of pairwise constraint violations); LB_E (lower bound on excessive interference); LB_R (lower bound on number of radios receiving excessive interference)

begin
  $unitList \leftarrow U$
  $T \leftarrow 0$
  $LB_E \leftarrow 0$
  $LB_R \leftarrow 0$
  while $|unitList| > 1$ **do**
    $S \leftarrow$ maximum clique formed among pairwise interference constraints in $unitList$
    if $|S| < |C|$
      exit while
    else
      $\alpha \leftarrow \left\lfloor \frac{|S|}{|C|} \right\rfloor$
      $\beta \leftarrow |S| \bmod |C|$
      $\tau \leftarrow \frac{\alpha\beta\,(\alpha+1) + (|C| - \beta)\,\alpha\,(\alpha-1)}{2}$
      $T \leftarrow T + \tau$
    endif;
    $eList \leftarrow \{\ \}$
    $rList \leftarrow \{\ \}$
    for $(u, v) \in S$
      $e \leftarrow$ excessive interference if (u, v) are assigned same channel
      $r \leftarrow$ number of radios receiving excessive interference if (u, v) are assigned same channel
      $eList \leftarrow eList \cup e$
      $rList \leftarrow rList \cup r$
    next;
    Sort $eList$ and $rList$ descending
    $LB_E \leftarrow LB_E +$ total excessive interference from top τ elements in $eList$
    $LB_R \leftarrow LB_R +$ total number of radios receiving excessive interference from top τ elements in $rList$
    $unitList \leftarrow unitList \backslash S$
  end;
end;

We provide these lower bounds on the unit-penalty method in Appendix A in Table A.5 for each time step and with various levels of channel availability (i.e., $|C|$), where 100% availability is the full MO-CAP solution and lesser percentages indicate a reduced number of available channels (available in Table A.1). We calculate and provide the lower bound on total amount of received excessive interference (in dBm) in Table A.6. By happenstance, the lower bounds on the total number of radios receiving excessive interference are equal in each case to the lower bound on the number of pairwise constraint violations (Table A.5), so we do not duplicate this table.

Note that these bounds are theoretical and do not indicate the presence of a feasible solution at the lower bound, as they consider a bound only on the cliques within the (pairwise) interference graph, and not the entire MI-CAP. We find that the lower bound on the number of pairwise constraint violations is useful in determining the goodness of our MI-CAP solutions, as one of our methods approaches and even occasionally achieves optimality. However, the lower bound on the amount of received excessive interference is of little use, as it is far below our observed solutions and provides relative optimality gaps of nearly 100% in all cases (in watts). This is not surprising, as this method considers only the excessive interference among the violated pairwise constraints within the examined cliques, and not interference among all units (pairwise and higher-order).

Note also these lower bounds apply even to those time steps which have non-zero optimality gaps in the MO-CAP, as these results depend on a *given* number of available channels, not on the true minimum required number of channels.

5.3.3.2 Weighted Penalties

We next consider the case of weighted penalties. In general, τ may not be a feasible lower bound for the costs of violations in a weighted clique. Suppose we select the τ lowest-cost arcs within a weighted clique by simply sorting and selecting them. Selecting an arc in this way monochromatically colors the associated vertices, incurring a penalty. We illustrate by continuing the example of Figure 5.2 in Figure 5.3. If we are choosing $\tau = 2$ arcs, perhaps

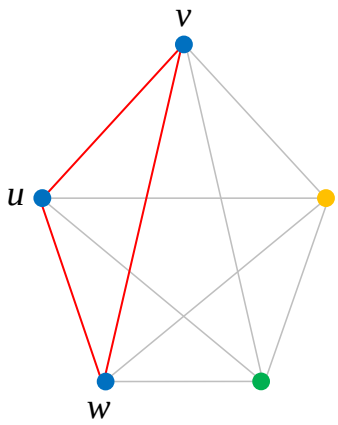


Figure 5.3: A simple example of how selecting monochromatic arcs (indicated by red) may inadvertently result in another monochromatic arc.

we select arcs (u, v) and (u, w). They share vertex u, and so now we also inadvertently incur the cost associated with arc (v, w), forming a triangle when we intend to select two disjoint arcs and thus incurring a larger cost.

Further, in a weighted clique it may be possible to select a less-expensive set of arcs with cardinality greater than τ. For example, suppose the triangle we form in the previous example is of lower cost than any set of two non-adjacent arcs.

In an attempt to establish a tighter lower bound for the weighted-penalty MI-CAP and building on the methodology of Montemanni et al. (2001), we transform the problem as follows. Let a node $i \in N$ represent an arc in the original clique S. Each node has an associated $penalty_i$ equal to the original $penalty_{uv}$ in the Restricted IP formulation. For each triangle formed in the original clique S (e.g., the triangle (u, v, w) in Figure 5.3), we create a hyper-arc $(u, v, w) \in H$. Let the binary decision variable Z_i indicate whether node i is selected, i.e., the associated pairwise constraint in the clique S is violated:

$$Z_i = \begin{cases} 1, & \text{if node } i \text{ is selected} \\ 0, & \text{otherwise} \end{cases} \qquad \forall i \in N. \tag{5.23}$$

We wish to minimize the total penalty, so our objective function is:

$$\min \sum_{i \in N} penalty_i Z_i. \tag{5.24}$$

From the result of Montemanni et al. (2001), we must select at least τ arcs from the clique S, which we enforce via the constraint:

$$\sum_{i \in N} Z_i \geq \tau. \tag{5.25}$$

To ensure that we appropriately penalize all monochromatic arcs (e.g., arc (v, w) in Figure 5.3), we include the constraints:

$$Z_i \geq Z_j + Z_k - 1 \qquad \forall\, (i, j, k) \in H \tag{5.26}$$

which force node i to be selected if both nodes j and k are selected, among all hyper-arcs $(i, j, k) \in H$.

To illustrate these constraints and the relationship between the original clique S and the current transformation, consider Figure 5.4, comprising units u, v, w, and x from the clique S. Each node in this transformation (indicated in blue font, e.g., node i) represents an arc in clique S (e.g., (u, v)). We enumerate all hyper-arcs H by identifying triangles in the clique and indicate them by node label, so in this example H comprises permutations on $\{(i, k, l), (i, j, m), (j, l, n), (k, m, n)\}$. Suppose $\tau = 2$, and nodes i and k are selected. In the original problem, this indicates that units u, v, and x are now monochromatically colored. The hyper-arc $(l, i, k) \in H$, so the constraint $Z_l \geq Z_i + Z_k - 1$ ensures that node l, though not intentionally selected, is now selected and generates a penalty representing the co-channel interference between units u and x.

Since the structure of the graph is a clique and we have enumerated all hyper-arcs, these constraints also induce the selection of any other arcs required to model monochromatic

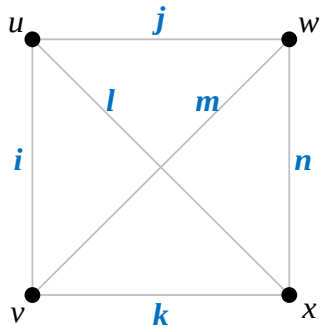


Figure 5.4: Simple illustration of the relationship between units in the original clique S (u, v, w, and x) and nodes (i, j, k, l, m, and n) in the transformation used to generate a lower bound to the weighed MI-CAP.

coloring of units. Suppose $\tau = 3$, and suppose nodes i, j, and k are selected, thus monochromatically coloring all four units u, v, w, and x. From (5.26), constraints $Z_l \geq Z_i + Z_k - 1$ and $Z_m \geq Z_i + Z_j - 1$ immediately ensure nodes l and m are selected. Now, both $Z_n \geq Z_k + Z_m - 1$ and $Z_n \geq Z_j + Z_l - 1$ ensure the selection of node n, yet the use of binary variables (e.g., Z_n) ensures that any node generates at most one penalty.

Our *minimum-cost weighted clique lower bound problem* is summarized as follows:

Minimum-Cost Weighted Clique Lower Bound Formulation

Index and Set Use

$i \in N$	node representing an arc in the original clique (*alias* j, k)
$(i, j, k) \in H$	hyper-arc comprising nodes i, j, and k, representing a triangle in the original clique

Input Data

$penalty_i$	penalty of selecting node i	[number of radios]
τ	minimum number of selected nodes	[number of nodes]

Decision Variables

Z_i	binary variable indicating whether node i is selected

Formulation

$$\min_{Z} \quad \sum_{i \in N} penalty_i Z_i \tag{B0}$$

$$\sum_{i \in N} Z_i \geq \tau \tag{B1}$$

$$Z_i \geq Z_j + Z_k - 1 \qquad \forall\, (i, k, k) \in H \tag{B2}$$

$$Z_i \in \{0, 1\} \qquad \forall i \in N \tag{B3}$$

The minimum-cost weighted clique lower bound problem is a pure 0-1 integer program. The objective function (B0) minimizes the sum of penalties associated with selecting nodes $i \in N$. Constraint (B1) ensures that at least τ nodes must be selected. Constraints (B2) ensure that the selection of any two adjacent nodes within a hyper-arc $(i, j, k) \in H$ result in the selection of the third node in the hyper-arc.

We use CPLEX to solve the problem for each time step and with varying τ, and present the results in Table A.7. Empirically, we find that this method provides far better performance than simply solving the original Restricted IP formulation or the CP formulation (Section 5.3.6) on the maximum clique. In each time step and case, we are able to solve this problem to optimality in less than a second, whereas we are unable to solve the Restricted IP formulation or CP formulation (on the maximum clique) to optimality with 50% channel availability, even after extremely long (24 hour) runtimes. This problem does not consider channel or color assignment, and is thus much simpler and results in smaller

problem instances (in terms of variables and constraints) than the original Restricted IP formulation.

5.3.4 MI-CAP Restricted IP Formulation Solution Method

Having established lower bounds to the MI-CAP Restricted IP Formulation of both unit and weighted penalties, we continue describing our solution method. We determine the number of available channels $|C|$ at each given time step using MO-CAP, or possibly based on an allocation provided to our spectrum manager from a senior decision-maker or a policy. We pre-process the pairwise interference constraints using Python, and then solve the problem at each time step using CPLEX. We run this analysis with both unit penalties ($penalty_{uv} = 1$) and weighted penalties.

5.3.5 MI-CAP Restricted IP Formulation Results (Unit Penalties)

For each time step, we consider various levels of channel availability, where 100% availability is the full MO-CAP solution, and lesser percentages indicate a reduced number of available channels, thereby inducing excessive co-channel interference (see Table A.1 for the number of channels available at each time step). We run CPLEX for 500 seconds for each case. The number of pairwise violations (i.e., the objective function to this problem if $penalty_{uv} = 1, \forall\,(u,v) \in A$) are displayed in Table 5.2. In Appendix A, we provide the number of radios receiving excessive interference (Table A.8), and the total excessive interference (i.e., the objective function of the MI-CAP Full Standard Formulation) (Table A.9).

Even with the full (i.e., original MO-CAP solution) number of available channels, the solver is unable to find a solution that eliminates pairwise violations. In general and as expected, these violations (and the total excessive interference) increase as the number of available channels decreases. This relationship is not always monotonic increasing (e.g., time step 1 from 90% to 80%), as the solver may make more or less progress depending on the number of available channels, but holds true on average.

Table 5.2: Total number of pairwise violations using the MI-CAP Restricted IP formulation, with varying numbers of available channels, 500 second CPLEX runtimes, and unit penalties.

	Pairwise Violations, by Channel Availability					
Time Step	**100%**	**90%**	**80%**	**70%**	**60%**	**50%**
1	55	72	95	109	133	329
2	73	85	98	218	227	302
3	86	83	111	190	204	280
4	86	101	118	246	241	283
5	84	96	117	203	251	125
6	72	96	110	238	259	254
7	67	84	105	177	341	243
8	79	106	197	239	225	380
9	77	85	145	209	200	206
10	76	87	106	221	199	184
11	75	89	107	202	264	254
12	63	77	85	191	297	287
13	83	102	166	104	247	134
14	79	103	158	266	152	305
15	67	81	97	107	188	195
16	71	92	107	182	219	282
17	69	84	93	209	385	260
18	71	86	171	213	220	268
19	75	96	185	245	208	111
20	71	79	99	231	183	171
Average	**74.0**	**89.2**	**123.5**	**200.0**	**232.2**	**242.7**

We note that the logarithmic basis of dBm may be misleading in terms of interpreting the differences in Table A.9. The average amount of excessive interference received with 50% of the required channels (-19.74 dBm) is in fact roughly 21 times more powerful than the interference received with 100% channel availability.

We also note that the number of pairwise violations and the excessive interference is considerably greater in the first time step than in any other time step. As we note in Section 4.2.2, this occurs because within the MEF amphibious assault scenario, the units have just reached the beach at the first time step and are relatively close to one another. Thereafter, the units spread apart and are better able to reduce co-channel interference (see Figure 3.3).

In Table 5.3, we provide an estimate of the operational impact using network availability, i.e., the number of radios that are able to communicate with their network control node.

Table 5.3: Network availability (for radios) using the MI-CAP Restricted IP formulation, with varying numbers of available channels, 500 second runtimes, and unit penalties.

	Network Availability (radios), by Channel Availability					
Timestep	**100%**	**90%**	**80%**	**70%**	**60%**	**50%**
1	63.0%	48.2%	46.3%	43.8%	40.5%	42.2%
2	58.5%	54.6%	45.9%	44.0%	43.7%	35.0%
3	45.7%	46.8%	42.8%	50.0%	49.3%	40.2%
4	48.6%	72.7%	51.1%	49.8%	38.5%	44.5%
5	50.3%	50.4%	48.1%	49.4%	43.2%	48.0%
6	52.9%	51.9%	43.8%	51.2%	46.5%	42.1%
7	70.5%	45.4%	43.3%	52.3%	44.4%	40.8%
8	65.6%	54.5%	50.8%	42.6%	48.6%	45.1%
9	71.1%	68.8%	44.7%	68.3%	45.2%	45.4%
10	58.7%	60.5%	53.5%	58.6%	53.5%	52.6%
11	52.7%	52.8%	52.6%	53.6%	54.2%	35.1%
12	59.5%	52.8%	51.0%	59.8%	50.2%	49.7%
13	67.6%	51.0%	57.0%	58.8%	52.7%	47.3%
14	63.0%	59.7%	54.6%	55.0%	56.4%	42.3%
15	58.8%	56.9%	52.6%	45.2%	42.0%	53.8%
16	59.1%	56.7%	49.5%	50.9%	50.7%	52.5%
17	56.8%	57.3%	52.0%	47.4%	49.5%	43.8%
18	75.3%	62.6%	49.8%	59.6%	50.1%	45.1%
19	56.1%	54.8%	57.9%	58.2%	45.3%	49.1%
20	64.7%	54.2%	57.7%	56.7%	47.1%	50.0%
Average	**59.9%**	**55.6%**	**50.3%**	**52.8%**	**47.6%**	**45.2%**

5.3.6 MI-CAP Restricted IP Formulation Results (Weighted Penalties)

Next, in an attempt to reduce the number of radios receiving excessive interference, we re-run the analysis but set each $penalty_{uv}$ equal to the number of radios in u and v that will receive excessive interference if the two units are assigned the same channel. Initially, we run each case for 500 seconds. The results are presented in Appendix A in Tables A.10, A.11, and A.12. We observe that in general, the solutions are actually less desirable than those obtained using unit penalties: the number of pairwise constraint violations and the

total number of radios receiving excessive interference is greater. This occurs because the weighted (non-unit) penalties remove the symmetry of the problem, making the problem much more difficult for the solver.

Because of this added computational burden and in order to see a reduction in received interference using weighted penalties, we must run the solver for a longer period of time. We run the solver for 6000 seconds for each time step and level of channel availability, and present the number of violations and availability results in Tables 5.4 and 5.5. The number radios receiving excessive interference and the amount excessive interference are presented in Tables A.13 and A.14.

Table 5.4: Total number of pairwise violations using the MI-CAP Restricted IP formulation, with varying numbers of available channels, 6000 second CPLEX runtimes, and weighted penalties.

	Pairwise Violations, by Channel Availability					
Time Step	**100%**	**90%**	**80%**	**70%**	**60%**	**50%**
1	14	23	25	41	53	77
2	14	13	34	47	53	87
3	15	21	33	59	61	121
4	16	25	32	47	82	95
5	48	30	36	48	98	110
6	17	25	33	45	62	97
7	13	18	32	50	65	93
8	7	21	40	54	66	106
9	3	14	21	40	52	83
10	4	12	23	40	55	79
11	1	6	15	29	49	72
12	1	7	15	26	43	54
13	6	16	21	38	51	74
14	0	13	22	33	49	72
15	8	18	19	32	50	61
16	9	16	28	41	68	71
17	8	18	32	40	53	77
18	2	13	12	35	53	69
19	6	13	19	41	54	74
20	10	17	21	36	49	81
Average	**74.0**	**89.2**	**123.5**	**200.0**	**232.2**	**242.7**

Table 5.5: Network availability (for radios) using the MI-CAP Restricted IP formulation, with varying numbers of available channels, 6000 second runtimes, and weighted penalties.

	Network Availability (radios), by Channel Availability					
Timestep	**100%**	**90%**	**80%**	**70%**	**60%**	**50%**
1	85.4%	77.2%	76.2%	65.4%	58.7%	51.4%
2	86.6%	85.1%	72.2%	65.6%	59.8%	51.1%
3	86.9%	72.0%	70.4%	59.7%	60.9%	40.5%
4	84.8%	82.0%	68.9%	67.4%	55.8%	50.7%
5	78.0%	67.7%	70.6%	65.3%	53.6%	50.0%
6	83.8%	78.7%	73.1%	73.0%	73.0%	56.8%
7	84.6%	80.5%	73.0%	59.7%	51.6%	48.0%
8	94.3%	82.2%	77.4%	75.4%	60.9%	51.3%
9	96.2%	83.1%	82.3%	73.2%	65.7%	63.3%
10	92.9%	84.5%	81.7%	74.7%	65.9%	56.4%
11	95.1%	90.9%	85.7%	75.4%	69.8%	59.4%
12	93.6%	87.0%	81.1%	76.2%	69.6%	60.1%
13	94.1%	88.0%	80.8%	80.8%	64.0%	50.7%
14	94.4%	88.4%	80.6%	75.4%	66.3%	61.6%
15	91.4%	84.9%	81.3%	66.9%	61.6%	63.4%
16	92.0%	84.9%	74.1%	70.8%	59.5%	59.0%
17	89.1%	86.2%	69.4%	67.2%	59.7%	58.5%
18	94.9%	87.8%	83.3%	75.8%	69.3%	53.5%
19	95.0%	87.3%	78.2%	76.0%	62.6%	57.1%
20	90.4%	82.0%	74.8%	73.3%	68.2%	59.9%
Average	**59.9%**	**55.6%**	**50.3%**	**52.8%**	**47.6%**	**45.2%**

We compare the performance of the unit penalties and weighted penalties method with 70% channel availability in Figures 5.5 and 5.6. In Figure 5.5, we see that the weighted penalty method provides a significant reduction in the number of radios receiving excessive interference, but roughly similar levels of received excessive interference. In Figure 5.6, we see that the weighted penalty method obtains solutions with greater numbers of pairwise constraint violations, but in conjunction with Figure 5.5, we conclude that these violations are in general of less cost.

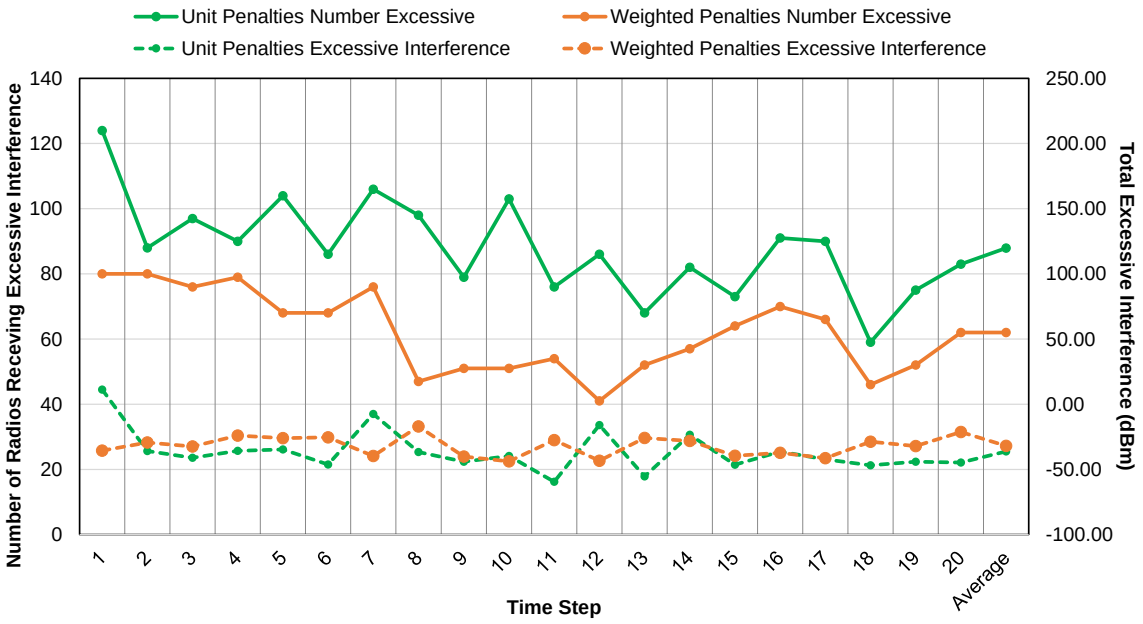


Figure 5.5: Comparison of number of radios receiving excessive interference (out of 1887), and the total received interference, between the unit penalty and weighted penalty IP methods (6000 sec run times), with 70% channel availability.

5.4 MI-CAP Constraint Programming Formulation

As with the MO-CAP, in the MI-CAP we are not particularly concerned with the actual channel number assigned to a group of units, so long as each group receives an assignment. The MI-CAP Restricted Integer Programming Formulation does not leverage this property, and may waste computational effort by explicitly considering channel assignment as an index. The MI-CAP is natural candidate for constraint programming, where channel assignment is represented as the value of a variable, instead of as an index on a variable.

We reformulate the MI-CAP as a CP problem. If the number of available channels $|C|$ is less than the optimal MO-CAP solution, then the problem is *over-constrained*, i.e., one or more pairwise constraints must be violated. We associate a penalty with each constraint and then use CP to attempt to minimize the penalties associated with these violations. In

Figure 5.6: Comparison of number of the total number of pairwise constraint violations, between the unit penalty and weighted penalty IP methods (6000 sec run times), with 70% channel availability.

the constraint programming literature, this approach is referred to as the *optimal soft arc consistency problem* (Cooper et al. 2007, Rossi et al. 2006).

Similar to the MO-CAP CP formulation, let the variable $W_u \in C$ indicate the channel assigned to unit $u \in U$, where the domain of each variable is equal to the number of available channels $|C|$. We model all pairwise constraints in the problem, indicating that two given units u and v are not allowed to be assigned the same channel, for all pairs $(u, v) \in A$. We use a CP logical construct to generate a variable $P_i \in \mathbb{R}$ for $i = 1, 2, \ldots, |A|$, if any pairwise constraint is violated. This is specified logically as:

$$(W_u = W_v) \implies P_i = penalty_i \qquad \forall\,(u, v) \in A. \tag{5.27}$$

That is, if $W_u = W_v$, P_i equals $penalty_i$, where $penalty_i$ is a scalar indicating the penalty incurred for the violation. Note this implies a preprocessed mapping $(u, v) \mapsto i, \forall\,(u, v) \in A$. As with the IP formulation, we consider both unit penalties and weighted penalties.

We wish to minimize the total penalties, so our objective function is:

$$\min \sum_{i=1}^{|A|} P_i. \tag{5.28}$$

Our MI-CAP constraint programming formulation is summarized as follows:

MI-CAP Constraint Programming Formulation

Index and Set Use

$u \in U$	unit (*alias* v)
$c \in C$	channel
$(u, v) \in A$	arc indicating u and v cannot occupy the same channel
i	penalty number, for $i = 1, 2, \ldots, \lvert A\rvert$

Input Data

$penalty_i$	possible penalty incurred if $W_u = W_v$	[number of radios]

Decision Variables

W_u	integer variable indicating the channel assignment of unit u
P_i	continuous variable equal to $penalty_i$ if $W_u = W_v$

Formulation

$$\min_{P,W} \quad \sum_{i=1}^{|A|} P_i \tag{N0}$$

$$(W_u = W_v) \implies P_i = penalty_i \qquad \forall\,(u, v) \in A, i = 1, 2, \ldots, |A| \tag{N1}$$

$$W_u \in C \qquad \forall u \in U \tag{N2}$$

$$P_i \in \mathbb{R} \qquad \forall i = 1, 2, \ldots, |A| \tag{N3}$$

The MO-CAP CP formulation attempts to find a feasible assignment of integer values for the variables $W_c \in C$ that minimizes the total penalties $\sum_{i=1}^{|A|} P_i$ (N0). Constraints (N1) are the logical statements that force P_i to $penalty_i$ if $W_u = W_v$. Note this is a relaxation of the MI-CAP Full Standard Formulation in that only pairwise constraints are considered.

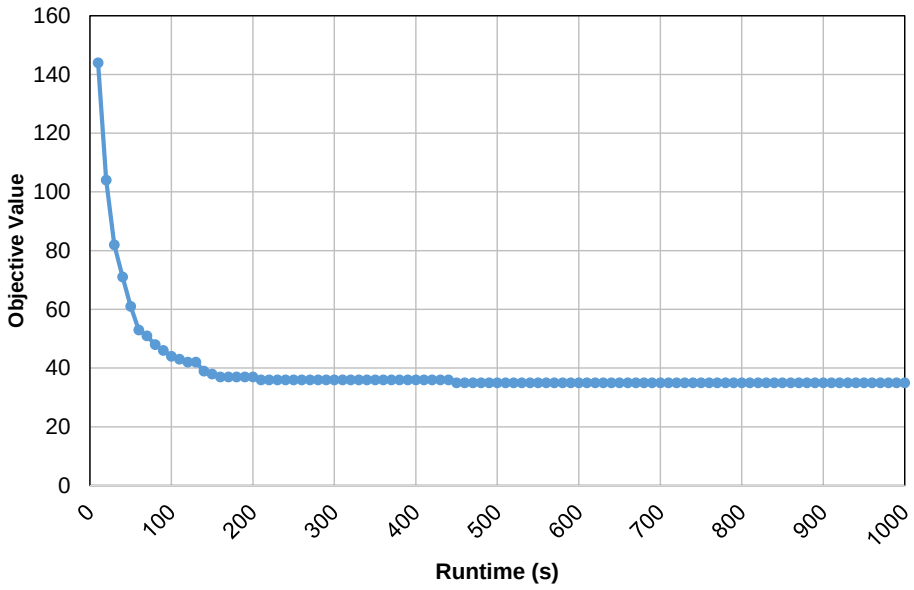


Figure 5.7: MI-CAP CP objective values obtained with varying runtimes for the first time step of the MEF scenario, from 10 to 1000 seconds, in ten second increments. No improvement is observed beyond 440 seconds.

5.4.1 MI-CAP CP Solution Method

As in our previous methods, we use Python to determine the pairwise constraints. We then use OPL to formulate the problem, and solve using IBM CPLEX CP Optimizer. We provide partial Python and OPL code in Appendix E. For both the unit and weighted penalties methods, we use CP runtimes of 500 seconds, as empirically we find no benefit of longer runtimes. This is demonstrated for the first time step in Figure 5.7, where we show the objective value obtained (for the unit penalty method) with CP runtimes varying from 10 to 1000 seconds, in ten second increments. No improvement to the objective value is made beyond runtimes of 440 seconds.

5.4.2 MI-CAP CP Formulation Results (Unit Penalties)

We again consider various levels of channel availability, where 100% availability is the full MO-CAP solution, and lesser percentages indicate a reduced number of available channels, thereby inducing excessive co-channel interference. We run CP Optimizer for 500 seconds for each case with unit penalties. The number of pairwise violations are displayed in Table 5.6 (where bold values indicate optimal solutions). In Table 5.7, we provide the network availability (i.e., the number of radios able to communicate with their respective network control radio). In Appendix A, we provide the total number of radios receiving excessive interference (Table A.15), and the total excessive interference (i.e., the objective function of the MI-CAP Full Standard Formulation) (Table A.16).

Both Table 5.6 and 5.7 demonstrate the improved performance of this method over the Restricted IP formulation (i.e., Tables 5.2 and 5.3); we compare these methods explicitly in Section 5.5.1.

5.4.3 MI-CAP CP Formulation Results (Weighted Penalties)

We next conduct the same analysis but with weighted penalties. Unlike the IP method, we are able to observe the difference between the unit penalty and weighted penalty variants without greatly increasing the solver runtime. We present the number of pairwise violations in Table 5.8 and network availability in Table 5.9. In Appendix A we provide the numbers of radios receiving excessive interference (Table A.17) and the total received excessive interference (Table A.18).

We compare the performance of the CP unit and weighted penalties methods in Figures 5.8 and 5.9, focusing on the cases with 70% channel availability at each time step. The weighted penalties method obtains solutions with less total excessive interference and fewer radios receiving excessive interference.

Table 5.6: Total number of pairwise violations using the MI-CAP CP formulation, with varying numbers of available channels, 500 second CP runtimes, and unit penalties. Bold values indicate optimal solutions.

	Number of Pairwise Violations, by Channel Availability					
Time Step	**100%**	**90%**	**80%**	**70%**	**60%**	**50%**
1	**0**	**5**	**10**	**14**	26	38
2	**0**	**4**	**8**	**12**	22	41
3	**0**	**3**	7	20	31	54
4	**0**	**3**	7	17	27	43
5	**0**	**4**	8	14	33	53
6	**0**	4	8	14	30	40
7	**0**	**4**	**8**	15	26	46
8	**0**	**3**	10	21	35	58
9	**0**	**4**	**7**	14	23	41
10	**0**	**4**	**7**	15	24	41
11	**0**	**4**	**7**	**10**	22	42
12	**0**	**3**	**7**	**10**	17	31
13	**0**	**3**	**6**	11	22	37
14	**0**	4	7	11	21	38
15	**0**	**4**	**8**	**12**	23	33
16	**0**	4	8	11	22	33
17	**0**	**3**	**7**	12	21	36
18	**0**	**4**	**7**	13	22	39
19	**0**	**3**	**6**	**9**	18	36
20	**0**	**4**	**8**	13	23	38
Average	**0**	**3.7**	**7.6**	**13.4**	**24.4**	**40.9**

5.5 Comparison of MI-CAP Solution Methods

To provide a qualitative sense of the results of the MI-CAP, we generate Figure 5.10 using Gephi (Gephi Consortium 2016, Bastian et al. 2009) and the CP MI-CAP solution for the first time step of the MEF scenario with 70% channel availability (32 channels) and unit penalties. The color of each arc and node depicts channel assignment; the size of each node is relative to the total amount of excessive interference received by that unit (the smallest size indicates no excessive interference); and the width of each arc is relative to the total interference between units on the same channel. One may expect units located in relatively dense clusters to experience the greatest co-channel interference, but Figure 5.10

Table 5.7: Network availability (for radios) using the MI-CAP CP formulation, with varying numbers of available channels, unit penalties, and 500 second runtimes.

	Network Availability (radios), by Channel Availability					
Timestep	**100%**	**90%**	**80%**	**70%**	**60%**	**50%**
1	93.8%	88.6%	76.2%	74.2%	60.4%	54.1%
2	97.6%	92.1%	84.6%	69.3%	70.7%	56.1%
3	96.9%	87.1%	83.0%	72.1%	63.8%	51.0%
4	99.5%	94.6%	86.9%	79.9%	71.4%	66.3%
5	96.5%	90.7%	87.4%	74.8%	64.8%	57.2%
6	96.1%	91.5%	87.9%	77.4%	68.3%	62.3%
7	97.3%	93.4%	84.5%	72.7%	70.6%	55.7%
8	98.0%	93.5%	85.6%	71.8%	63.4%	51.9%
9	95.7%	89.9%	85.3%	77.7%	71.8%	61.4%
10	96.0%	92.7%	88.3%	76.5%	70.2%	64.8%
11	96.4%	91.7%	87.3%	85.8%	73.0%	61.2%
12	96.1%	92.1%	86.2%	83.1%	72.7%	68.2%
13	93.0%	87.2%	88.6%	79.1%	70.2%	65.5%
14	93.9%	89.2%	85.6%	85.1%	75.4%	67.0%
15	97.4%	93.8%	84.5%	80.0%	74.7%	65.3%
16	96.3%	95.0%	86.8%	81.1%	68.5%	63.6%
17	95.5%	90.0%	87.1%	78.5%	67.7%	61.0%
18	93.7%	89.6%	83.9%	76.0%	64.9%	61.3%
19	92.1%	91.1%	84.1%	79.8%	78.2%	63.2%
20	98.4%	94.1%	85.1%	79.7%	72.6%	57.9%
Average	**96.0%**	**91.4%**	**85.5%**	**77.7%**	**69.7%**	**60.7%**

demonstrates that this need not be the case, as these units may share channels with units located farther away.

In the next two sections, we compare the performance of our MI-CAP solution methods for both unit and weighted penalties.

5.5.1 Comparison with Unit Penalties

Of our three MI-CAP solution methods, CP provides by far the best performance. We obtain superior results even when the method is run for just 180 seconds. The method obtains optimality (i.e., no pairwise constraint violations) in 55 of these 120 cases. In fact, the average number of radios receiving excessive interference and the total excessive

Table 5.8: Total number of pairwise violations using the MI-CAP CP formulation, with varying numbers of available channels, 500 second CP runtimes, and weighted penalties. Bold values indicate optimal solutions.

	Number of Pairwise Violations, by Channel Availability					
Time Step	**100%**	**90%**	**80%**	**70%**	**60%**	**50%**
1	**0**	**5**	12	18	33	46
2	**0**	**4**	**8**	20	29	53
3	**0**	4	10	27	46	78
4	**0**	4	11	21	35	55
5	**0**	**4**	11	21	40	62
6	**0**	6	11	20	34	54
7	**0**	**4**	10	21	30	60
8	**0**	5	16	30	42	72
9	**0**	**4**	8	18	32	59
10	**0**	**4**	8	19	34	57
11	**0**	**4**	8	14	34	53
12	**0**	**3**	**7**	**10**	23	37
13	**0**	**3**	9	17	31	50
14	**0**	4	9	15	34	47
15	**0**	**4**	**8**	13	25	43
16	**0**	4	9	13	26	41
17	**0**	**3**	10	18	28	47
18	**0**	**4**	**7**	14	27	45
19	**0**	**3**	7	11	25	48
20	**0**	**4**	10	16	31	49
Average	**0**	**4.0**	**9.5**	**17.8**	**32.0**	**52.8**

interference found by CP at the worst channel availability (50%) is comparable to the average values found using the clustering and IP methods with no channel degradation.

We provide a comparison of the relative optimality gaps achieved by each of our methods in terms of minimizing the number of pairwise violations (note this is not the objective of the clustering formulation). Figure 5.11 displays the results with unit penalties for each method (depicted in separate colors) and for each level of channel availability (depicted as separate lines), where in general the top line within a color group is 50% channel availability and the bottom line is 100% channel availability. It is clear from this figure that CP provides much more desirable performance.

Table 5.9: Percentage network availability (for radios) using the MI-CAP CP formulation, with varying numbers of available channels, weighted penalties, and 500 second runtimes.

	Network Availability (radios), by Channel Availability					
Timestep	**100%**	**90%**	**80%**	**70%**	**60%**	**50%**
1	93.5%	92.7%	83.4%	74.4%	66.7%	56.1%
2	97.4%	91.3%	87.2%	70.6%	67.8%	61.2%
3	96.7%	89.8%	85.6%	76.3%	64.0%	50.5%
4	99.4%	90.4%	88.7%	77.5%	66.6%	57.7%
5	96.3%	91.1%	82.6%	71.6%	65.7%	54.8%
6	95.9%	91.3%	87.7%	75.8%	74.1%	63.3%
7	97.2%	93.0%	82.7%	75.4%	68.5%	53.5%
8	97.8%	90.0%	81.2%	74.5%	63.7%	50.6%
9	95.4%	92.7%	88.1%	80.4%	75.7%	60.7%
10	95.9%	92.8%	89.1%	79.2%	66.6%	64.2%
11	96.2%	96.2%	86.3%	84.8%	70.6%	65.4%
12	95.9%	94.0%	86.5%	83.4%	77.1%	68.6%
13	92.7%	91.1%	84.8%	83.1%	71.1%	64.5%
14	93.6%	89.7%	87.3%	82.0%	77.1%	67.9%
15	97.2%	92.1%	87.8%	80.6%	73.5%	68.7%
16	96.1%	90.0%	89.7%	79.8%	73.9%	65.3%
17	95.3%	95.8%	86.0%	81.9%	74.9%	65.4%
18	93.4%	87.1%	87.4%	78.2%	75.3%	61.6%
19	91.7%	93.9%	88.9%	79.3%	68.0%	61.3%
20	98.3%	94.8%	90.7%	78.9%	68.6%	67.3%
Average	**95.8%**	**92.0%**	**86.6%**	**78.4%**	**70.5%**	**61.4%**

We then focus on more specific comparative results, considering the results of each solution method with 70% channel availability and unit penalties, in Figures 5.12 and 5.13. Figure 5.12 displays the total number of pairwise constraint violations at each time step (and on average), using each solution method. The lower bound T is indicated as a red hash mark. Note the objective function of the clustering formulation does not aim to minimize these violations. On average with 70% channel availability, the CP solution method finds solutions with 93.3% fewer pairwise interference constraint violations than the IP method, and 95.9% fewer than the clustering method.

In Figure 5.13, the solid lines display the total number of radios receiving excessive interference (indicated on the left axis), and the dashed lines display the total excessive

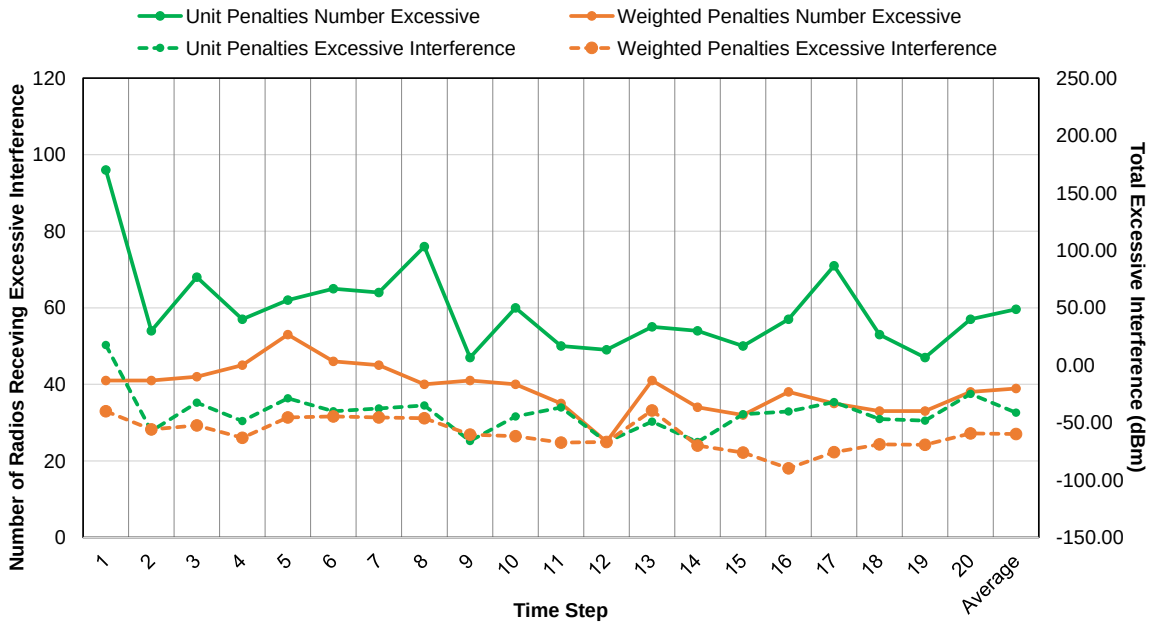


Figure 5.8: Comparison of number of radios receiving excessive interference (out of 1887), and the total received interference, between the unit penalty and weighted penalty CP methods, with 70% channel availability and 500 sec run times.

interference in dBm (indicated on the right axis). On average with 70% channel availability, the CP solution method finds solutions with nearly 54% fewer radios receiving excessive interference than the IP method, and 60% fewer than the clustering method. This results in the CP solutions having 84.1% less total excessive interference than the IP method, and 81.6% less than the clustering method (both in terms of watts).

In Figure 5.14, we present a comparison of the network availability results for each of MI-CAP methods (with unit penalties) (depicted in separate colors) and for each level of channel availability (depicted as separate lines), where in general the top line within a color group is 50% channel availability and the bottom line is 100% channel availability. Here again, it is clear that CP provides much more desirable performance.

We next provide a more qualitative comparison of the results generated by these methods, using the same type of graph as Figure 4.4. We consider the first time step of the

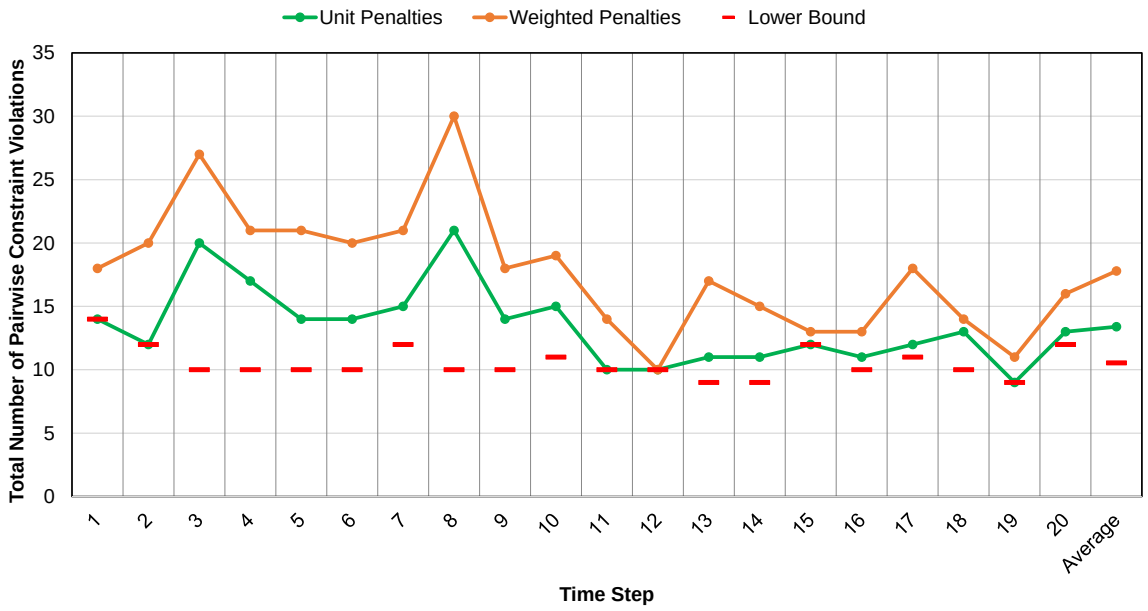


Figure 5.9: Comparison of the total number of pairwise constraint violations between the unit penalty and weighted penalty CP methods, with 70% channel availability and 500 sec run times.

MEF scenario with 70% availability and unit penalties. Figure 5.15 displays the results for the clustering method, Figure 5.16 for the IP method, and Figure 5.17 for the CP method. Points over the red line indicate that the associated radio receives excessive interference. Clearly, the CP solution provides much more desirable solutions than the other two methods.

5.5.2 Comparison with Weighted Penalties

Our spectrum manager is most concerned with minimizing the total received interference (M0), regardless of technique, and we observe that weighted penalties provides better performance for the IP and CP techniques. We again provide a comparison of the relative optimality gaps achieved by each of our methods in terms of minimizing the number of pairwise weighted violations. Figure 5.18 displays the results with weighted penalties for

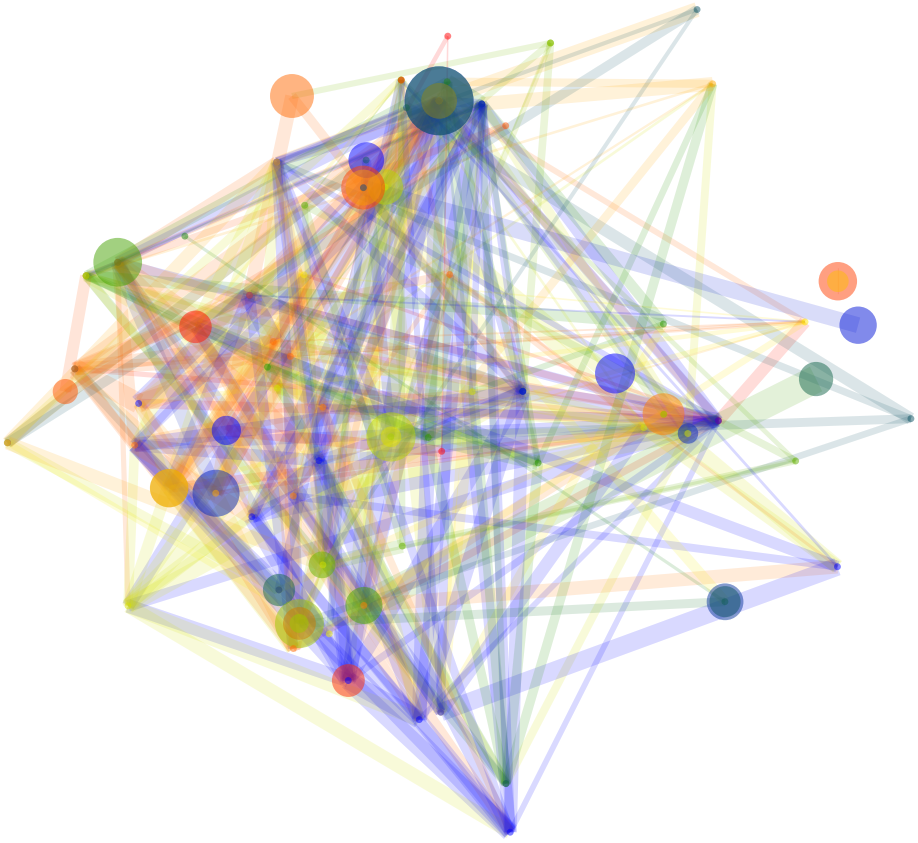


Figure 5.10: Depiction of co-channel interference during the first time step of the MEF scenario with 32 available channels. Color indicates channel assignment, the size of each node is relative to total excessive interference at that unit, and the width of each arc is relative to total co-channel interference between units.

each method (depicted in separate colors) and for each level of channel availability (depicted as separate lines), where in general the top line within a color group is 50% channel availability and the bottom line is 100% channel availability. It is clear from this figure that again CP provides much more desirable performance. Note the sharp spike when the IP method is able to obtain an optimal solution on time step 14, with 100% channel availability.

We now focus on more specific comparative results, considering the results of each solution method with 70% channel availability and weighted penalties, in Figures 5.19 and 5.20.

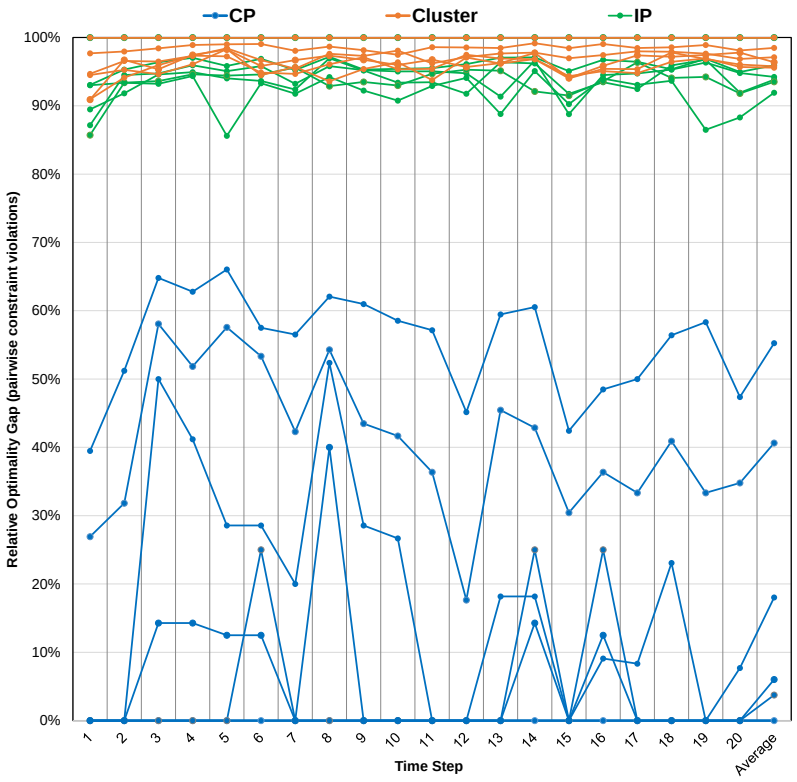


Figure 5.11: Relative optimality gap (in terms of pairwise constraint violations) for the three MI-CAP solution methods, for each time step and for various levels of channel degradation with unit penalties, where in general the bottom line in each group represents 100% channel availability.

IP and CPLEX with weighted penalties are run for 6000 seconds; CP with weighted penalties is run for 500 seconds, and clustering is run for 200 iterations. Figure 5.19 compares the number of radios receiving excessive interference (equivalent to the objective functions of the weighted IP and CP formulations) for the IP and CP methods. The lower bound is calculated using the method described in Section 5.3.3.2. On average with 70% channel availability, the CP solution method finds solutions with 37.3% fewer radios receiving excessive interference than the IP method.

In Figure 5.20, we compare all three solutions methods. The solid lines display the total number of radios receiving excessive interference (indicated on the left axis), and the

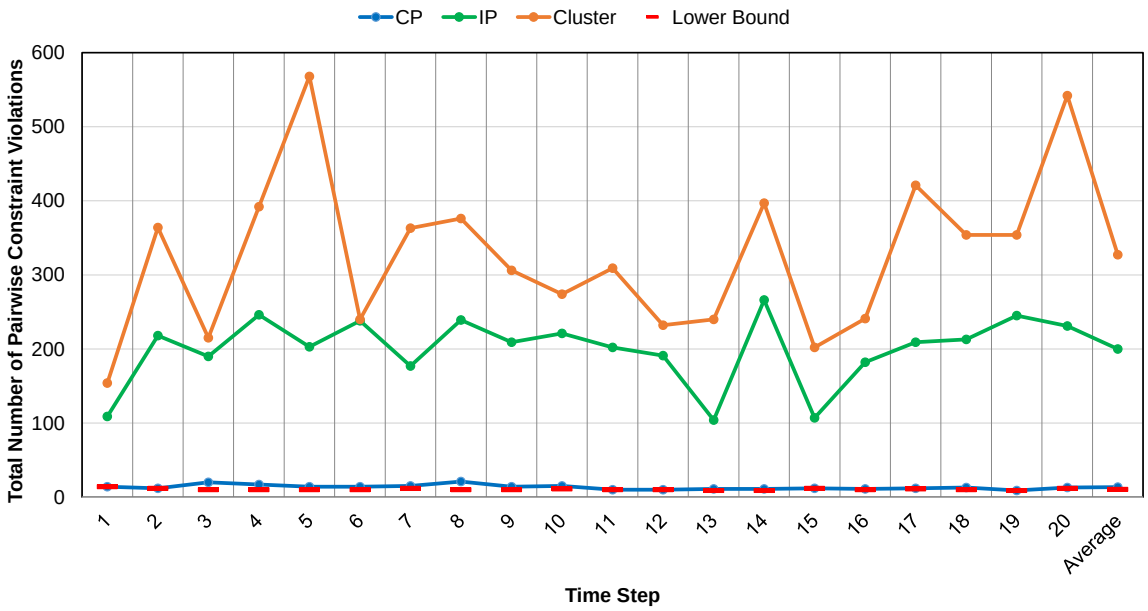


Figure 5.12: Comparison of the number of pairwise interference constraint violations in the MI-CAP, using the IP, clustering, and CP solution methods with 70% channel availability and unit penalties.

dashed lines display the total excessive interference in dBm (indicated on the right axis). On average with 70% channel availability, the CP solution method finds solutions with nearly 100% less total excessive interference than the IP and clustering methods (in terms of watts).

In Figure 5.21 we present a comparison of the network availability results for each of MI-CAP methods (with weighted penalties) (depicted in separate colors) and for each level of channel availability (depicted as separate lines), where in general the top line within a color group is 50% channel availability and the bottom line is 100% channel availability. The CP method continues to provide the best overall performance.

As with the unit penalty section, we next provide a more qualitative comparison of the results generated by these methods. We consider the first time step of the MEF scenario with 70% availability and weighted penalties. Figure 5.22 displays the results for the IP

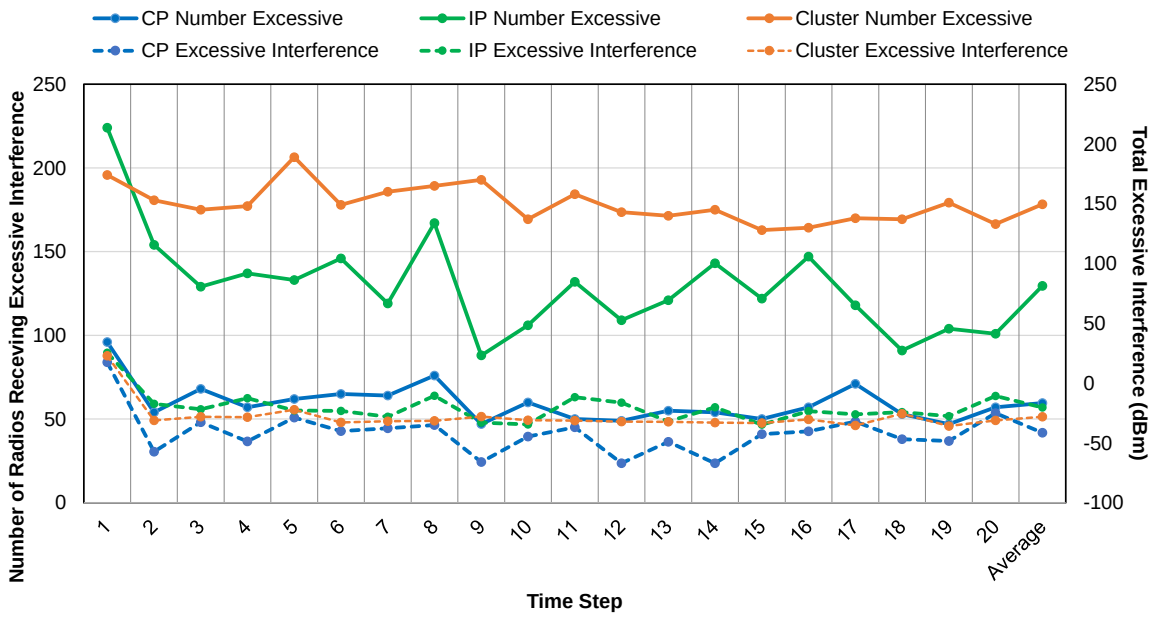


Figure 5.13: Comparison of the number of radios receiving excessive interference and the total cumulative excessive interference in the MI-CAP, using the IP (500 seconds), clustering (200 iterations), and CP (500 seconds) solution methods with 70% channel availability and unit penalties.

method and Figure 5.23 for the CP method (the clustering method is in Figure 5.15). Points over the red line indicate that the associated radio received excessive interference. The CP solution provides much more desirable solutions than the other two methods.

We note that none of our techniques provide useful lower bounds to the amount of total excessive interference. Aardal et al. (2007) note that the MI-CAP is a notoriously difficult problem for this reason. In Section 7.2, we recommend further research on how to find tighter lower bounds to the MI-CAP.

5.6 Estimating the Marginal Value of an Additional Channel

During interviews with the U.S. Marine Corps spectrum officer, Nicholas et al. (2013a) find that the number of available channels during a large USMC operation may be far fewer

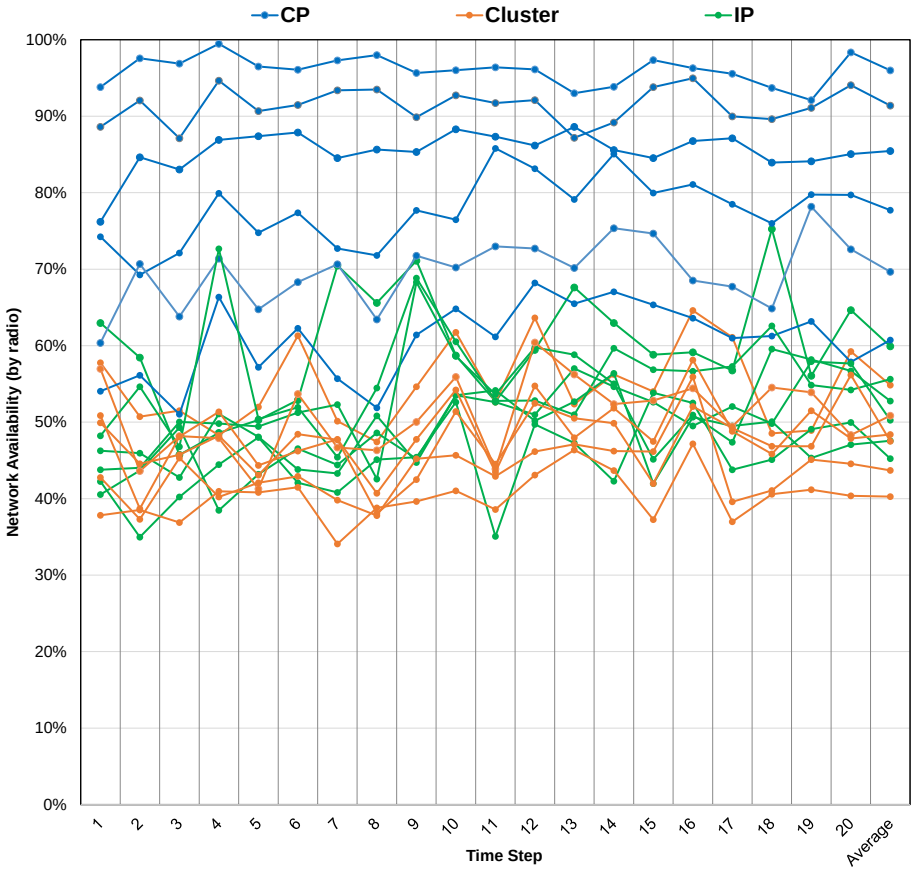


Figure 5.14: Network availability (in terms of radios) for the three MI-CAP solution methods, for each time step and for various levels of channel degradation with unit penalties, where in general the top line in each group represents 100% channel availability.

than indicated by our MO-CAP analysis. Our spectrum manager may wish to indicate to higher headquarters the marginal value (in terms of network availability) of one additional channel. That is, how much better will the MANETs perform if the Marine Expeditionary Force is allotted one additional channel? Our CP technique is fast enough to allow our spectrum manager to consider this problem.

To demonstrate, we run our weighted MI-CAP CP technique on the first time step, varying the number of available channels from 1 to 46 (the optimal MO-CAP solution) and with 500 second runtimes, and then calculate network availability as we did in Section 5.1.2.

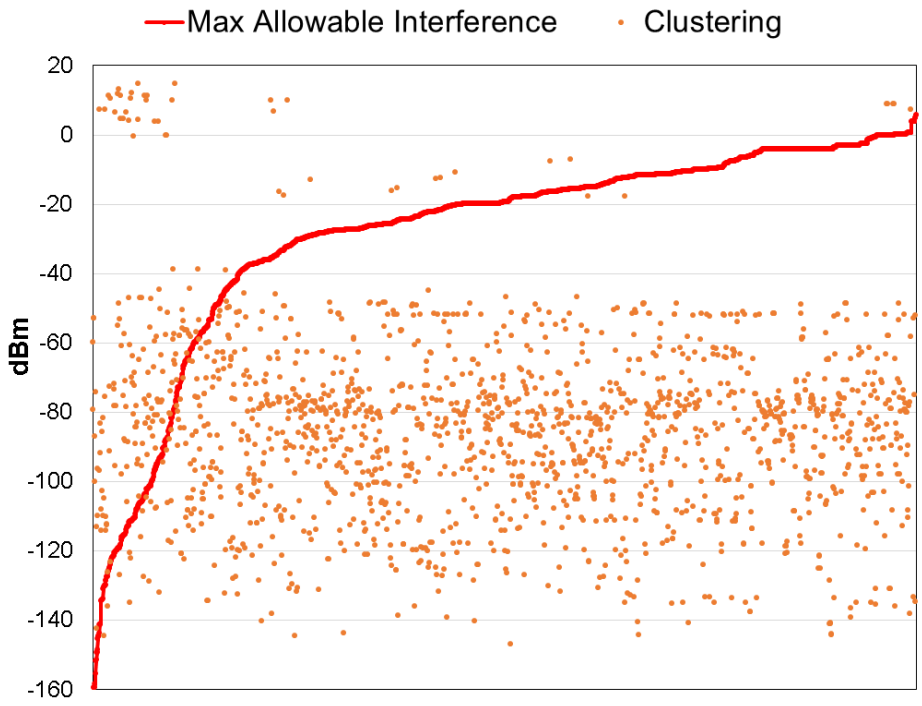


Figure 5.15: Depiction of the received interference (dots) and interference threshold (red line) for each of the 1887 radios in the MEF scenario (time step one), using the clustering method, 70% channel availability, and unit penalties.

The results are presented in Figure 5.24, where the horizontal axis indicates the number of available channels. The green line indicates the percentage of radios receiving excessive interference, and the blue and red lines respectively indicate the percentage network availability by radio and by unit. In general, there is increased excessive interference and reduced network availability as the number of available channels is reduced; the non-monotonic inconsistencies occur because of numerical differences in the progress of the CP Optimizer in solving the problem, i.e., the problem is not solved to optimality.

On average, the inclusion of one additional channel results in an increase of network availability (by radios) of approximately 4.7%. This information can be quickly generated by our CP technique and used by a spectrum manager to justify additional spectrum.

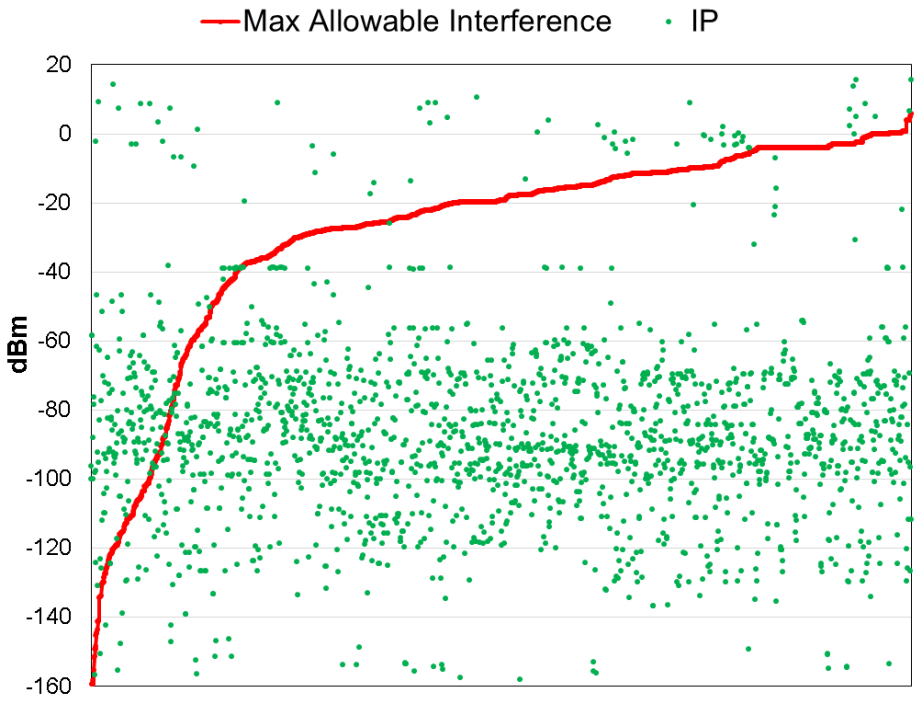


Figure 5.16: Depiction of the received interference (dots) and interference threshold (red line) for each of the 1887 radios in the MEF scenario (time step one), using the integer programming method, 70% channel availability, and unit penalties.

5.7 MI-CAP Sensitivity Analysis

As we did with the MO-CAP, we conduct sensitivity analysis on our MI-CAP CP formulation and solution method to determine its robustness to small perturbations in inputs. Specifically, we randomly perturb our received signal strength values ρ_{rs} by up to $\pm 10\%$ (uniform random distribution), and then re-run our method, for each time step and with 70% channel availability. In each case, we find that the number of radios receiving excessive interference is roughly the same as the control case (i.e., no perturbation in input values) (left side of Table 5.10).

We next wish to examine how much the solution itself (i.e., the assignment of channels to units) changes due to these perturbations. We use the method described in Section 4.5, and

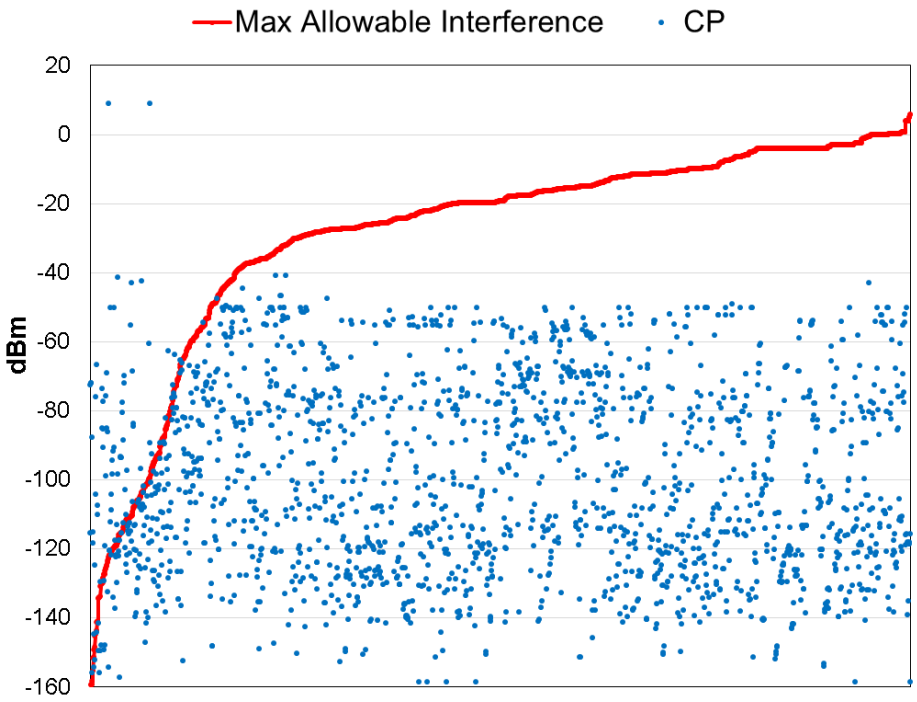


Figure 5.17: Depiction of the received interference (dots) and interference threshold (red line) for each of the 1887 radios in the MEF scenario (time step one), using the constraint programming method, 70% channel availability, and unit penalties.

present the results in the right side of Table 5.10, where each entry indicates the percentage of units that must be assigned to a new channel (compared to the unperturbed control case), for each time step and with varying levels of perturbation from ±0.5% to ±10%. As with the MO-CAP (Table 4.6) and again due to the vast symmetry in the problem, even small levels of perturbation result in large percentages of units being assigned to different groups.

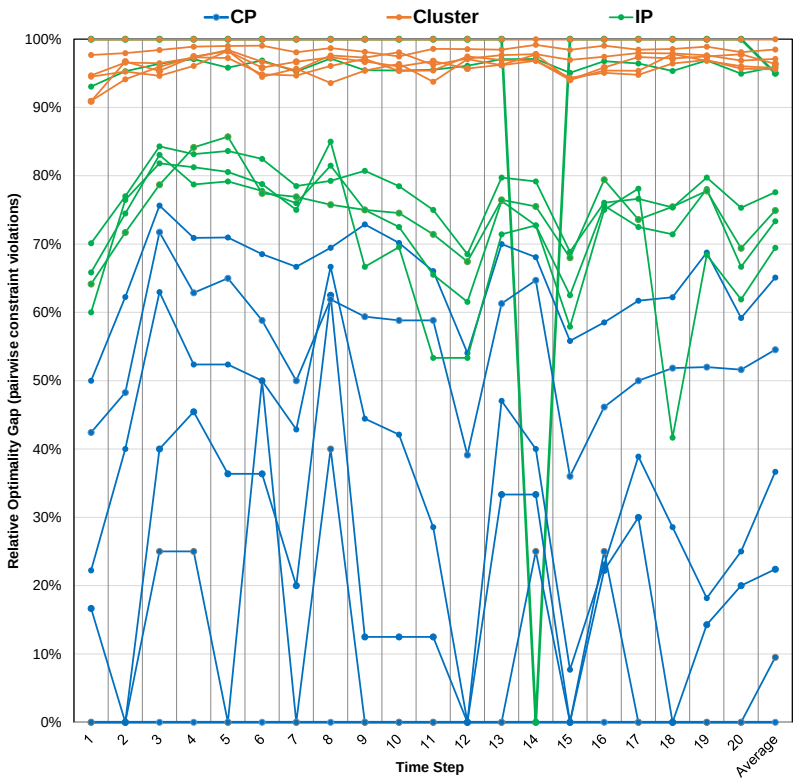


Figure 5.18: Relative optimality gap (in terms of pairwise constraint violations) for the three MI-CAP solution methods, for each time step and for various levels of channel degradation with weighted penalties, where in general the bottom line in each group represents 100% channel availability.

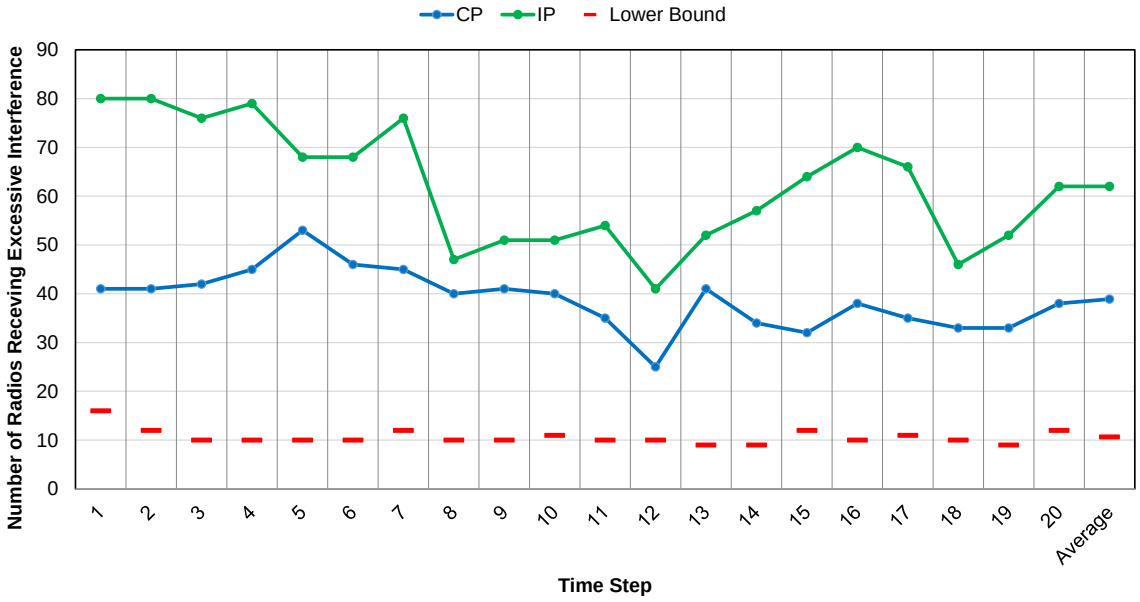


Figure 5.19: Comparison of the number of radios receiving excessive interference in the MI-CAP, using the IP (6000 seconds) and CP (500 seconds) solution methods with 70% channel availability and weighted penalties.

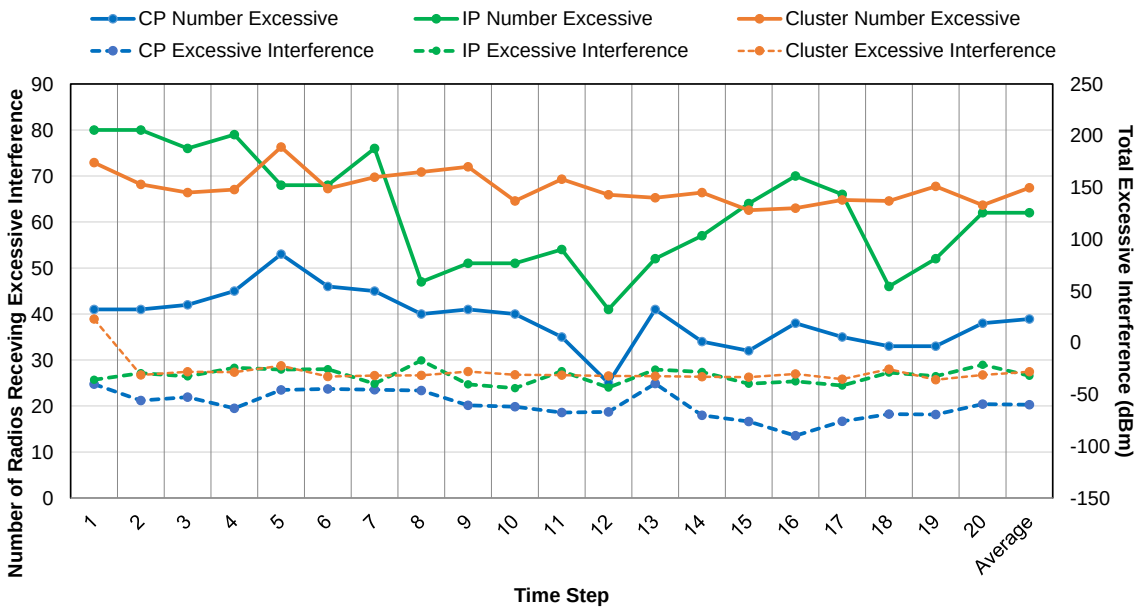


Figure 5.20: Comparison of the number of radios receiving excessive interference and the total cumulative excessive interference in the MI-CAP, using the IP (500 seconds), clustering (200 iterations), and CP (500 seconds) solution methods with 70% channel availability and weighted penalties.

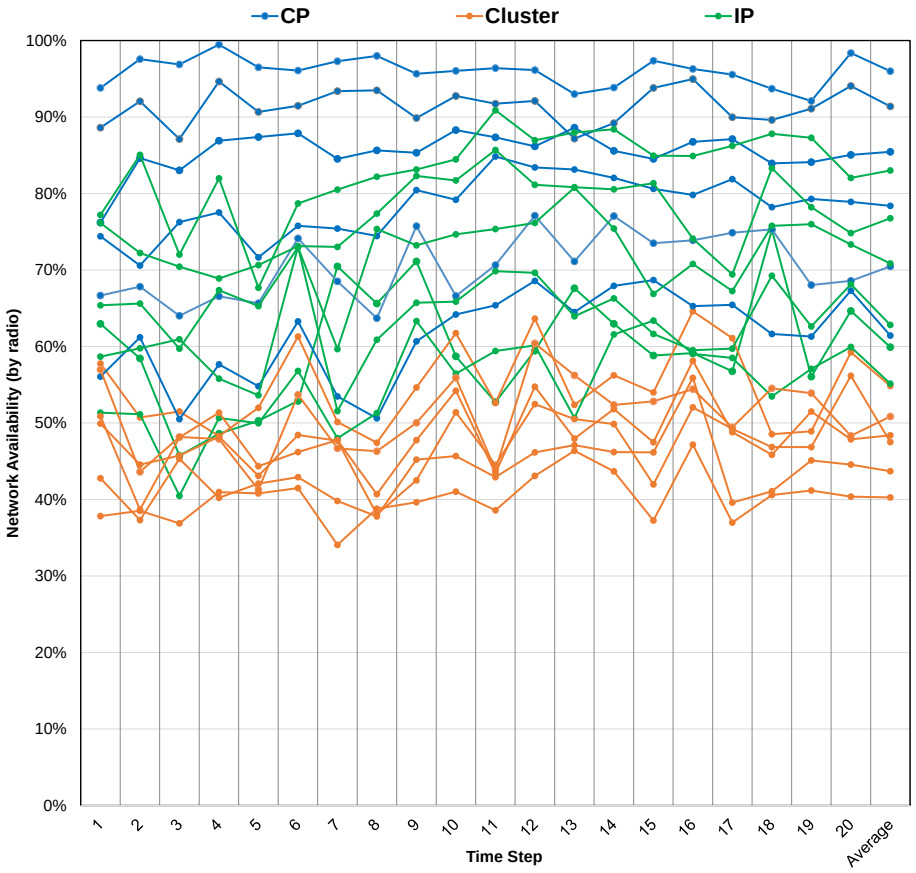


Figure 5.21: Network availability (in terms of radios) for the three MI-CAP solution methods, for each time step and for various levels of channel degradation with weighted penalties, where in general the top line in each group represents 100% channel availability.

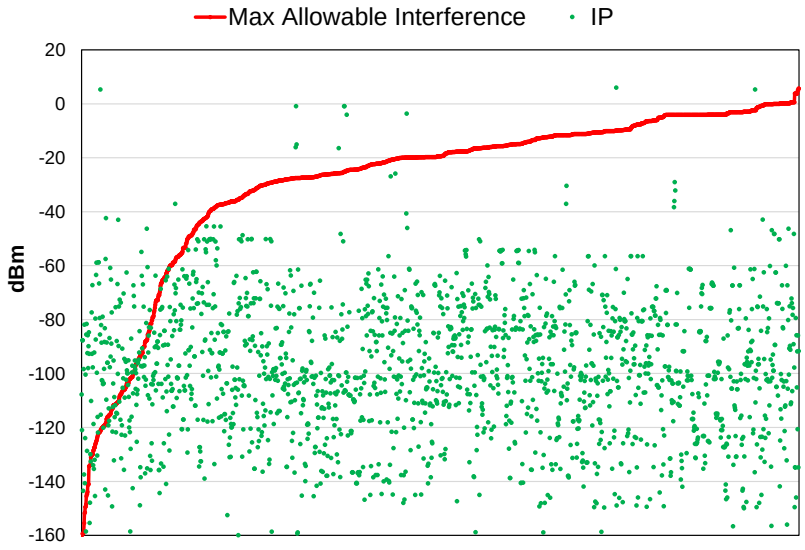


Figure 5.22: Depiction of the received interference (dots) and interference threshold (red line) for each of the 1887 radios in the MEF scenario (time step one), using the integer programming method, 70% channel availability, and weighted penalties.

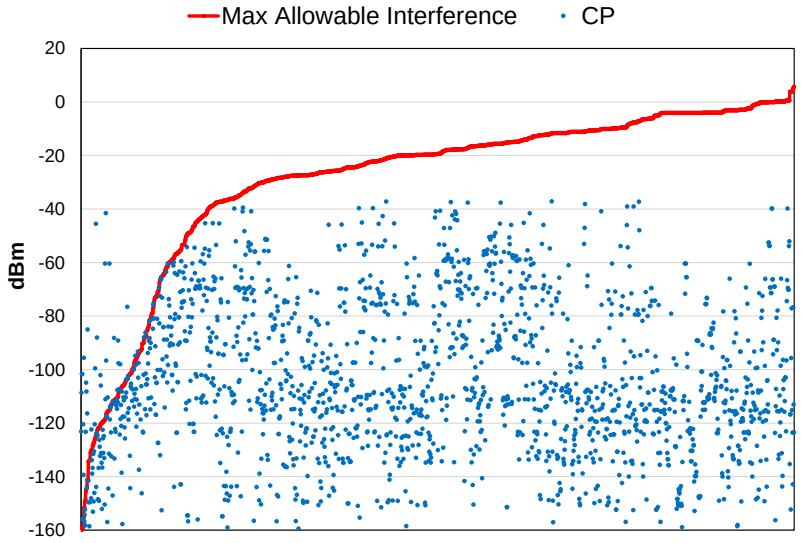


Figure 5.23: Depiction of the received interference (dots) and interference threshold (red line) for each of the 1887 radios in the MEF scenario (time step one), using the constraint programming method, 70% channel availability, and weighted penalties.

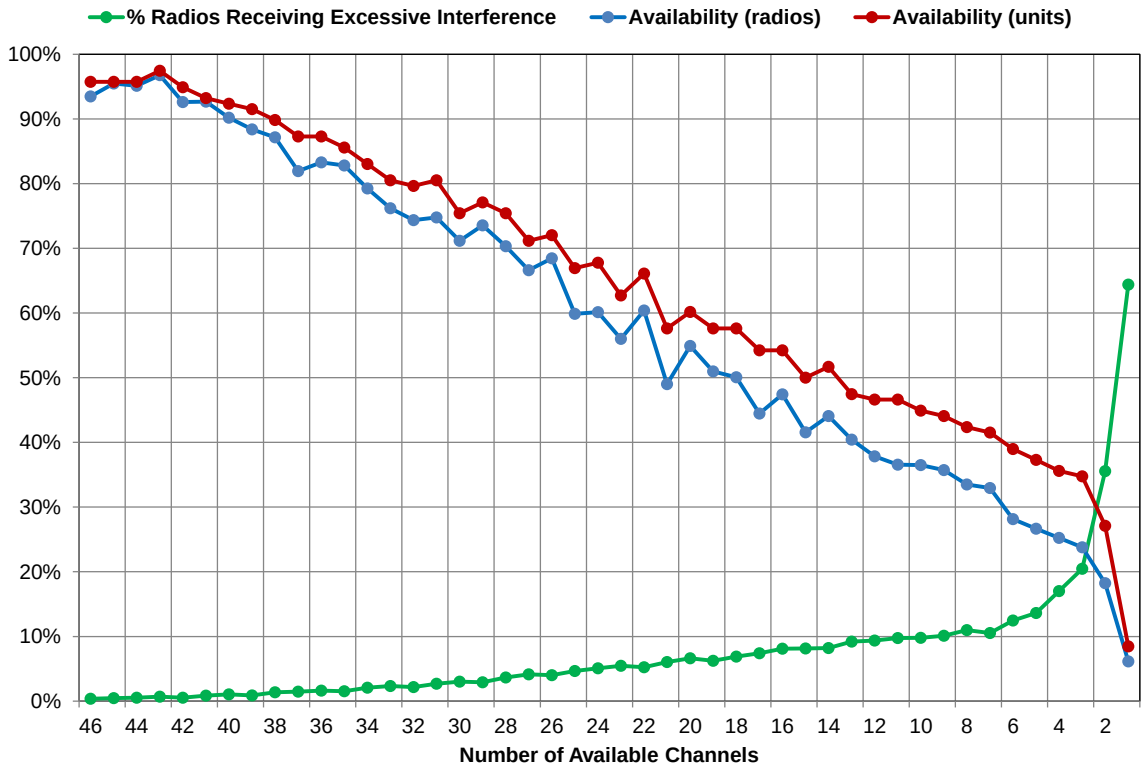


Figure 5.24: Percentage of radios receiving excessive interference, and network availability (by percentage) of individual radios and units, for the first time step of the MEF scenario when solved using the MI-CAP CP technique with weighted penalties and 500 second runtimes, for 1-46 available channels. This provides an indication of the marginal value of an additional channel.

Table 5.10: MI-CAP sensitivity analysis results for each time step, including the number of radios receiving excessive interference and the percentage of units that must change channel (compared to the unperturbed control case), for given levels of random perturbation of input values.

	Radios Receiving Excessive Interference					**% Units Changing Groups**			
Time Step	**No Perturbation**	±0.5%	±1.0%	±5%	±10%	±0.5%	±1.0%	±5%	±10%
1	96	6	84	96	107	51.7%	46.6%	58.5%	50.8%
2	54	101	52	63	57	49.2%	49.2%	54.2%	52.5%
3	68	51	79	77	62	58.5%	53.4%	55.9%	53.4%
4	57	81	60	67	55	0.0%	47.5%	55.9%	50.8%
5	62	56	72	59	62	0.0%	55.1%	52.5%	62.7%
6	65	62	79	57	65	0.0%	52.5%	61.0%	53.4%
7	64	65	65	68	67	53.4%	51.7%	60.2%	53.4%
8	76	61	90	70	78	50.0%	61.9%	51.7%	56.8%
9	47	76	66	64	58	0.0%	57.6%	56.8%	55.1%
10	60	47	57	62	54	0.0%	55.1%	59.3%	51.7%
11	50	60	55	54	50	45.8%	51.7%	57.6%	44.1%
12	49	53	57	59	52	0.0%	59.3%	61.0%	55.9%
13	55	49	51	58	60	46.6%	50.8%	47.5%	51.7%
14	54	60	64	55	59	53.4%	55.9%	62.7%	34.7%
15	50	53	49	49	48	55.1%	0.0%	51.7%	55.9%
16	57	51	53	51	55	0.0%	46.6%	52.5%	52.5%
17	71	57	65	65	58	52.5%	52.5%	55.9%	55.9%
18	53	52	56	52	57	49.2%	59.3%	50.8%	55.1%
19	47	47	56	49	55	46.6%	55.1%	57.6%	47.5%
20	57	51	60	56	64	0.0%	49.2%	46.6%	53.4%
Average	**59.6**	**57.0**	**63.5**	**61.6**	**61.2**	**30.6%**	**50.6%**	**55.5%**	**52.4%**

Chapter 6: Minimum-Cost Channel Assignment Problem over Time

This chapter describes the minimum-cost channel assignment problem over time (MC-CAP-T), which aims to minimize the total received interference, given a fixed number of channels. This problem reflects the real-world challenge of a spectrum manager attempting to minimize the total number of channel changes over time, given MO-CAP or MI-CAP solutions at each moment (or time step). We wish to avoid myopic "flip-flopping" solutions because they waste the time of the radio operators and require coordination and synchronization among potentially many dispersed units, which may be difficult to achieve in battlefield conditions.

One straightforward approach would simply modify the MO-CAP objective function by introducing an additional index $t \in T$ to each variable and input parameter to represent *time steps*. One could also introduce a penalty term, say p, to penalize changing channels from one step to the next. The objective function could thus be stated:

$$\min \sum_{t \in T} \sum_{c \in C} \sum_{u \in U} p|X_u^{ct} - X_u^{c,t+1}|. \tag{6.1}$$

That is, we could aim to minimize the number of times a radio must change channels over time. However, we explore this and similar formulations and find this approach to be computationally intractable. Essentially this formulation seeks alternate optimal MO-CAP or MI-CAP solutions at each time step in order to reduce the number of required channel changes. Such searches are known to be computationally challenging and often relatively fruitless because the alternate optima are often not located anywhere near each other within the solution tree. That is, having obtained solutions to the MO-CAP or MI-CAP does

not markedly increase our ability to find other comparable solutions. Our exploratory experiments with this type of formulation are insolvable in any reasonable amount of time.

Instead, we define the scope of our temporal problem to use the solution of the MO-CAP or MI-CAP as an input, and then assign actual channel numbers (or colors) to units, rather than also (and concurrently) considering the assignment of units to channels. To illustrate this difference, we introduce the concept of a group g of units that co-occupy a channel at a given time step. These groups follow from the solutions obtained at each time step from either the MO-CAP or MI-CAP problems. We wish to determine an actual channel number (e.g., channel 5) to assign to each group at each time step. A *naïve* approach would simply assign channel numbers to the groups as they appear in order. In practice, this produces surprisingly bad solutions as group membership (i.e., the units assigned to each group) may change significantly from time step to time step, and thus an excessively large cost is incurred if one simply dictates that group 1 is always assigned channel 1, etc.

The minimum-cost channel assignment problem over time (MC-CAP-T) aims to minimize the cost incurred by channel changes over time by assigning channels to groups of units. We first describe a full standard formulation of the problem and note the difficulties in solving it. We then describe our decomposition formulation that solves to optimality in polynomial time, and present our results.

6.1 MC-CAP-T Full Standard Formulation

To describe the full MC-CAP-T problem, we introduce the index $t \in T$ to represent each time step. Let $c \in C^t$ indicate the channels at each time, where $|C^t|$ is the number of channels required at time t. Let $g^t \in G$ be a group of units that co-occupy a channel at time t, where $g^t \subset U$ and g^t is indexed by $u_1, u_2, \ldots, u_{|g^t|}$. These groups are pre-calculated using the results of either the MO-CAP or MI-CAP, or other solution technique.

Let the binary variable X_u^{ct} indicate if unit u is using channel c at time t:

$$X_u^{ct} = \begin{cases} 1, & \text{if unit } u \text{ uses channel } c \text{ at time } t \\ 0, & \text{otherwise} \end{cases} \qquad \forall u \in U, c \in C^t, t \in T. \tag{6.2}$$

We wish to count the number of times a unit changes channel, so let the binary variable W_u^t indicate whether unit u changes channel from t to $t+1$:

$$W_u^t = \begin{cases} 1, & \text{if unit } u \text{ changes channel from } t \text{ to } t+1 \\ 0, & \text{otherwise} \end{cases} \qquad \forall u \in U, t \in T. \tag{6.3}$$

To enforce this definition, we include the constraints:

$$W_u^t \geq X_u^{c,t+1} - X_u^{ct} \qquad \forall c \in C^t, t \in T, t \neq |T|. \tag{6.4}$$

That is, W_u^t is one if a unit u is assigned channel c at $t+1$, but not at t. (Defining W_u^t in only this direction prevents double-counting of channel changes.)

Each time a unit is assigned a new channel, all radios in that unit must change channels, so we use the scalar $radios_u$ to indicate the number of radios assigned to unit u. Our objective function is thus:

$$\min \sum_{t \in T} \sum_{u \in U} radios_u W_u^t. \tag{6.5}$$

Our Full Standard Formulation of the MC-CAP-T is summarized as follows:

MC-CAP-T Full Standard Formulation

Index and Set Use

$u \in U$	unit (*alias* v)
$t \in T$	time step
$c \in C^t$	channel at time t, where $\|C^t\|$ is the number of channels available at time t
$g^t \in G$	group of units (i.e., units on the same channel) at time t

Input Data

$radios_u$	number of radios assigned to unit u

Decision Variables

X_u^{ct}	binary variable indicating whether u is using c at t
W_u^t	binary variable indicating whether u must change channel from t to $t+1$

Formulation

$$\min_{W,X} \quad \sum_{t \in T} \sum_{u \in U} radios_u W_u^t \tag{T0}$$

$$X_u^{ct} = X_v^{ct} \qquad \forall u, v \in g^t, c \in C^t, t \in T \tag{T1}$$

$$\sum_{c \in C} X_{u_1}^{ct} = 1 \qquad \forall u_1 \in g^t, g^t \in G, t \in T \tag{T2}$$

$$\sum_{u_1 \in g^t} X_{u_1}^{ct} = 1 \qquad \forall g^t \in G, c \in C^t, t \in T \tag{T3}$$

$$W_u^t \geq X_u^{c,t+1} - X_u^{ct} \qquad \forall c \in C^t, t \in T, t \neq |T| \tag{T4}$$

$$W_u^t \in \{0,1\} \qquad \forall u \in U, t \in T \tag{T5}$$

$$X_u^{ct} \in \{0,1\} \qquad \forall u \in U, c \in C^t, t \in T \tag{T6}$$

The MC-CAP-T FSF is a pure 0-1 integer program. The objective function (T0) minimizes the total cumulative cost of changing channels from one time step to the next. Constraints (T1) ensure that all units within a group are assigned the same channel, at each time step. Constraints (T2) ensure that the first unit u_1 in each group $g^t \in G$ is assigned exactly one channel; together with (T1), this ensures all units in each group are assigned the same channel. Constraints (T3) ensure each available channel at time t is used exactly once. Constraints (T4) enforce the definition of W_u^t. Notice that constraints (T2) and (T3) are

constraints of the classic integer *assignment problem*, a network problem with polynomial time complexity. We will exploit this fact in our decomposition method.

6.1.1 MC-CAP-T FSF Solution Method and Computational Challenges

We solve the MO-CAP or MI-CAP problems to optimality for each time step, and then use the solutions of the problem to determine group assignments $g^t \in G$. We then unsuccessfully attempt to solve the MC-CAP-T FSF directly using CPLEX. While the problem has structure similar to that of the assignment problem, the FSF has additional constraints (T1 and T4) that make this formulation much less efficient. After 1000 seconds of computation, the solver has a 90% optimality gap, even when just considering one time step. Clearly, this method is not suited for processing multiple time steps concurrently.

6.2 MC-CAP-T Decomposition Formulation

We reformulate the problem based on the key insights that the actual channel number (or color, or any other label) is arbitrary; that is, we do not particularly care what channel is assigned to a group of units, as we assume that the net effect of different operating frequencies is negligible. We also observe that the cost of changing the channel assignment of a group from t to $t+1$ depends only on the unit membership of each group at t and $t+1$; that is, the costs of channel assignment can be decomposed by time step. Together, these insights allow us to decompose and solve the problem by time step and still maintain global convergence.

Our MC-CAP-T *Decomposition Formulation* (DF) aims to associate each group g at time t to a group h at time $t+1$ at least cost, where $cost^t_{gh}$ is a function of the difference in unit membership between g and h. Specifically,

$$cost^t_{gh} \equiv \sum_{u \in h \setminus g} radios_u \qquad \forall\, (g,h,t) \in A. \tag{6.6}$$

That is, the cost from g to h is the number of radios from units that are in group h but not in group g. This method of calculating costs prevents double-counting when a unit moves from an existing channel to a new channel. Note this cost function assumes all units and radios have the same importance, but that need not be the case: one could associate scalar weights to each radio to model relative importance.

Let the binary variable Y_{gh}^t indicate if group g at t is associated with group h at $t+1$, and let $(g,h,t) \in A$ be the arcs representing the set of possible associations between groups g and h:

$$Y_{gh}^t = \begin{cases} 1, & \text{if } g \text{ at } t \text{ is associated with } h \text{ at } t+1 \\ 0, & \text{otherwise} \end{cases} \qquad \forall\, (g,h,t) \in A. \tag{6.7}$$

One could simplify this formulation further by dropping the $t \in T$ index, but we retain the notation to aid in describing our decomposition approach. Our objective function minimizes the sum total costs of associating each group g at t with group h at $t+1$:

$$\min \sum_{(g,h,t)\in A} cost_{gh}^t Y_{gh}^t. \tag{6.8}$$

Figure 6.1 provides a visual representation of the process of associating groups at each time step, where for each time step the column of squares on the left represents groups g and on the right represents groups h. The number of groups (and their unit membership) is determined by the solutions from the MO-CAP or MI-CAP, so some time steps may have more or fewer groups than others. For those time steps with fewer groups than the maximum, we create *virtual groups* (indicated in Figure 6.1 by dashed lines), representing a placeholder group with no assigned units. In this sense, a group in this formulation represents both a collection of units to be assigned the same channel, and a placeholder for the channel itself, i.e., $|G|$ is equal to the number of available channels across the scenario.

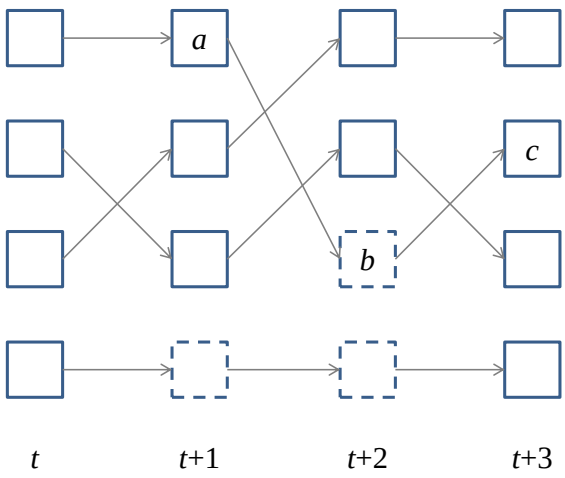


Figure 6.1: Example of the association of groups (blue boxes) at each time step. Virtual groups (comprising no units) are represented by dashed lines. Gray arrows indicate the association of a group at a given time step with another group at the next time step.

At each time step, each group g must be associated with a group h, indicated by gray lines between groups in Figure 6.1. We illustrate via two examples. When a real group a (i.e., comprising units) at $t+1$ is associated with a virtual group b at $t+2$, no cost is incurred because the units in a are assigned to other groups (not in b) at $t+2$. When a virtual group b is associated with a real group at c at $t+3$, the cost equals the total number of radios in c, since each unit in c was previously assigned to a different group.

Note that in this formulation (unlike the Full Standard Formulation), there is no variable or index representing a particular channel; the association Y_{gh}^{t} implies one. After solving the problem, the paths created by associating each g with an h at the next time step (i.e., the gray lines in Figure 6.1) represent discrete channels. By assigning a channel number to each of these paths, we effectively solve the MC-CAP-T.

The Decomposition Formulation of the MC-CAP-T is summarized as follows:

MC-CAP-T Decomposition Formulation

Index and Set Use

$g \in G$	group of units (i.e., units on the same channel) (*alias* h)
$t \in T$	time step
$(g, h, t) \in A$	arcs representing possible association of g at t with h at $t+1$

Input Data

$cost^t_{gh}$	cost of associating g at t to h at $t+1$ [number of radios]

Decision Variables

Y^t_{gh}	binary variable indicating whether group g at t is associated with group h at $t+1$

Formulation

$$\min_{Y} \sum_{(g,h,t)\in A} cost^t_{gh} Y^t_{gh} \tag{D0}$$

$$\sum_{h\in G} Y^t_{gh} = 1 \qquad \forall\,(g,\cdot,t) \in A \tag{D1}$$

$$\sum_{g\in G} Y^t_{gh} = 1 \qquad \forall\,(\cdot,h,t) \in A \tag{D2}$$

$$Y^t_{gh} \in \{0,1\} \qquad \forall\,(g,h,t) \in A \tag{D3}$$

The MC-CAP-T Decomposition Formulation is a pure 0-1 integer program. The objective function (D0) minimizes the cost of associating each group g with a group h at successive time steps. Constraints (D1) ensure that each group g at t is associated with a group h at $t+1$. Likewise, constraints (D2) ensure that each group h at $t+1$ is associated with a group g at t.

This formulation is considerably simpler than the Full Standard Formulation, and has fewer decision variables. Further, we observe that this problem has special structure that allows us to decompose the problem by time step. Specifically, at each time step we are solving a classic integer assignment problem (as evidenced by the assignment constraints (D1) and (D2)). Global convergence is maintained because at each time step, the cost of channel changes depends only on the assignments at t and $t+1$. The actual assigned channel (i.e., its number) is arbitrary, since all channels provide the same performance and each group must have a channel. Thus this formulation exhibits *optimal substructure* that

allows us to efficiently solve each time step to optimality and then combine our results to solve the entire problem to optimality.

We observe that this type of decomposition does not apply to the MC-CAP-T Full Standard Formulation, because the decision to assign a unit u to a channel c at t will affect both the costs of transitioning from $t-1$ to t and from t to $t+1$.

6.2.1 MC-CAP-T Decomposition Formulation Solution Method

Leveraging the optimal substructure of the problem, we decompose each time step of the problem sequentially. Since we can completely decompose the problem by time step (i.e., the decisions at each stage do not depend on others), one could also use a parallel approach to solve each problem simultaneously. One could also solve this problem as a *minimum-cost network flow problem* (Ahuja et al. 1993), which we also do using the network optimizer in CPLEX and obtain the same solutions (and thereby verify our decomposition approach). However, our decomposition approach is more amenable to the consideration of new, additional time steps later in a scenario (or indeed, in between two existing time steps), as it does not require that we resolve the entire problem. If new time steps are considered, then only the MO-CAP/MI-CAP problems for those steps need to be solved. Similarly, if only some of the time steps have changes, then only those MO-CAP/MI-CAP problems need to be re-solved, rather than every time step.

We implement our solution in Python. We first calculate all of the $cost^t_{gh}$ values for each possible $(g, h, t) \in A$. For our MEF scenario with 20 time steps, this is $46 \times 45 \times (20-1) = 40,204$ values (we do not need to calculate costs to arrive at the first time step). We then solve the assignment problem at each time step using a variation of the Hungarian (or Munkres) algorithm (Kuhn 1955), which solves to global optimality in $\mathcal{O}(n)^3$ time. One may assign channels after each iteration of the assignment problem (our approach), or "follow the path" through each time step after solving the entire problem. We provide partial Python code for solving the MC-CAP-T in Appendix F. The following pseudo-code describes our algorithm:

Algorithm *MC-CAP-T*

Input: MO-CAP or MI-CAP solutions at each time step

Output: $X_u^{ct}, \forall u \in U, c \in C, t \in T$ (unit channel assignments for all time steps)

begin
    Calculate $cost_{gt}^t, \forall (g, h, t) \in A$
    $channel \leftarrow 1$
    for $g \in G : t = 1$
        $\Gamma_g \leftarrow channel$ // Assign channels to groups during first time step
        $channel \leftarrow channel + 1$
    next;
    for $t = 1, 2, \ldots, t - 1$
        Solve the MC-CAP-T for t using Hungarian / Munkres algorithm
        Store Y_{gh}^t values
        for $g, h \in (g, h, t)$
            if $Y_{gh}^t = 1$
                $\Gamma_h \leftarrow \Gamma_g$ // Assign channels to groups for time step t
            endif;
        next;
    next;
    for $g \in G$
        for $u \in g$
            $X_u^{\Gamma_g^t} \leftarrow 1$ // Assign channels to units
        next;
    next;
end;

6.2.2 MC-CAP-T Decomposition Formulation Results

We first use a naïve method of determining channel assignment, based on the order that groups appear in a solution. Next, we use our methodology, including using Python and the Munkres / Hungarian algorithm to solve an assignment problem at each time step, and then determine channel assignments based on the solutions to these sub-problems.

Figure 6.2 displays a comparison of the naïve method and our decomposition method, where the vertical axis indicates the number of required channel changes, for each time step in the MEF scenario using MO-CAP solutions as inputs (solved via the lazy constraint

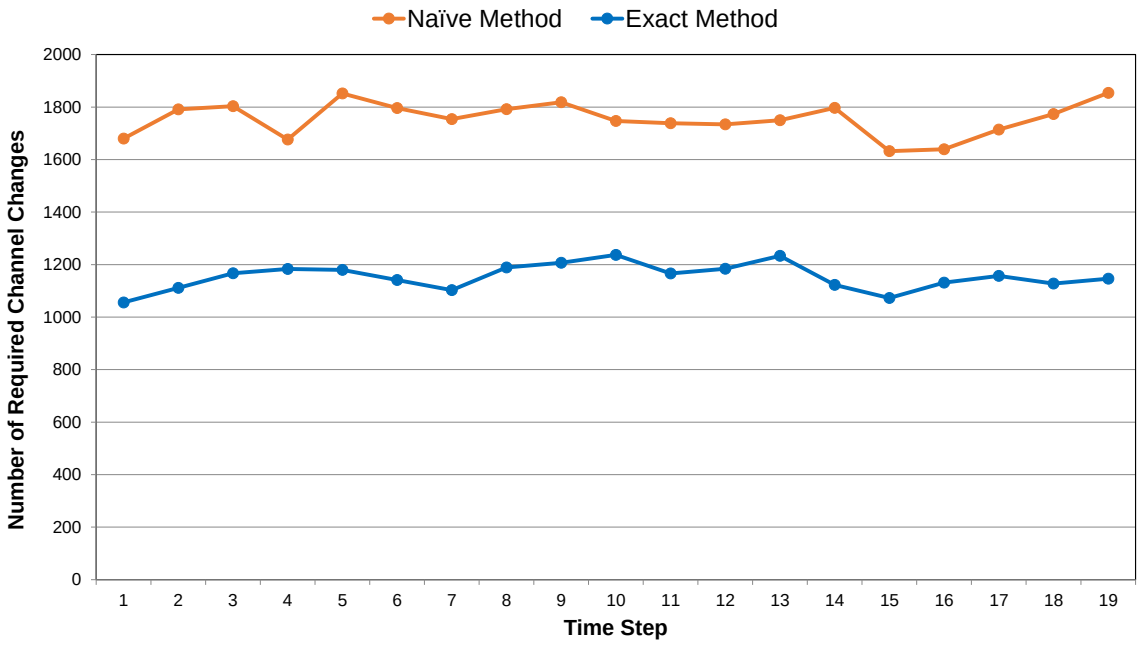


Figure 6.2: Number of required channel changes in the MC-CAP-T, using the naïve and exact methods.

method). The naïve method requires a total of 33,340 channel changes, whereas our decomposition method (which solves in less than 53 seconds) requires 21,915 channel changes, a reduction of 34%.

Figure 6.3 is another method of visualizing the results of this comparison. For both the naïve and decomposition methods, a row represents a unit, where reddish units are larger (comprising up to 25 radios each) and greenish units are smaller, each column represents a time step, and a blank entry indicates that no channel change is required for that unit at that time step. This visualization provides a qualitative sense of how much better the decomposition method (which provides an exact solution) is at reducing channel changes, especially for larger (and thus more penalizing) units.

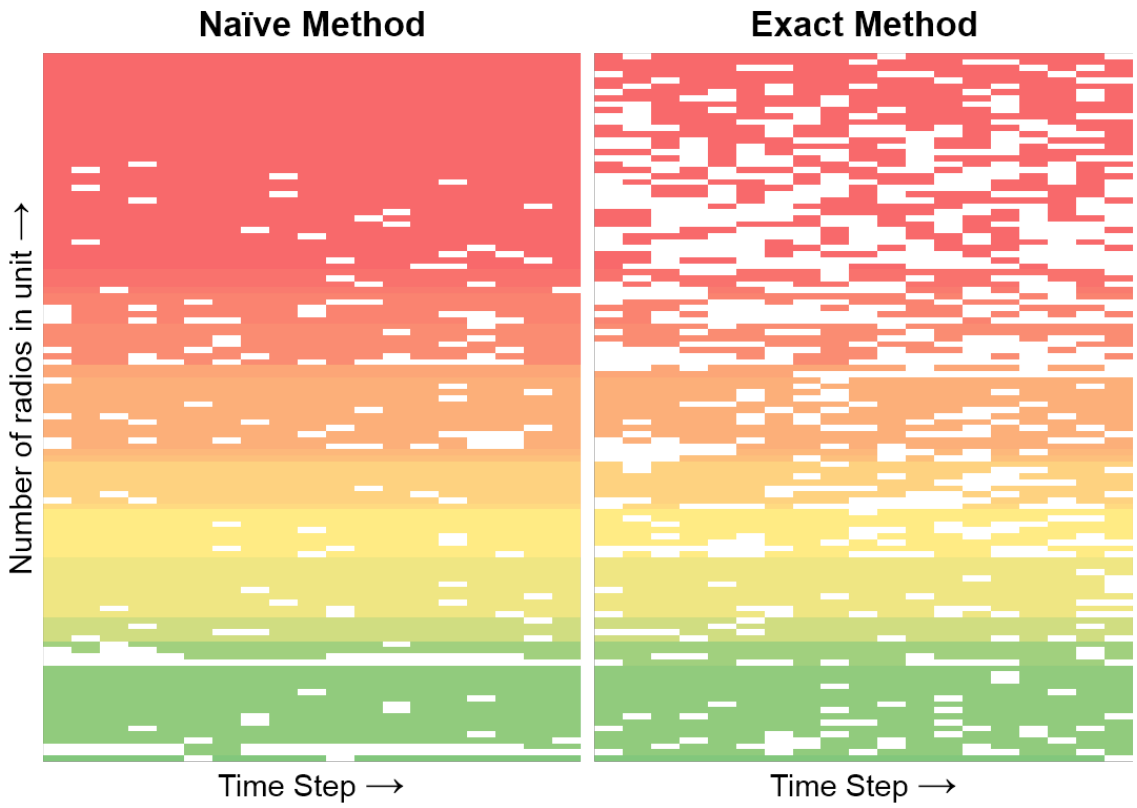


Figure 6.3: Results of MC-CAP-T, where each row represents a unit, and each column represents a time step. White indicates that the channel assignment remains the same (i.e., no cost), and color indicates that a different channel is assigned at the next time step. Red indicates larger units (more radios); green indicates smaller units.

Our decomposition method very quickly provides an exact solution to the MC-CAP-T, and unlike other formulations, does not need to be completely resolved when new or different time steps are introduced into a scenario.

Chapter 7: Conclusions and Future Research

7.1 Conclusions

We consider the challenges faced by a spectrum manager in allocating spectrum to support MANET radios in a tactical military environment. During planning before an operation, the spectrum manager may wish to determine the minimum number of required channels to support operations (i.e., MO-CAP). If the resulting number of channels is larger than can be provided, or if after an allocation the situation changes and fewer channels are available, the spectrum manager may now need to determine the best allocation of the available channels. In this case, "best" means minimizing the total amount of received excessive interference (i.e., MI-CAP). Throughout operations, the spectrum manager wishes to minimize the total number of channel changes over time (i.e., MC-CAP-T), as each change requires coordination across the battlefield and manual intervention by a radio operator.

We describe our model of MANET communications, which – though a simplification of reality – captures the most important aspect of tactical radio communications: signal and interference propagation over rough terrain. We generate realistic but unclassified datasets based on U.S. Marine Corps combat scenarios. We describe and provide evidence of the computational challenges of the cumulative-interference CAP problem. For this reason, the vast majority of research considers only pairwise interference constraints. We demonstrate that for our tactical MANET radios, pairwise interference constraints insufficiently capture interference that can inhibit the ability of a radio to communicate.

We describe an integer programming formulation of the cumulative interference MO-CAP, and develop a method for solving realistic, full-size instances the problem to global or near global optimality in a reasonable amount of time using lazy constraints and maximum clique constraints. We also use constraint programming to quickly obtain lower bounds to the problem.

We develop and compare three methods of solving the cumulative interference MI-CAP. We also describe a method for bounding the goodness of MI-CAP solutions. We find that our CP approach obtains certifiably-good solutions in less time than our IP technique or clustering heuristic. We describe the operational impact of excessive interference in terms of network availability, a metric that may more closely speak to the demands of tactical operations than the specific dBm or watts of excessive interference.

We develop a globally-optimal method for quickly solving the MC-CAP-T, minimizing the number of required channel changes over time using the solutions from either the MO-CAP or MI-CAP. Our MC-CAP-T method can be implemented without commercial solvers, and can easily be re-run to account for changes in the future operating state.

7.2 Future Research

Current methods of assigning channels to support tactical radios during military operations are quite rudimentary. An obvious next step would be to demonstrate the utility of our methods in field experiments, and then to incorporate the methods into spectrum planning software, such as SPEED or Spectrum XXI.

The MI-CAP remains a notoriously difficult problem. Future research is needed on bounding the amount of received excessive interference to help provide a better understanding of the goodness of obtained solutions. Our work shows that for the instances tested, pairwise interference alone captures a significant (but not total) portion of interference resulting in network unavailability. Our use of lazy constraints and the maximum clique constraint to consider cumulative interference maximizes the efficient use of available spectrum. More study should reveal if this is true in most real-world situations.

We find great utility in using constraint programming to bound the goodness of solutions to the MO-CAP, and to obtain good solutions to the MI-CAP. However, there are few general rules when formulating CP problems, and we often result to trial-and-error in order to find formulations that obtain useful results. Future research is needed on proving and

articulating general strategies for CP (see, e.g., Refalo (2004), Rossi et al. (2006), Hooker (2011)).

One could develop and apply variations of our methods to the other CAPs described in Aardal et al. (2007), including maximum service, minimum blocking, and minimum span problems. Also, our work assumes that spectrum – though scarce – will not be contested, yet military operations are likely to take place within actively-contested EM environments. Future research could consider the allocation of spectrum in the presence of a determined adversary (see, e.g., London (2015), Wu et al. (2012), El-Bardan et al. (2014)).

Appendix A: Data Tables

This appendix provides various data tables of the results of our methods.

Table A.1: Number of available channels during the MI-CAP analysis, for each time step and channel availability level, where 100% channel availability is the best known solution to the MO-CAP.

		Channel Availability				
Time Step	**Best known MO-CAP Solution**	**90%**	**80%**	**70%**	**60%**	**50%**
1	46	41	36	32	27	23
2	37	33	29	25	22	18
3	35	31	28	24	21	17
4	34	30	27	23	20	17
5	33	29	26	23	19	16
6	36	32	28	25	21	18
7	37	33	29	25	22	18
8	31	27	24	21	18	15
9	32	28	25	22	19	16
10	34	30	27	23	20	17
11	33	29	26	23	19	16
12	36	32	28	25	21	18
13	32	28	25	22	19	16
14	31	27	24	21	18	15
15	38	34	30	26	22	19
16	36	32	28	25	21	18
17	37	33	29	25	22	18
18	31	27	24	21	18	15
19	30	27	24	21	18	15
20	37	33	29	25	22	18

Table A.2: Total number of pairwise constraint violations using the MI-CAP clustering formulation, with varying numbers of available channels and 200 iterations.

	Pairwise Violations, by Channel Availability					
Time Step	**100%**	**90%**	**80%**	**70%**	**60%**	**50%**
1	111	216	188	154	210	419
2	302	195	233	364	255	422
3	162	190	170	215	321	354
4	259	272	222	392	490	409
5	386	404	443	568	506	1057
6	196	315	211	240	272	308
7	161	207	175	363	281	462
8	228	227	252	376	409	343
9	188	215	259	306	449	347
10	177	158	364	274	301	468
11	214	283	185	309	305	289
12	268	206	245	232	554	580
13	183	195	257	240	391	393
14	150	358	272	397	378	473
15	146	255	263	202	268	336
16	142	316	272	241	305	345
17	173	194	344	421	302	344
18	516	279	334	354	595	477
19	276	272	254	354	379	491
20	181	210	255	542	378	463
Average	**221.0**	**248.4**	**259.9**	**327.2**	**367.5**	**439.0**

Table A.3: Total number of radios (out of 1887) receiving excessive interference using the MI-CAP clustering formulation, with varying numbers of available channels and 200 iterations.

	Radios Receiving Excessive Interference, by Channel Availability					
Time Step	**100%**	**90%**	**80%**	**70%**	**60%**	**50%**
1	123	141	177	174	207	229
2	129	136	158	153	166	162
3	125	139	140	145	163	182
4	150	146	144	148	178	160
5	149	170	165	189	187	213
6	123	144	154	149	175	162
7	132	145	146	160	169	192
8	152	158	170	165	188	176
9	130	150	146	170	161	163
10	110	117	133	137	149	155
11	125	146	156	158	169	176
12	109	122	138	143	176	176
13	131	126	137	140	156	164
14	111	126	138	145	149	137
15	120	121	135	128	147	158
16	85	114	119	130	131	142
17	114	118	135	138	165	159
18	131	113	135	137	177	163
19	146	135	145	151	169	183
20	124	118	124	133	151	165
Average	**125.95**	**134.25**	**144.75**	**149.65**	**166.65**	**170.85**

Table A.4: Total excessive interference (in dBm) using the MI-CAP clustering formulation, with varying numbers of available channels and 200 iterations.

	Total Excessive Interference (dBm), by Channel Availability					
Time Step	**100%**	**90%**	**80%**	**70%**	**60%**	**50%**
1	17.70	19.07	21.43	22.88	23.96	24.86
2	-33.30	-30.68	-29.82	-31.02	-28.93	-26.09
3	-27.96	-27.91	-28.00	-28.09	-25.81	-23.61
4	-24.44	-31.61	-27.57	-28.42	-28.56	-23.80
5	-30.51	-26.02	-28.41	-22.15	-24.84	-23.68
6	-24.09	-28.24	-28.01	-32.73	-18.91	-21.89
7	-34.80	-34.37	-33.54	-31.73	-30.51	-26.28
8	-30.31	-33.29	-30.10	-31.44	-26.86	-30.93
9	-33.98	-31.70	-28.07	-27.93	-28.28	-28.73
10	-36.04	-37.20	-34.02	-30.95	-30.70	-32.14
11	-33.50	-28.92	-27.72	-31.29	-27.26	-26.07
12	-34.15	-31.96	-31.23	-32.11	-29.19	-30.46
13	-34.27	-32.36	-33.52	-32.23	-30.81	-36.50
14	-38.29	-37.06	-33.24	-32.94	-32.99	-32.90
15	-39.07	-37.14	-33.14	-33.26	-31.71	-30.08
16	-44.36	-34.22	-37.24	-30.20	-33.13	-28.95
17	-36.06	-41.55	-37.53	-35.25	-37.55	-31.81
18	-24.37	-25.25	-26.26	-25.50	-24.84	-24.16
19	-40.02	-39.93	-32.71	-35.77	-29.96	-27.68
20	-33.07	-40.01	-33.42	-31.05	-34.88	-29.80
Average	**-30.75**	**-30.52**	**-28.61**	**-28.06**	**-26.59**	**-25.53**

Table A.5: Lower bound on the number of pairwise interference constraint violations in the MI-CAP Restricted IP and CP formulations, for each time step and with varying channel availability.

	LB of Pairwise Violations (T), by Channel Availability					
Time Step	**100%**	**90%**	**80%**	**70%**	**60%**	**50%**
1	0	5	10	14	19	23
2	0	4	8	12	15	20
3	0	3	6	10	13	19
4	0	3	6	10	13	16
5	0	4	7	10	14	18
6	0	3	7	10	14	17
7	0	4	8	12	15	20
8	0	3	6	10	16	22
9	0	4	7	10	13	16
10	0	4	7	11	14	17
11	0	4	7	10	14	18
12	0	3	7	10	14	17
13	0	3	6	9	12	15
14	0	3	6	9	12	15
15	0	4	8	12	16	19
16	0	3	7	10	14	17
17	0	3	7	11	14	18
18	0	4	7	10	13	17
19	0	3	6	9	12	15
20	0	4	8	12	15	20
Average	**0**	**3.55**	**7.05**	**10.55**	**14.1**	**17.95**

Table A.6: Lower bound on the total amount of received excessive interference in the MICAP Restricted IP and CP formulations, for each time step and with varying channel availability.

	LB of Received Excessive Interference, by Channel Availability					
Time Step	**100%**	**90%**	**80%**	**70%**	**60%**	**50%**
1	-	-125.34	-117.89	-114.89	-111.80	-109.38
2	-	-138.61	-131.28	-127.72	-125.96	-123.40
3	-	-145.69	-136.84	-131.02	-128.13	-125.33
4	-	-152.12	-145.38	-139.96	-136.02	-132.54
5	-	-149.27	-143.41	-139.83	-131.97	-128.37
6	-	-156.12	-148.01	-144.92	-142.09	-140.07
7	-	-157.55	-151.62	-147.75	-144.96	-140.75
8	-	-157.77	-152.25	-147.71	-138.23	-132.23
9	-	-146.92	-142.32	-138.97	-136.22	-134.14
10	-	-154.34	-147.80	-141.74	-139.08	-136.99
11	-	-154.35	-149.10	-140.92	-136.07	-132.29
12	-	-150.84	-143.08	-138.54	-134.16	-131.54
13	-	-148.91	-142.31	-138.39	-135.49	-132.85
14	-	-145.02	-140.16	-136.67	-133.68	-131.45
15	-	-147.28	-141.95	-137.52	-133.62	-129.89
16	-	-148.20	-141.95	-138.83	-135.55	-133.53
17	-	-157.88	-151.33	-147.88	-145.87	-143.25
18	-	-143.74	-139.00	-136.28	-133.39	-129.68
19	-	-145.20	-140.59	-136.77	-133.80	-131.38
20	-	-144.83	-135.47	-131.97	-129.91	-127.07
Average	**-**	**-148.5**	**-142.09**	**-137.91**	**-134.3**	**-131.31**

Table A.7: Lower bound on the number of radios receiving excessive interference in the MI-CAP Restricted IP and CP formulations with weighted penalties, for each time step and with varying channel availability.

	LB of Radios Receiving Excessive Interference, by Channel Availability					
Time Step	**100%**	**90%**	**80%**	**70%**	**60%**	**50%**
1	0	5	10	16	25	33
2	0	4	8	12	15	24
3	0	3	6	10	13	19
4	0	3	6	10	14	20
5	0	4	7	10	15	22
6	0	3	7	10	15	20
7	0	4	8	12	15	22
8	0	3	6	10	16	23
9	0	4	7	10	14	18
10	0	4	7	11	14	17
11	0	4	7	10	16	22
12	0	3	7	10	16	21
13	0	3	6	9	12	17
14	0	3	6	9	12	17
15	0	4	8	12	17	21
16	0	3	7	10	15	19
17	0	3	7	11	14	18
18	0	4	7	10	13	20
19	0	3	6	9	12	16
20	0	4	8	12	15	21
Average	**0**	**3.55**	**7.05**	**10.65**	**14.9**	**20.5**

Table A.8: Total number of radios (out of 1887) receiving excessive interference using the MI-CAP Restricted IP formulation, with varying numbers of available channel, 500 second CPLEX runtimes, and unit penalties.

	Radios Receiving Excessive Interference, by Channel Availability					
Time Step	**100%**	**90%**	**80%**	**70%**	**60%**	**50%**
1	104	170	156	224	255	284
2	106	117	134	154	135	172
3	123	137	131	129	136	159
4	135	89	139	137	167	172
5	137	133	148	133	175	144
6	127	125	147	146	154	181
7	75	120	142	119	174	169
8	83	126	113	167	153	137
9	83	91	131	88	163	149
10	112	121	124	106	152	153
11	127	132	131	132	124	177
12	100	109	124	109	146	159
13	88	125	124	121	137	166
14	103	128	119	143	134	161
15	102	112	117	122	140	117
16	105	90	134	147	131	137
17	100	105	137	118	153	167
18	73	108	118	91	131	157
19	120	124	100	104	142	153
20	78	107	103	101	132	136
Average	**104.1**	**118.5**	**128.6**	**129.6**	**151.7**	**162.5**

Table A.9: Total excessive interference (in dBm) using the MI-CAP Restricted IP formulation, with varying numbers of available channels, 500 second CPLEX runtimes, and unit penalties.

	Total Excessive Interference (dBm), by Channel Availability					
Time Step	**100%**	**90%**	**80%**	**70%**	**60%**	**50%**
1	15.96	23.52	16.71	25.52	26.94	29.21
2	-32.70	-25.59	-13.83	-17.19	-30.26	-21.89
3	-31.01	-20.44	-13.68	-21.78	-16.52	-24.32
4	-23.81	-24.79	-28.65	-12.49	-16.92	-22.87
5	-32.58	-32.55	-21.47	-22.79	-24.85	-27.32
6	-26.85	-34.55	-15.59	-23.09	-26.31	-14.73
7	-40.78	-31.71	-34.73	-28.05	-7.68	-26.48
8	-15.82	-27.74	-30.02	-10.52	-25.84	-27.51
9	-20.03	-29.14	-25.89	-33.02	-23.69	-26.91
10	-18.70	-19.26	-29.60	-34.48	-17.60	-15.42
11	-8.83	-32.73	-34.98	-11.76	-18.91	-11.56
12	-30.05	-37.30	-38.80	-16.24	-28.87	-35.42
13	-36.78	-28.20	-14.16	-31.92	-30.85	-21.73
14	-22.08	-21.94	-26.51	-20.15	-33.09	-14.65
15	-23.43	-27.90	-21.20	-34.68	-18.44	-33.36
16	-30.25	-33.79	-3.13	-23.39	-33.73	-28.78
17	-43.35	-40.37	-26.70	-26.01	-3.58	-10.12
18	-24.57	-35.81	-30.02	-24.22	-32.82	-12.98
19	-26.55	-24.00	-30.05	-27.53	-29.40	-17.23
20	-48.15	-26.15	-15.48	-10.69	-31.99	-30.77
Average	**-26.02**	**-26.52**	**-21.89**	**-20.22**	**-21.22**	**-19.74**

Table A.10: Total number of pairwise constraint violations using the MI-CAP Restricted IP formulation, with varying numbers of available channels, 500 second CPLEX runtimes, and weighted penalties.

	Pairwise Violations, by Channel Availability					
Time Step	**100%**	**90%**	**80%**	**70%**	**60%**	**50%**
1	74	72	95	109	133	352
2	73	85	98	202	243	285
3	86	83	111	228	209	283
4	86	102	118	223	310	239
5	84	96	117	214	362	163
6	72	96	110	214	304	285
7	65	84	105	180	277	281
8	80	106	216	272	266	270
9	77	88	195	235	228	167
10	76	87	106	256	262	171
11	74	89	107	216	259	291
12	63	77	85	210	258	265
13	84	102	179	199	112	194
14	79	103	204	392	198	358
15	67	81	97	107	185	119
16	71	93	107	195	231	277
17	69	84	93	224	329	183
18	81	86	197	216	291	287
19	87	96	206	230	167	168
20	73	79	103	266	169	235
Average	**76.1**	**89.5**	**132.5**	**219.4**	**239.7**	**243.7**

Table A.11: Total number of radios (out of 1887) receiving excessive interference using the MI-CAP Restricted IP formulation, with varying numbers of available channels, 500 second CPLEX runtimes, and weighted penalties.

	Radios Receiving Excessive Interference, by Channel Availability					
Time Step	**100%**	**90%**	**80%**	**70%**	**60%**	**50%**
1	123	170	156	224	255	266
2	106	117	134	139	143	162
3	123	137	131	155	131	163
4	135	122	139	137	155	157
5	137	133	148	144	130	134
6	127	125	147	144	157	153
7	75	120	142	118	175	150
8	117	126	115	144	128	156
9	77	142	104	114	148	160
10	112	121	124	110	137	155
11	67	132	131	125	156	166
12	100	109	124	116	127	148
13	90	125	120	137	130	151
14	103	128	116	121	133	152
15	102	112	117	122	130	96
16	105	105	134	108	116	146
17	100	105	137	115	131	134
18	105	108	118	130	118	137
19	73	124	91	118	109	161
20	107	107	91	116	134	152
Average	**104.2**	**123.4**	**126.0**	**131.9**	**142.2**	**155.0**

Table A.12: Total excessive interference (in dBm) using the MI-CAP Restricted IP formulation, with varying numbers of available channels, 500 second CPLEX runtimes, and weighted penalties.

	Total Excessive Interference (dBm), by Channel Availability					
Time Step	**100%**	**90%**	**80%**	**70%**	**60%**	**50%**
1	21.19	23.52	16.71	25.52	26.94	25.94
2	-32.70	-25.59	-13.83	-16.77	-19.91	-22.11
3	-31.01	-20.44	-13.68	-20.60	-31.36	-24.52
4	-23.81	-26.18	-28.65	-28.51	-26.86	-17.47
5	-32.58	-32.55	-21.47	-24.04	-23.69	-30.40
6	-26.85	-34.55	-15.59	-20.26	-24.30	-13.22
7	-40.78	-31.71	-34.73	-23.78	-7.45	-25.54
8	-37.33	-27.74	-29.17	-8.25	-24.74	-25.40
9	-19.94	-23.49	-22.33	-28.17	-27.35	-5.35
10	-18.70	-19.26	-29.60	-34.17	-33.82	-17.46
11	-37.47	-32.73	-34.98	-19.52	-30.37	-28.56
12	-30.05	-37.30	-38.80	-16.26	-26.96	-31.45
13	-36.16	-28.20	-43.27	-28.78	-28.93	-25.79
14	-22.08	-21.94	-15.54	-18.63	-17.26	-25.34
15	-23.43	-27.90	-21.20	-34.68	-22.65	-46.74
16	-30.25	-14.33	-3.13	-32.54	-24.50	-31.00
17	-43.35	-40.37	-26.70	-39.00	-27.31	-31.81
18	-31.84	-35.81	-26.00	-39.85	-38.85	-22.42
19	-31.44	-24.00	-29.37	-20.49	-21.72	-28.28
20	-28.68	-26.15	-39.08	-19.34	-30.55	-19.35
Average	**-27.86**	**-25.34**	**-23.52**	**-22.41**	**-23.08**	**-22.31**

Table A.13: Total number of radios (out of 1887) receiving excessive interference using the MI-CAP Restricted IP formulation, with varying numbers of available channels, 6000 second CPLEX runtimes, and weighted penalties.

	Radios Receiving Excessive Interference, by Channel Availability					
Time Step	**100%**	**90%**	**80%**	**70%**	**60%**	**50%**
1	19	43	48	80	96	128
2	22	24	47	80	78	108
3	22	38	53	76	89	136
4	22	42	60	79	103	115
5	59	51	55	68	101	118
6	28	47	62	68	83	102
7	21	30	48	76	89	114
8	16	28	37	47	78	101
9	12	23	37	51	82	91
10	7	22	34	51	68	93
11	8	16	28	54	74	89
12	7	16	28	41	59	84
13	10	22	31	52	81	110
14	8	24	34	57	67	89
15	12	27	34	64	75	84
16	17	29	47	70	82	90
17	15	28	59	66	77	90
18	8	24	33	46	62	85
19	10	24	33	52	70	83
20	17	30	48	62	70	86
Average	**104.2**	**123.4**	**126.0**	**131.9**	**142.2**	**155.0**

Table A.14: Total excessive interference (in dBm) using the MI-CAP Restricted IP formulation, with varying numbers of available channels, 6000 second CPLEX runtimes, and weighted penalties.

	Total Excessive Interference (dBm), by Channel Availability					
Time Step	**100%**	**90%**	**80%**	**70%**	**60%**	**50%**
1	-70.37	-63.00	-56.25	-35.67	-33.68	12.00
2	-80.34	-60.18	-46.39	-29.38	-36.81	-37.89
3	-86.37	-59.18	-70.96	-32.47	-42.97	-25.57
4	-58.58	-54.43	-29.07	-24.08	-31.33	-35.18
5	-37.99	-54.94	-55.43	-25.98	-36.78	-35.24
6	-58.32	-56.11	-38.80	-25.46	-37.65	-39.28
7	-79.65	-80.34	-48.71	-39.66	-35.99	-36.12
8	-74.35	-73.97	-53.17	-17.07	-40.65	-29.18
9	-82.54	-38.36	-44.54	-40.23	-23.46	-27.62
10	-74.18	-71.08	-75.65	-43.86	-44.90	-43.38
11	-92.41	-85.63	-67.65	-27.52	-54.17	-40.35
12	-78.07	-69.95	-43.46	-43.32	-56.43	-51.53
13	-85.15	-50.38	-50.99	-25.83	-45.53	-38.12
14	-74.30	-55.37	-53.80	-28.24	-53.02	-48.49
15	-84.52	-76.53	-77.64	-39.52	-42.22	-48.98
16	-62.44	-73.09	-65.95	-37.33	-53.05	-55.63
17	-92.52	-78.63	-56.37	-41.39	-50.83	-50.27
18	-88.79	-78.17	-63.84	-28.84	-50.30	-54.91
19	-91.43	-46.64	-70.57	-32.20	-37.58	-38.69
20	-87.00	-80.67	-60.28	-21.37	-58.04	-36.10
Average	**-27.86**	**-25.34**	**-23.52**	**-22.41**	**-23.08**	**-22.31**

Table A.15: Total number of radios receiving excessive interference using the MI-CAP CP formulation, with varying numbers of available channels, 500 second CP runtimes, and unit penalties.

	Radios Receiving Excessive Interference, by Channel Availability					
Time Step	**100%**	**90%**	**80%**	**70%**	**60%**	**50%**
1	7	52	50	96	140	215
2	4	23	44	54	80	113
3	13	26	44	68	97	134
4	9	18	39	57	79	102
5	5	26	41	62	108	134
6	7	27	46	65	96	109
7	9	27	38	64	86	127
8	7	19	44	76	91	116
9	7	25	38	47	79	107
10	8	19	33	60	78	110
11	10	24	39	50	73	108
12	6	19	43	49	77	86
13	12	27	38	55	80	100
14	10	28	38	54	75	94
15	6	26	45	50	72	92
16	8	20	38	57	74	102
17	7	18	31	71	87	101
18	9	18	30	53	81	82
19	12	18	32	47	71	99
20	7	20	36	57	73	99
Average	**8.15**	**24**	**39.35**	**59.6**	**84.85**	**111.5**

Table A.16: Total excessive interference (in dBm) using the MI-CAP CP formulation, with varying numbers of available channels, 500 second CP runtimes, and unit penalties.

	Total Excessive Interference (dBm), by Channel Availability					
Time Step	**100%**	**90%**	**80%**	**70%**	**60%**	**50%**
1	-66.14	17.46	6.95	17.53	21.11	22.13
2	-61.47	-66.05	-39.93	-57.30	-25.06	-36.53
3	-85.82	-78.37	-44.65	-32.59	-20.78	-20.83
4	-73.98	-48.82	-62.94	-48.66	-38.46	-31.10
5	-53.66	-53.50	-50.87	-28.78	-32.89	-33.13
6	-80.31	-44.84	-42.88	-40.03	-37.03	-42.64
7	-80.50	-55.29	-53.54	-37.67	-34.62	-28.38
8	-88.26	-94.32	-35.48	-35.02	-28.62	-11.68
9	-84.93	-55.87	-59.01	-65.91	-39.59	-36.94
10	-88.64	-48.95	-78.21	-44.59	-37.24	-51.67
11	-83.68	-78.45	-42.52	-36.84	-41.00	-36.83
12	-83.77	-63.72	-16.09	-66.93	-41.69	-42.28
13	-62.60	-51.68	-60.21	-49.08	-38.33	-36.79
14	-67.34	-69.14	-68.73	-66.94	-60.20	-22.10
15	-75.03	-52.59	-45.31	-42.65	-38.91	-39.05
16	-60.91	-63.42	-23.60	-40.29	-55.33	-46.14
17	-81.47	-80.67	-76.84	-32.07	-29.98	-32.76
18	-72.39	-60.06	-48.56	-46.84	-44.44	-44.27
19	-80.87	-73.09	-57.87	-48.26	-23.08	-22.87
20	-63.54	-60.80	-79.72	-24.97	-51.70	-28.47
Average	**-74.77**	**-59.11**	**-49.00**	**-41.39**	**-34.89**	**-31.12**

Table A.17: Total number of radios receiving excessive interference using the MI-CAP CP formulation, with varying numbers of available channels, 500 second CP runtimes, and weighted penalties.

	Radios Receiving Excessive Interference, by Channel Availability					
Time Step	**100%**	**90%**	**80%**	**70%**	**60%**	**50%**
1	7	16	31	41	78	103
2	4	13	18	41	53	86
3	13	14	26	42	66	95
4	9	16	23	45	66	87
5	5	14	30	53	70	96
6	7	13	31	46	66	88
7	9	11	20	45	57	86
8	7	16	30	40	65	91
9	7	17	19	41	56	82
10	8	10	20	40	60	79
11	10	11	24	35	58	81
12	6	10	19	25	47	67
13	12	15	27	41	62	87
14	10	15	21	34	49	70
15	6	12	17	32	54	69
16	8	14	21	38	54	76
17	7	8	19	35	52	74
18	9	16	22	33	47	73
19	12	9	19	33	55	73
20	7	15	21	38	60	71
Average	**8.15**	**13.25**	**22.9**	**38.9**	**58.75**	**81.7**

Table A.18: Total excessive interference (in dBm) using the MI-CAP CP formulation, with varying numbers of available channels, 500 second CP runtimes, and weighted penalties.

	Total Excessive Interference (dBm), by Channel Availability					
Time Step	**100%**	**90%**	**80%**	**70%**	**60%**	**50%**
1	-66.14	-64.89	-58.97	-40.04	-53.95	-41.57
2	-61.47	-62.24	-67.88	-55.84	-67.01	-44.02
3	-85.82	-101.39	-65.22	-52.46	-32.21	-41.01
4	-73.98	-88.72	-53.70	-63.25	-40.87	-38.30
5	-53.66	-88.68	-53.67	-45.53	-45.49	-41.06
6	-80.31	-76.18	-44.36	-44.67	-36.78	-22.78
7	-80.50	-90.58	-80.81	-45.43	-55.54	-41.81
8	-88.26	-74.45	-75.00	-46.13	-54.99	-39.07
9	-84.93	-62.51	-66.38	-60.48	-53.28	-48.49
10	-88.64	-74.80	-91.14	-61.77	-63.92	-41.04
11	-83.68	-84.67	-76.43	-67.39	-61.64	-46.80
12	-83.77	-93.73	-67.35	-66.89	-63.11	-53.35
13	-62.60	-66.77	-66.88	-39.47	-39.26	-39.17
14	-67.34	-71.40	-66.94	-70.03	-55.39	-41.84
15	-75.03	-76.33	-70.17	-76.08	-54.11	-55.80
16	-60.91	-64.11	-61.19	-89.79	-63.50	-55.55
17	-81.47	-80.76	-65.60	-75.80	-66.47	-49.15
18	-72.39	-71.98	-71.97	-69.02	-62.37	-41.03
19	-80.87	-85.31	-78.17	-69.25	-23.09	-51.90
20	-63.54	-62.74	-59.39	-59.36	-59.49	-54.19
Average	**-74.77**	**-77.11**	**-67.06**	**-59.93**	**-52.62**	**-44.40**

Appendix B: MO-CAP RSF Code

This appendix provides partial computer code to solve the MO-CAP Restricted Standard Formulation using Python, Pyomo, and CPLEX.

```
# Import packages
import globalVars as globalVars # Contains global variables
import Utilities as util # Basic utilities, like reading in input files and pre-
    processing interference values
import cplexUtilities as cpUtil # CPLEX utilities, like adding pairwise constraints
    and checking if units can be on same channel
import numpy as np
import mpmath as mp # Import mpmath library, for abitrary-precision floating-point
    numbers
mp.dps = globalVars.MP_PRECISION # Set precision of mpmath floats to given number of
    decimal places
import math
import copy # For copying arrays
import itertools as itertools # For iterating over combinations of units in
    LazyConstraints callback
import cplex
from cplex.callbacks import LazyConstraintCallback
from pyomo.environ import *
from pyomo.opt import * #SolverFactory, SolverStatus, TerminationCondition # Needed to
     execute `opt = SolverFactory(`cplex')' and run solver
from os import path # For checking if a file exists
import pickle # For saving pickled array
import time # For measuring elapsed processing time

# Determines if the given units can share a channel (checks each way)
def canTheseUnitsShareChannelAssignment(unitList):
    for unitA in unitList: # Check each unit against all the others in unitList
        otherUnitList = copy.deepcopy(unitList)
        otherUnitList.remove(unitA) # Create a list of all the other units in
            unitList
        for i in range(globalVars.unitPointer[unitA],globalVars.unitPointer[unitA]+
            globalVars.numberSubUnits[unitA]): # For each radio in unitA
            if any(globalVars.maxInterference[i] < np.sum(globalVars.
                interferenceMatrix[i,otherUnitList,0])): return False # Over all
                other units, on channel 0
    return True

# Find cliques (among the pairwise constraints) and add as constraints to CPLEX
    problem (if needed)
def addCliqueConstraints(pickledPairwiseConstraintsFileName, pyomoProblem = None):
    numberCliqueConstraintsAdded = 0
    if not pyomoProblem is None: numberUnits = len(pyomoProblem.u)
    else: numberUnits = globalVars.numberUnits
```

```
    canUnitsShareChannelAssignmentArray = util.calcUnitsShareChannelAssignmentArray(
        pickledPairwiseConstraintsFileName, globalVars.numberUnits) # Calculate and
        return the canUnitsShareChannelAssignmentArray

    # Create NetworkX graph object from pairwise constraints
    import networkx as netX
    theGraph = netX.Graph() # Create default graph
    theGraph.add_nodes_from(range(0,globalVars.numberUnits-1)) # Add all units as
        nodes (-1 because of how NetworkX creates nodes)
    for i in range(globalVars.numberUnits):
        for j in range(globalVars.numberUnits):
            if i < j and not canUnitsShareChannelAssignmentArray[i,j]: theGraph.
                add_edge(i,j)

    # Find maximal cliques (largest clique, for each node). The biggest maximal
        clique is the maximum clique
    maximalCliqueGenerator = netX.find_cliques(theGraph) # A generator of all of the
        maximal cliques
    maximalCliqueList = [] # Copy to list to work with it
    for i in maximalCliqueGenerator: maximalCliqueList.append(i)
    [maximumCliqueLength, maximumClique] = max(enumerate(maximalCliqueList), key =
        lambda tup: len(tup[1])) # Lambda function to get maximum clique (biggest
        maximal)

    # Add maximum clique to the pyomoProblem, if one was sent as argument
    if not pyomoProblem is None:
        for c in pyomoProblem.c:
            theRule = pyomoProblem.X['u' + str(maximumClique[0]),c]
            for i in range(1, len(maximumClique)): theRule = theRule + pyomoProblem.X
                ['u' + str(maximumClique[i]),c]
            pyomoProblem.Cuts.add(theRule <= 1)

    print 'Maximum clique (size ' + str(len(maximumClique)) + '): ', sorted(
        maximumClique)

    return numberCliqueConstraintsAdded, maximumClique

# Can't pass arguments to CPLEX callbacks, so need to use these globals
unitList = [] # List of the CPLEX variables for unit, in sorted order
unitDict = {} # Looks up a unit by unit number and channel, and returns the number of
     associated CPLEX variable
numberFixedUnits = 0 # Number of units whose associated variables have been fixed (so
     CPLEX can't change them)

# Generage utilization constraints: A channel is counted if it gets used
def cap_utilizationConstraint_rule(model, u, c):
    return model.X[u,c] <= model.Y[c]

# Generage multiplicity constraints: Each unit must have exactly one channel assigned
def cap_multiplicityConstraint_rule(model, u):
    return sum(model.X[u,c] for c in model.c) == 1
```

```
# Generage a constraint to provide a lower bound to the objective value, in hopes of
    speeding up convergence
def cap_objectiveValueLBConstraint_rule(model):
    return sum(model.Y[c] for c in model.c) >= 35

# Objective rule definition (doesn't work when you declare within the Objective
    function)
def cap_objective_rule(model):
    return sum(model.Y[c] for c in model.c)

# Calculate and return the current channelAssignment for the given CPLEX object,
    depending on calling routine
def calcCurrentCPLEXChannelAssignment(solver, isPyomoInterface):
    currentChannelAssignment = np.empty(globalVars.numberUnits, dtype=np.int32) #
        CPLEX's current channel assignment solution, by unit
    counter = 0
    nonSharerCounter = 0
    if isPyomoInterface:
        unitStart = globalVars.NUMBER_AVAILABLE_CHANNELS # Advance to the X_u values,
             which follow after the Y_c values
        global numberFixedUnits
        counter = unitStart - numberFixedUnits
    for i in range(0, globalVars.numberUnits):
        for c in range(0, globalVars.NUMBER_AVAILABLE_CHANNELS):
            if (not isPyomoInterface and round(solver.get_values(str('X(u' + str(i) +
                 '_' + str(c) + ')'))) == 1) or (isPyomoInterface and round(solver.
                get_values(counter)) == 1):
                currentChannelAssignment[i] = c
                counter = counter + (globalVars.NUMBER_AVAILABLE_CHANNELS-c)
                break
            counter += 1
    return currentChannelAssignment

# Pyomo and CPLEX lazyConstraintCallbacks go here to check feasibility, add lazy
    constraints (if needed), and (if solving MO-CAP-T), add solution to solutionPool
    and constrain solution from being used again.
def addLazyConstraints(solver, model = None, isPyomoInterface = True):
    # Read in current solution
    print 'Lazy constraint callback.  Reading in current CPLEX solution...',
    currentChannelAssignment = np.empty(globalVars.numberUnits, dtype=np.int32) #
        CPLEX current channel assignment solution, by unit
    currentChannelAssignment = calcCurrentCPLEXChannelAssignment(solver,
        isPyomoInterface) # Get current channelAssignment
    currentObjectiveValue = round(solver.get_objective_value())
    if len(np.unique(currentChannelAssignment)) <> currentObjectiveValue:
        print 'There is a problem in calculating the currentChannelAssignment in the
            lazy constraint callback.'
    print 'Done.'

    # Check if current CPLEX solution has any radios receiving excessive interference
    print 'Calculating total received interference at each radio...',
```

```
numberReceivedExcessiveInterference = 0 # Number of radios receiving excessive
    interference
receivedInterference = np.zeros(globalVars.numberRadios,dtype=mp.mpf) # Total
    received interference, by radio
numberReceivedExcessiveInterference, receivedInterference = util.
    calcReceivedInterference(currentChannelAssignment)
print 'Done.'

# No violations.  If running MO-CAP-T, then check to see if solution needs to be
    added to solutionPool
if numberReceivedExcessiveInterference == 0:
    print 'No cumulative interference violations exist; current solution is
        feasible.  Channels required: ' + str(currentObjectiveValue) + '.
        Current solver time: ' + str(time.time() - globalVars.globalStartTime)
    globalVars.isCPLEXSolutionFeasible = True

# Violations exist.  Find them and add packing constraints preventing them
elif numberReceivedExcessiveInterference > 0:
    print 'Cumulative interference constraint violations exist.  Current solution
         requires ' + str(currentObjectiveValue) + ' channels.  Finding
        violations...'
    constrainedAssignmentsByChannel = [[] for i in range(0,globalVars.
        NUMBER_AVAILABLE_CHANNELS)] # A list of lists, where the first index is
        channel and the second is a list of units that can't all be assigned that
         channel
    counter = 0

    # Loop through all radios to find violations, and add the unit to list
    for i in range(0, globalVars.numberUnits):
        for j in range(0, globalVars.numberSubUnits[i]):
            if receivedInterference[counter] > globalVars.maxInterference[counter
                ,currentChannelAssignment[i]]:
                constrainedAssignmentsByChannel[currentChannelAssignment[i]].
                    append(i) # Add this unit to the list of units that can't be
                    assigned this channel
                counter = counter + (globalVars.numberSubUnits[i] - j)
                break
            counter += 1

    # Loop through all channels and add to constrainedAssignmentsByChannel all
        units on a violated channel that haven't yet been added
    for c in range(0, globalVars.NUMBER_AVAILABLE_CHANNELS):
        if len(constrainedAssignmentsByChannel[c]) > 0: # If there are violations
             on this channel, add all units not already added (checked using the
            .count() method)
            for i in range(0, globalVars.numberUnits):
                if currentChannelAssignment[i] == c and
                    constrainedAssignmentsByChannel[c].count(i) == 0:
                    constrainedAssignmentsByChannel[c].append(i)

    # Add original packing constraints (i.e., allow |S|-1 units in the subset S
        of units, on the assigned (and violated) channel)
```

```
print 'Adding triplet and higher-order packing constraints as lazy
    constraints...'
for c in range(0, globalVars.NUMBER_AVAILABLE_CHANNELS):
    if len(constrainedAssignmentsByChannel[c]) > 0: # If there are violations
         on this channel
        restrictedUnitList = [] # List of constraints of units on this c that
             can't be assigned together (using actual unitNumber), to be
            converted into CPLEX index number
        LHS = [] # Combined list of variableList and coefficientList

        if len(constrainedAssignmentsByChannel[c]) == 3: # If exactly three,
            add all three
            theList = []
            for unit in constrainedAssignmentsByChannel[c]: theList.append(
                unit)
            restrictedUnitList.append(theList)
        elif len(constrainedAssignmentsByChannel[c]) > 3: # If more than
            three, check for disallowed tuples among the four or more units
            tupleSize = 3
            while tupleSize <= globalVars.MAX_INTERFERENCE_TUPLE_SIZE and len
                (restrictedUnitList) == 0: # Check until you hit tupleSize
                limit, or a disallowed tuple is found
                listOfUnitCombinations = itertools.combinations(
                    constrainedAssignmentsByChannel[c], tupleSize) # Get all
                    combinations of tupleSize among units (where order doesn'
                    t matter)
                for unit in listOfUnitCombinations: # Check if a combination
                    isn't allowed; if so, add to list
                    theList = list(unit) # Convert tuple to list
                    if not cpUtil.canTheseUnitsShareChannelAssignment(theList
                        ): # If they can't share
                        restrictedUnitList.append(theList) # Add each unit to
                             the associated constraint
                tupleSize += 1 # Increase tupleSize for next iteration
        else:
            print 'Problem: There are not enough units assigned to this
                channel for there to be a violation.'

        # Loop over all units for each constraint on this channel and add to
            packing constraint
        for constraint in restrictedUnitList:
            variableList = [] # CPLEX indices (not the actual names/numbers)
            coefficientList = [] # Always one, for each unit
            actualUnitNames = [] # For printing the unit names
            for unit in constraint:
                if isPyomoInterface: variableList.append('x' + str(unitDict[
                    int(unit),c]))
                else: variableList.append(str('X(u' + str(unit) + '_' + str(c
                    ) + ')'))
                coefficientList.append(1.0)
                actualUnitNames.append(int(unit))
```

```
                    LHS = [variableList, coefficientList] # CPLEX API needs LHS to be
                        variables, then coefficients
                    solver.add(LHS, 'L', len(constraint) - 1) # Add packing
                        constraint to problem
                    globalVars.numberLazyConstraintsAdded[len(constraint)] += 1
                    #print LHS, len(restrictedUnitList) - 1
                    print actualUnitNames, c
                    # If only one unique channel, add the same constraint on every
                        channel
                    if globalVars.NUMBER_UNIQUE_CHANNELS == 1:
                        for chan in range(0, globalVars.NUMBER_AVAILABLE_CHANNELS):
                            if chan <> c:
                                variableList = []
                                coefficientList = []
                                LHS = []
                                for unit in constraint:
                                    if isPyomoInterface: variableList.append('x' +
                                        str(unitDict[int(unit),chan]))
                                    else: variableList.append(str('X(u' + str(unit) +
                                        '_' + str(chan) + ')'))
                                    coefficientList.append(1.0)
                                LHS = [variableList, coefficientList] # CPLEX API
                                    needs LHS to be variables, then coefficients
                                #print LHS, len(restrictedUnitList) - 1
                                solver.add(LHS, 'L', len(constraint) - 1) # Add
                                    packing constraint to problem
    print 'Continuing CPLEX search...'

# Lazy constraint callback: Using the current CPLEX solution, check if any assignment
    violates the cumulative interference constraints.  If so, add a packing
    constraint preventing that assignment.
# For now, assumes all channels are the same frequency.
def callback_LazyConstraint(solver, model = None):
    addLazyConstraints(solver.cplex, model, True)

# CPLEX lazy constraint callback (similar but different from above; can't just use
    the same code because this refers to .self object, unlike above)
class cplexLazyConstraintCallback(LazyConstraintCallback):
    def __call__(self):
        addLazyConstraints(self, None, False)

# Solve using CPLEX
def solveMOCAPUsingCPLEX(pickledPairwiseConstraintsFileName, initialSolutionFileName,
    useCPLEXAPI, adjustForNonSharingUnit, addMaxClique):
    # See if any units require their own channel assignment, and if so, temporarily
        adjust input data so CPLEX doesn't consider these units
    nonSharerList = None
    if adjustForNonSharingUnit:
        print 'Adjusting input data for units that can't share channels...'
        nonSharerList = cpUtil.removeNonSharingUnits()

    # Create a solver
```

```
print 'Creating Pyomo model for CPLEX...'
opt = SolverFactory('cplex',solver_io='python')
cap = ConcreteModel() # Create the problem

# Indices
print 'Creating CPLEX indices...',
cap.u = Set(initialize=['u' + str(u) for u in range(0, globalVars.numberUnits)],
    ordered=True) # Index on each unit
cap.c = Set(initialize=[c for c in range(0, globalVars.NUMBER_AVAILABLE_CHANNELS)
    ], ordered=True) # Index on channels
print 'Done.'

 # Variables
print 'Creating variables...',
cap.Y = Var(cap.c, domain=Binary, initialize = 0) # Indicates if channel c is
    being used
cap.X = Var(cap.u, cap.c, domain=Binary, initialize = 0) # Indicates if unit u is
     using channel c
print 'Done.'

if not initialSolutionFileName is None: # If initial solution was provided, read
    it in and use it
    print 'Reading in initial X and Y values...',
    util.readSolutionFile(solutionFileName = initialSolutionFileName,
        shiftChannelAssignmentsToZero = True, nonSharerList = nonSharerList) #
        Read in solution to globalVars.channelAssignment (will be overwritten)
    for i in range(0, globalVars.numberUnits): # For each unit
        cap.Y[globalVars.channelAssignment[i]].value = 1 # Assign Y values
        cap.X['u'+ str(i), globalVars.channelAssignment[i]].value = 1 # Assign
            unit X values
    print 'Done.'

# Objective function
print 'Creating objective function...',
cap.Obj = Objective(rule = cap_objective_rule, sense=minimize)
print 'Done.'

# Constraints
print 'Creating CAP constraints...',
cap.utilizationConstraints = Constraint(cap.u, cap.c, rule=
    cap_utilizationConstraint_rule)
cap.multiplicityConstraints = Constraint(cap.u, rule=
    cap_multiplicityConstraint_rule)
#cap.objectiveValueLBConstraint = Constraint(rule=
    cap_objectiveValueLBConstraint_rule) # Provide a lower bound to the objective
     value, in hopes of speeding up convergence
cap.Cuts = ConstraintList() # To add cuts dynamically using Cuts.add()
print 'Done.'

maximumClique = []
if addMaxClique:
    print 'Adding maximal clique constraint...'
```

```
    numberCliqueConstraintsAdded = 0

    numberCliqueConstraintsAdded, maximumClique = cpUtil.addCliqueConstraints(
        pickledPairwiseConstraintsFileName, cap)

print 'Creating pairwise interference constraints...'
#cpUtil.calcHigherOrderConstraints(pickledPairwiseConstraintsFileName) # Comment
    out if not wanting to print higher-order constraints
globalVars.numberLazyConstraintsAdded = np.zeros(globalVars.
    MAX_INTERFERENCE_TUPLE_SIZE,dtype=np.int32) # Number of lazy constraints
    added, by the number of units in lazy constraint (starting with zero at index
     0)
print str(cpUtil.addPairwiseConstraints(pickledPairwiseConstraintsFileName,
    maximumClique, cap)) + ' pairwise constraints added (for one channel).' #
    Add pairwise constraints and print number

# The following is needed when using Pyomo-CPLEX callbacks to get current
    solution directly from CPLEX, because CPLEX doesn't know original names, only
     the index order (which is alphabetical)
if not useCPLEXAPI: # Using lazy constraints, and the Pyomo-CPLEX interface
    opt._allow_callbacks = True
    opt._initialize_callbacks(cap)
    opt.set_callback('lazycut-callback', callback_fn=callback_LazyConstraint) #'
        lazycut-callback' # Available callbacks listed on line 1168 in \Lib\site
        -packages\pyomo\solvers\plugins\solvers\CPLEXDirect.py
    global unitList # The CPLEX order of unit X variables
    unitList = [] # Reset, in case running CPLEX more than once in a row
    for i in range(0, globalVars.numberUnits): unitList.append('u' + str(i))
    global unitDict # Looks up unit number and channel, and returns the CPLEX var
        iable number
    unitDict = {} # Reset, in case running CPLEX more than once in a row
    unitStart = globalVars.NUMBER_AVAILABLE_CHANNELS # Advance to the X_u values,
         which follow after the Y_C
    counter = unitStart+1 # +1 needed b/c CPLEX variables begin with x1, not x0
    for unit in unitList:
        for c in range(0, globalVars.NUMBER_AVAILABLE_CHANNELS):
            unitDict[int(unit[1:]),c]=counter
            counter += 1

# Solve CAP using CPLEX with Pyomo interface
if not useCPLEXAPI: # If using the Pyomo-CPLEX interface
    print 'Pyomo model created.  Solving using CPLEX and Pyomo interface...',
    opt.options['threads'] = 8 # Options
    opt.options['timelimit'] = globalVars.CPLEX_TIME_LIMIT
    # opt.set_options('mip_ordertype=3') # An example of sending options to CPLEX
    capResults = opt.solve(cap, tee=True) # Run CPLEX, showing output, but not
        keeping intermediate files # Use warmstart=True to use initial feasible
        solution; Use keepfiles=False to not keep intermediary files
    print 'Done.'
    print 'Loading results from CPLEX...',
    cap.solutions.load_from(capResults) # Load results
    print 'Done.'
```

```
    # Check if problem is feasible
    if capResults.solver.termination_condition == TerminationCondition.infeasible
        : # Infeasible;
        print 'Problem is infeasible.'
        return "infeasible"
    elif not globalVars.isCPLEXSolutionFeasible: # Lazy constraint callback never
         found a feasible sol'n
        print 'CPLEX could not find a feasible solution.'
        if not initialSolutionFileName is None:
            print 'Resorting to initial solution...' # Which has been stored all
                along in globalVars.channelAssignment
        else:
            print 'No initial solution given; returning without solution.'
            return 'infeasible'
    else: # Problem is feasible; load results into variables
        if not initialSolutionFileName is None and len(np.unique(globalVars.
            channelAssignment[0:globalVars.numberUnits])) < cap.Obj.expr():
            print 'Initial feasible solution was better than that found by CPLEX.
                  Resorting to initial solution...' # Stored in globalVars.
                channelAssignment
        else: # CPLEX seems to have worked
            globalVars.channelAssignment[:] = -999
            for u in cap.u: # Get assignments from CPLEX solution
                for c in cap.c:
                    if round(cap.X[u,c].value) == 1.0: globalVars.
                        channelAssignment[int(u[1:])] = c

    if not nonSharerList is None and len(nonSharerList) > 0: cpUtil.
        reinsertNonSharingUnits(nonSharerList) # If non-channel-sharing units
        have been removed, re-insert them
    globalVars.numberRequiredChannels = len(np.unique(globalVars.
        channelAssignment[0:globalVars.numberUnits]))

    if nonSharerList is None and (len(np.unique(globalVars.channelAssignment[0:
        globalVars.numberUnits])) <> round(cap.Obj.expr()) or any(globalVars.
        channelAssignment[:] == -999)):
        print 'There was a problem in calculating the channel assignments.'

    print 'Problem is feasible.  ', str(round(globalVars.numberRequiredChannels))
        , ' channels required.'
    if globalVars.isCPLEXSolutionFeasible: # If CPLEX found a feasible solultion,
         print the number of lazy constraints
        print str(np.sum(globalVars.numberLazyConstraintsAdded[:])) + ' lazy
            constraints added.'
        for i in range(0, globalVars.MAX_INTERFERENCE_TUPLE_SIZE):
            if globalVars.numberLazyConstraintsAdded[i] > 0: print str(globalVars
                .numberLazyConstraintsAdded[i]) + ' lazy constraints of size ' +
                str(i) + ' added.'
    return 'feasible'
else: # Using a direct connection to CPLEX, by writing an LP file first
```

```
print 'Pyomo model created.  Saving as LP file and setting up CPLEX interface
    ...',
cap.write('pyomoCAPModel.lp', io_options={'symbolic_solver_labels':True}) #
    Write Pyomo model as an LP file
capCPLEX = cplex.Cplex('pyomoCAPModel.lp')
theCPLEXLazyCallback = capCPLEX.register_callback(cplexLazyConstraintCallback
    ) # Register the lazy constraint callback, if needed
print 'Solving using CPLEX Python API...',
capCPLEX.parameters.parallel = -1 # Force CPLEX to be opportunistic in
    multithreading; otherwise, with callbacks, it will be deterministic (and
    use only one thread)
capCPLEX.parameters.threads = 8
capCPLEX.parameters.timelimit = globalVars.CPLEX_TIME_LIMIT
capCPLEX.solve()

print 'Done.  Loading solution...',
capResults = capCPLEX.solution
print 'Done.'

# Check if problem is feasible
if not capResults.is_primal_feasible(): # Infeasible;
    print 'Problem is infeasible.'
    return 'infeasible'
else: # Problem is feasible; load results into variables
    globalVars.numberRequiredChannels = capResults.get_objective_value()
    print 'Problem is feasible.  ', str(round(globalVars.
        numberRequiredChannels,0)), ' channels required.'
    print str(np.sum(globalVars.numberLazyConstraintsAdded[:])) + ' lazy
        constraints added.'
    globalVars.channelAssignment = calcCurrentCPLEXChannelAssignment(
        capResults, False, False)
    if len(np.unique(globalVars.channelAssignment)) <> round(capResults.
        get_objective_value()):
        print 'There was a problem in calculating the channel assignments.'
    return 'feasible'
```

Appendix C: MO-CAP CP Code

This appendix provides partial computer code to bound the MO-CAP Restricted Standard Formulation using constraint programming with Python and CPLEX CP Solver.

```
# Import packages
import globalVars as globalVars # Contains global variables
import Utilities as util # Basic utilities, like reading in input files and pre-
    processing interference values
import cplexUtilities as cpUtil # CPLEX utilities, like adding pairwise constraints
    and checking if units can be on same channel
import numpy as np
import itertools as itertools # For iterating over combinations of units in
    LazyConstraints callback
from os import path # For checking if a file exists
import time # For measuring elapsed processing time
import pickle # For saving pickled array
import pandas as pd # For importing .csv files (faster than openpyxl)
import subprocess # For running OPL model as a subprocess

# Write the constraint satisfaction problem in OPL format, with the given
    constraintList
def writeOplFile(constraintList, numberChannels, maximumClique = None):
    print 'Creating OPL mod file...',
    modelFile = open('oplCPmodel.mod', 'wb')
    modelFile.write('using CP;\n')
    modelFile.write('range u= 0..' + str(globalVars.numberUnits-1) + ';\n') # Minus
        one b/c we index by zero
    modelFile.write('range c= 0..' + str(numberChannels-1) + ';\n') # Minus one b/c
        we index by zero
    modelFile.write('dvar int X[u] in c;\n')
    if not maximumClique is None: # Add index set for maximumClique
        theString = '{int} maximumClique = {'
        for i in sorted(maximumClique):
            theString = theString + str(i) + ','
        theString = theString[:-1] # Remove last comma
        theString = theString + '};'
        modelFile.write(theString)
    modelFile.write('subject to {\n')
    if not maximumClique is None: # Add maximumClique as allDifferent constraint
        modelFile.write('allDifferent(all(i in maximumClique) X[i]);\n')

    # Write all constraints
    for i in constraintList:
        modelFile.write(str(i) + '\n')

    modelFile.write('}\n') # Close constraints block
    modelFile.write('execute {\n')
    modelFile.write('var f=new IloOplOutputFile('oplCPModelOutput.txt');\n')
```

```
    modelFile.write('for(var i in u)\n')
    modelFile.write('    f.writeln(X[i]);\n')
    modelFile.write('f.close();\n')
    modelFile.write('}\n')
    modelFile.close()
    print 'Done.  ' + str(len(constraintList)) + ' constraints in model.'

# Read in currentChannelAssignment from OPL model's output
def getCurrentChannelAssignment():
    currentChannelAssignment = np.empty(globalVars.numberUnits, dtype=np.int32)  #
        Current channel assignment solution, by unit
    theDataFrame = pd.read_csv('oplCPModelOutput.txt',header=None) # Read in using
        pandas
    for i in range(0, globalVars.numberUnits):
        currentChannelAssignment[i] = theDataFrame.values[i,0]
    return currentChannelAssignment

# Find infeasibilities in currentChannelAssignment, and add constraints to
    constraintList to eliminate them
def addNewConstraints(numberChannels, receivedInterference, currentChannelAssignment,
     constraintList):
    constrainedAssignmentsByChannel = [[] for i in range(0,numberChannels)] # A list
        of lists, where the first index is channel and the second is a list of units
        that can't all be assigned that channel
    counter = 0

    # Loop through all radios to find violations, and add the unit to list
    for i in range(0, globalVars.numberUnits):
        for j in range(0, globalVars.numberSubUnits[i]):
            if receivedInterference[counter] > globalVars.maxInterference[counter,
                currentChannelAssignment[i]]:
                constrainedAssignmentsByChannel[currentChannelAssignment[i]].append(i
                    ) # Add this unit to the list of units that can't be assigned
                    this channel
                counter = counter + (globalVars.numberSubUnits[i] - j)
                break
            counter += 1

    # Loop through all channels and add to constrainedAssignmentsByChannel all units
        on a violated channel that haven't yet been added
    for c in range(0, numberChannels):
        if len(constrainedAssignmentsByChannel[c]) > 0: # If there are violations on
            this channel, add all units not already added (checked using the .count()
             method)
            for i in range(0, globalVars.numberUnits):
                if currentChannelAssignment[i] == c and
                    constrainedAssignmentsByChannel[c].count(i) == 0:
                    constrainedAssignmentsByChannel[c].append(i)

    # Add original packing constraints (i.e., allow |S|-1 units in the subset S of
        units, on the assigned (and violated) channel)
    for c in range(0, numberChannels):
```

```
        if len(constrainedAssignmentsByChannel[c]) > 0: # If there are violations on
            this channel
            restrictedUnitList = [] # List of constraints of units on this c that can
                't be assigned together
            if len(constrainedAssignmentsByChannel[c]) == 3: # If exactly three, add
                all three (since we've already added all pairs)
                restrictedUnitList.append(constrainedAssignmentsByChannel[c])
            elif len(constrainedAssignmentsByChannel[c]) > 3: # If more than three,
                check for disallowed n-tuples among the four or more units
                tupleSize = 3
                while len(restrictedUnitList) == 0 and tupleSize <= len(
                    constrainedAssignmentsByChannel[c]): # Go until a restricted
                    subset is found, or tupleSize is bigger than the number of units
                    (in which case, there's a problem)
                    listOfUnitCombinations = itertools.combinations(
                        constrainedAssignmentsByChannel[c], tupleSize) # Get all
                        combinations of tupleSize among units (where order doesn't
                        matter)
                    for unit in listOfUnitCombinations: # Check if a combination isn'
                        t allowed; if so, add to list
                        theList = list(unit) # Convert tuple to list
                        if not cpUtil.canTheseUnitsShareChannelAssignment(theList): #
                             If they can't share
                            restrictedUnitList.append(theList) # Add each unit to the
                                 associated constraint
                    tupleSize += 1 # Increase tupleSize for next iteration

            if len(restrictedUnitList) == 0:
                print 'Problem: There aren't violations on this channel.'
            # Loop over all units for each constraint on this channel and add to
                constraintList
            for constraint in restrictedUnitList: # For each channel with a
                constraint
                # Format: (X[i] == X[j]) && (X[j] == X[k]) => (X[k] != X[l]);
                theString = '''
                for i in range(0, len(constraint)-2): # First part of constraint
                    if i <> 0: theString = theString + ' && '
                    theString = theString + '(X[' + str(constraint[i]) + '] == X[' +
                        str(constraint[i+1]) + '])'
                theString = theString + ' => (X[' + str(constraint[len(constraint)
                    -2]) + '] != X[' + str(constraint[len(constraint)-1]) + ']);'
                #print 'New constraint: ' + theString
                constraintList.append(theString)
    return constraintList

# Solve the constraint satisfaction problem using the given number of channels.
# Parameter addHigherOrder=True will dynamically check the true (original problem)
    feasibility of a CP solution, add higher-order constraints, and resolve until
    infeasible
def solveConstraintSatisfactionProblem(inputFileName,
    pickledPairwiseConstraintsFileName, solutionFileName, numberChannels,
    addMaximumClique, addHigherOrder, maxIterations):
```

```
# Get all pairwise constraints and add to constraintsList
constraintList = [] # Dynamic list to hold constraints as they're added
print 'Opening pickledPairwiseConstraint file: ' + str(
    pickledPairwiseConstraintsFileName) + '...',
numberPairwiseConstraints = 0
canUnitsShareChannelAssignmentArray = calcUnitsShareChannelAssignmentArray(
    pickledPairwiseConstraintsFileName, numberUnits) # Calculate and return the
    canUnitsShareChannelAssignmentArray

maximumClique = None
if addMaximumClique:
    print 'Adding maximal clique constraint...'
    maximumClique = []
    # Create NetworkX graph object from pairwise constraints
    import networkx as netX
    theGraph = netX.Graph() # Create default graph
    theGraph.add_nodes_from(range(0,globalVars.numberUnits-1)) # Add all units as
         nodes (-1 because of how NetworkX creates nodes)
    for i in range(globalVars.numberUnits):
        for j in range(globalVars.numberUnits):
            if i < j and not canUnitsShareChannelAssignmentArray[i,j]: theGraph.
                add_edge(i,j)

    # Find maximal cliques (largest clique, for each node). The biggest maximal
        clique is the maximum clique
    maximalCliqueGenerator = netX.find_cliques(theGraph) # A generator of all of
        the maximal cliques
    maximalCliqueList = [] # Copy to list to work with it
    for i in maximalCliqueGenerator: maximalCliqueList.append(i)
    [maximumCliqueLength, maximumClique] = max(enumerate(maximalCliqueList), key
        = lambda tup: len(tup[1])) # Lambda function to get maximum clique (
        biggest maximal)
    print 'Maximum clique (size ' + str(len(maximumClique)) + '): ', sorted(
        maximumClique)

print 'Adding pairwise interference constraints...'
for i in range(0, globalVars.numberUnits):
    for j in range(0, globalVars.numberUnits):
        if addMaximumClique and maximumClique.count(i) > 0 and maximumClique.
            count(j) > 0: continue # These units are already in maximumClique;
            skip
        elif not i == j and i < j: # Only list pairs of constraints one way,
            since these are allDifferent constraints
            if canUnitsShareChannelAssignmentArray[i, j] == False: # These units
                 can't share a channel
                numberPairwiseConstraints += 1
                constraintList.append('X[' + str(i) + '] != X[' + str(j) + '];')
print 'Done. ' + str(numberPairwiseConstraints) + ' pairwise constraints added.'

# Loop until maxIterations, or until new lower bound found
iterations = 0
isLowerBound = False
```

```
while iterations < maxIterations and not isLowerBound:
    # Create OPL mod (model) file
    if addMaximumClique: writeOplFile(constraintList, numberChannels,
        maximumClique)
    else: writeOplFile(constraintList, numberChannels)

    print 'Running CPLEX CP Optimizer, iteration ' + str(iterations+1) + '...'
    try:
        exitCode = subprocess.check_call(['C:/Program Files/IBM/ILOG/
            CPLEX_Studio1262/opl/bin/x64_win64/oplrun', 'oplCPmodel.mod'])
    except:
        print str(numberChannels) + ' is infeasible for this current constraint
            satisfaction problem and establishes a lower bound (i.e., at least '
            + str(numberChannels+1) + ' channels are required. Current solver
            time: ' + str(time.time() - globalVars.globalStartTime)
        isLowerBound = True
        break # Break out of while loop

    if exitCode == 0:
        print str(numberChannels) + ' is feasible for this current constraint
            satisfaction problem.'
        print 'Reading in CP Optimizer solution to determine if it is feasible in
             original problem...'
        currentChannelAssignment = getCurrentChannelAssignment()
        numberReceivedExcessiveInterference, receivedInterference = util.
            calcReceivedInterference(currentChannelAssignment)
        if numberReceivedExcessiveInterference == 0:
            print 'The current solution is feasible in the original problem; this
                 may be a new optimal solution. Current solver time: ' + str(
                time.time() - globalVars.globalStartTime)
            isLowerBound = True # Break out of loop
            globalVars.channelAssignment = currentChannelAssignment # Save
                solution
            globalVars.numberRequiredChannels = len(np.unique(globalVars.
                channelAssignment))
            util.writeMOCAPSolutionFile(solutionFileName)
        else:
            print 'The current solution is NOT feasible for the original MO-CAP.
                 Current solver time: ' + str(time.time() - globalVars.
                globalStartTime)
            if addHigherOrder and iterations < maxIterations-1:
                print 'Adding new constraints to pursue a MO-CAP-feasible lower
                    bound...'
                addNewConstraints(numberChannels, receivedInterference,
                    currentChannelAssignment, constraintList)
        iterations += 1
    else:
        print 'Some other exit code.'
print 'Done.'
```

The following is an example of the OPL code generated by the above Python code.

```
using CP;
range u= 0..117;
range c= 0..45;
dvar int X[u] in c;
{int} maximumClique = {1,2,3};

subject to {
allDifferent(all(i in maximumClique) X[i]);
X[0] != X[1];
}

execute {
var f=new IloOplOutputFile('oplCPModelOutput.txt');
for(var i in u)
   f.writeln(X[i]);
f.close();
}
```

Appendix D: MI-CAP Clustering Code

This appendix provides partial computer code to solve the MI-CAP using the k-medoids clustering method.

```
# Portions originally based on https://github.com/salspaugh/machine_learning/blob/
    master/clustering/kmedoids.py by salepaugh.

# Import packages
import globalVars as globalVars # Contains global variables
import Utilities as util # Basic utilities, like reading in input files and pre-
    processing interference values
import numpy as np
import mpmath as mp # Import mpmath library, for abitrary-precision floating-point
    numbers
import sys

def assign_points_to_clusters(medoids, dissimilarityArray): # Assign each point to
    the closest medoid
    distances_to_medoids = dissimilarityArray[:,medoids]
    clusters = medoids[np.argmin(distances_to_medoids, axis=1)]
    clusters[medoids] = medoids
    return clusters

def compute_new_medoid(cluster, dissimilarityArray): # Pick and return that point in
    this cluster that minimizes distances; make it the new medoid in this cluster
    mask = np.ones(dissimilarityArray.shape)
    mask[np.ix_(cluster,cluster)] = 0.
    cluster_distances = np.ma.masked_array(data=dissimilarityArray, mask=mask,
        fill_value=10e9)
    costs = cluster_distances.sum(axis=1)
    return costs.argmin(axis=0, fill_value=10e9)

# Calculate and return total cost (i.e., total interference) of this assignment of
    clusters
def calcChannelAssignmentCosts(medoids, clusters, dissimilarityArray):
    # Create and populate a temporary channelAssignment array from clusters
    tempChannelAssignment = []

    medoidToChannel = {} # Dictionary indicating the channel assignment for a
        particular medoid
    for index, medoid in enumerate(medoids): medoidToChannel[medoid] = index #
        Populate dictionary

    # Map cluster assignments to channels
    for unit in clusters: tempChannelAssignment.append(medoidToChannel[unit])

    # Calculate interference
    excessiveInterferenceCounter, receivedInterference = util.
        calcReceivedInterference(tempChannelAssignment)
```

```
    return tempChannelAssignment, excessiveInterferenceCounter, receivedInterference

# Run k-medoids clustering algorithm, save result to globalVars.channelAssignment,
    and return
def kMedoidsCluster(dissimilarityArray, availableChannels, maxIterations,
    pickledPairwiseConstraintsFileName):
    print 'Opening pickledPairwiseConstraint file: ' + str(
        pickledPairwiseConstraintsFileName) + '...',
    canUnitsShareChannelAssignmentArray = calcUnitsShareChannelAssignmentArray(
        pickledPairwiseConstraintsFileName, numberUnits) # Calculate and return the
        canUnitsShareChannelAssignmentArray

        m = dissimilarityArray.shape[0] # number of points
    best_medoids = np.array([-1]*availableChannels) # Best found solution
    best_clusters = [] #assign_points_to_clusters(best_medoids, dissimilarityArray)
    bestReceivedInterference = mp.mpf('inf') # Best (least) total received
        interference thus far

    # Loop until maxIterations hit
    iterations = 0
    while iterations < maxIterations:
        # Pick c=availableChannels random medoids.
        curr_medoids = np.array([-1]*availableChannels)
        while not len(np.unique(curr_medoids)) == availableChannels:
            curr_medoids = np.array([random.randint(0, m - 1) for _ in range(
                availableChannels)]) # Randomly pick medoids
        old_medoids = np.array([-1]*availableChannels) # Doesn't matter what we
            initialize these to.
        new_medoids = np.array([-1]*availableChannels)

        # Loop until the medoids stop updating or maxIterations is hit
        clusters = assign_points_to_clusters(curr_medoids, dissimilarityArray) #
            Assign each point to cluster with closest medoid.
        while iterations < maxIterations and not ((old_medoids == curr_medoids).all()
            ):
            # Update cluster medoids to be lowest cost point.
            for curr_medoid in curr_medoids:
                cluster = np.where(clusters == curr_medoid)[0]
                new_medoids[curr_medoids == curr_medoid] = compute_new_medoid(cluster
                    , dissimilarityArray) # Find that point in this cluster that
                    minimizes distances; make it the new medoid in this cluster

            old_medoids[:] = curr_medoids[:]
            curr_medoids[:] = new_medoids[:]

            # Assign each point to cluster with closest medoid.
            clusters = assign_points_to_clusters(curr_medoids, dissimilarityArray)

            # Check assignment costs (i.e., interference)
```

```
            tempChannelAssignment, excessiveInterferenceCounter, receivedInterference
                = calcChannelAssignmentCosts(curr_medoids, clusters,
                dissimilarityArray)
            print 'Clustering iteration ' + str(iterations+1) + ': Received
                interference = ' + str(mp.fsum(receivedInterference)) + '; Number
                received excessive interference = ' + str(
                excessiveInterferenceCounter)

            # Save if this is new incumbent
            theSumInterference = mp.fsum(receivedInterference)
            if theSumInterference < bestReceivedInterference:
                print 'New incumbent solution.'
                best_medoids = np.copy(curr_medoids)
                best_clusters = list(clusters)
                bestReceivedInterference = theSumInterference

            iterations += 1

    # Save solution and return
    globalVars.channelAssignment, excessiveInterferenceCounter, receivedInterference
        = calcChannelAssignmentCosts(best_medoids, best_clusters, dissimilarityArray)
    print 'Calculating number of pairwise constraint violations...',
    pairwiseViolations = 0
    for i in range(globalVars.numberUnits):
        for j in range(globalVars.numberUnits):
            if i > j:
                if globalVars.channelAssignment[i] == globalVars.channelAssignment[j]
                    and canUnitsShareChannelAssignmentArray[i,j] == False:
                    pairwiseViolations += 1
    print 'Done.'

    print 'Done. Final clustering solution received interference = ,' + str(mp.fsum(
        receivedInterference)) + ', Number received excessive interference = ,' + str
        (excessiveInterferenceCounter) + ', Number pairwise violations = ,' + str(
        pairwiseViolations)
    return best_clusters, best_medoids
```

Appendix E: MI-CAP CP Code

This appendix provides partial computer code to solve the MI-CAP using Python and CPLEX CP Solver.

```
# Import packages
import globalVars as globalVars # Contains global variables
import Utilities as util # Basic utilities, like reading in input files and pre-
    processing interference values
import cplexUtilities as cpUtil # CPLEX utilities, like adding pairwise constraints
    and checking if units can be on same channel
import numpy as np
from os import path # For checking if a file exists
import mpmath as mp # Import mpmath library, for abitrary-precision floating-point
    numbers
import time # For measuring elapsed processing time
import pickle # For saving pickled array
import pandas as pd # For importing .csv files (faster than openpyxl)
import subprocess # For running OPL model as a subprocess

# Write the constraint satisfaction problem in OPL format, with the given
    constraintList
def writeOplFile(constraintList, numberChannels, numberPairwiseConstraints,
    maxPenalty, cpTimeLimit):
    print 'Creating OPL mod file...',
    modelFile = open('oplCPmodel.mod', 'wb')
    modelFile.write('using CP;\n')

    modelFile.write('range u= 0..' + str(globalVars.numberUnits-1) + ';\n') # Minus
        one b/c we index by zero
    modelFile.write('range c= 0..' + str(numberChannels-1) + ';\n') # Minus one b/c
        we index by zero
    modelFile.write('dvar int X[u] in c;\n')
    modelFile.write('range numberPenalties=0..' + str(numberPairwiseConstraints-1) +
        ';\n') # Number of pairwise penalties
    modelFile.write('range penaltyRange=0..' + str(maxPenalty) + ';\n') # Range of
        penalty values
    modelFile.write('dvar int penalty[numberPenalties] in penaltyRange;\n')
    modelFile.write('execute{ \n')
    modelFile.write('cp.param.timeLimit=' + str(cpTimeLimit) + ';\n')
    modelFile.write('}\n')
    modelFile.write('\n')
    modelFile.write('minimize sum(j in numberPenalties) penalty[j];\n')

    modelFile.write('subject to {\n')
    # Write all constraints
    for i in constraintList:
        modelFile.write(str(i) + '\n')
```

```
    modelFile.write('}\n') # Close constraints block
    modelFile.write('execute {\n')
    modelFile.write('var f=new IloOplOutputFile('oplCPModelOutput.txt');\n')
    modelFile.write('f.writeln(cp.getObjValue());\n')
    modelFile.write('for(var i in u)\n')
    modelFile.write('   f.writeln(X[i]);\n')
    modelFile.write('for(var i in numberPenalties)\n')
    modelFile.write('   f.writeln(penalty[i]);\n')
    modelFile.write('f.close();\n')
    modelFile.write('}\n')
    modelFile.close()
    print 'Done.  ' + str(len(constraintList)) + ' constraints in model.'

# Read in currentChannelAssignment from OPL model's output
def getCurrentChannelAssignment(numberPairwiseConstraints):
    currentChannelAssignment = np.empty(globalVars.numberUnits, dtype=np.int32)  #
        Current channel assignment solution, by unit
    theDataFrame = pd.read_csv('oplCPModelOutput.txt',header=None) # Read in using
        pandas
    objectiveValue = theDataFrame.values[0,0] # Number of pairwise violations (which
        is the objective function, assuming unweighted)
    for i in range(0, globalVars.numberUnits): currentChannelAssignment[i] =
        theDataFrame.values[i+1,0] # Get the channel assignment
    numberPairwiseViolations = 0
    for i in range(0, numberPairwiseConstraints): # Get the number of pairwise
        violations
        if theDataFrame.values[i+1+globalVars.numberUnits,0] >= 1:
            numberPairwiseViolations += 1
    return objectiveValue, numberPairwiseViolations, currentChannelAssignment

# Solve the MI-CAP optimal soft arc consistency problem using the given number of
    channels. pickledPairwisePenaltiesFileName is pickled array of penalties for each
     pairwise constraint violation.
def solveConstraintSatisfactionProblem(inputFileName,
    pickledPairwiseConstraintsFileName, numberChannels, cpTimeLimit,
    pickledPairwisePenaltiesFileName = None):
    # Get all pairwise constraints and add to constraintsList
    constraintList = [] # Dynamic list to hold constraints as they're added
    print 'Opening pickledPairwiseConstraint file: ' + str(
        pickledPairwiseConstraintsFileName) + '...',
    numberPairwiseConstraints = 0
    canUnitsShareChannelAssignmentArray = util.calcUnitsShareChannelAssignmentArray(
        pickledPairwiseConstraintsFileName, globalVars.numberUnits) # Calculate and
        return the canUnitsShareChannelAssignmentArray

    print 'Calculating pairwise constraint violation penalties...',
    penaltyArray = np.ones((globalVars.numberUnits, globalVars.numberUnits),dtype=np.
        int32) # Initialize to zero
    if not pickledPairwisePenaltiesFileName is None: # Use pickled penalty file
        if path.isfile('obj/'+ pickledPairwisePenaltiesFileName + '.pkl'): # Load
            pickled pairwise constraint violation penalties
            with open('obj/' + pickledPairwisePenaltiesFileName + '.pkl', 'rb') as f:
```

```
            penaltyArray = pickle.load(f)
        print 'Done.'
    else:
        print 'Problem: The pickledPairwisePenaltiesFileName file does not exist.
            '
        return
print 'Done.'

print 'Adding pairwise interference constraints...'
numberPairwiseConstraints = 0
maxPenalty = 0
for i in range(0, globalVars.numberUnits):
    for j in range(0, globalVars.numberUnits):
        if not i == j and i < j: # Only list pairs of constraints one way
            if canUnitsShareChannelAssignmentArray[i, j] == False:  # These units
                 can't share a channel
                constraintList.append('penalty[' + str(numberPairwiseConstraints)
                     + '] == ' + str(penaltyArray[i,j]) + ' * (X[' + str(i) + ']
                    == X[' + str(j) + ']);')
                numberPairwiseConstraints += 1
                if penaltyArray[i,j] > maxPenalty: maxPenalty = penaltyArray[i,j]
print 'Done. ' + str(numberPairwiseConstraints) + ' pairwise constraints added.'

# Create OPL mod (model) file
writeOplFile(constraintList, numberChannels, numberPairwiseConstraints,
    maxPenalty, cpTimeLimit)

print 'Running CPLEX CP Optimizer...'
try:
    exitCode = subprocess.check_call(['C:/Program Files/IBM/ILOG/CPLEX_Studio1262
        /opl/bin/x64_win64/oplrun', 'oplCPmodel.mod'])
except:
    print 'CPLEX CP Optimizer error.'

if exitCode == 0:
    print 'Reading in CP Optimizer solution...'
    objectiveValue, numberPairwiseViolations, currentChannelAssignment =
        getCurrentChannelAssignment(numberPairwiseConstraints)
    globalVars.channelAssignment = currentChannelAssignment # Save solution
    globalVars.numberRequiredChannels = len(np.unique(globalVars.
        channelAssignment))
    numberReceivedExcessiveInterference, receivedInterference = util.
        calcReceivedInterference(currentChannelAssignment)
    print 'Done.  CP solution objective value = ' + str(objectiveValue) + ';
        number of pairwise constraint violations = ' + str(
        numberPairwiseViolations) + '; received interference = ' + str(mp.fsum(
        receivedInterference)) + '; Number radios received excessive interference
         = ' + str(numberReceivedExcessiveInterference)
else:
    print 'Some other exit code.'
```

The following is an example of the OPL code generated by the above Python code.

```
using CP;
range u= 0..117;
range c= 0..34;
dvar int X[u] in c;
range numberPenalties=0..3748;
range penaltyRange=0..1;
dvar int penalty[numberPenalties] in penaltyRange;

execute{
cp.param.timeLimit=500;
}

minimize sum(j in numberPenalties) penalty[j];
subject to {
penalty[0] == 1 * (X[0] == X[1]);
}

execute {
var f=new IloOplOutputFile('oplCPModelOutput.txt');
f.writeln(cp.getObjValue());
for(var i in u)
   f.writeln(X[i]);
for(var i in numberPenalties)
   f.writeln(penalty[i]);
f.close();
}
```

Appendix F: MC-CAP-T Code

This appendix provides partial computer code to solve the MC-CAP-T Decomposition Formulation using Python.

```
# Import packages
import globalVars as globalVars # Contains global variables
import Utilities as util # Basic utilities, like reading in input files and pre-
    processing interference values
import numpy as np
import math
import munkres # Munkres / Hungarian algorithm, for solving the assignment problem in
    O(n^3) time
import sys
from os import path # For checking if a file exists

# Solve min-cost coloring problem using assignment problem formulation (to support MC
    -CAP-T) using Munkres code. Finds the least-cost coloring over time, given MC-
    CAP solutions at each timeStep
def solveMinCostColoringAssignmentUsingMunkres(numberTimeSteps, naiveSolutionFileName
    = None):
    # Identify groups of units (i.e., units assigned the same channel, for each time
        step)
    print 'Identifying groups of units...',
    groupList = util.getGroupList(globalVars.channelAssignment) # List of lists.
        First index is timeStep; Second is group number in that timeStep (not the
        same thing as channel number)
    numberGroups = np.max(globalVars.numberRequiredChannels)
    print 'Done.'

    # Calculate the cost of a naive coloring (i.e., just coloring in the order of
        groups as they appear)
    print 'Calculating cost of naive coloring...'
    naiveColoringCost = util.calcNaiveColorCost(groupList, naiveSolutionFileName)
    print 'Done. Naive coloring cost is ' + str(naiveColoringCost)

    # Calculate the association costs, i.e., the cost of associating g with h at
        timeStep t (h is at timeStep t)
    print 'Calculating costs of associating groups at each timeStep...',
    assignmentCosts = np.zeros((numberGroups, numberGroups, numberTimeSteps-1),dtype=
        np.int32) # Costs of associating g at t with h at timeStep t+1
    for t in range(0, numberTimeSteps-1): # Loop through each timeStep in groupList
        for g in range(numberGroups): # Get each group from current timeStep
          for h in range(numberGroups): # Get each group from next timeStep
            newUnitList = [] # Units that are in h but not g
            if len(groupList[t]) > g and len(groupList[t+1]) > h: # If real (non-
                virtual) groups exist at t and t+1, count all new units in t+1
                for theUnit in groupList[t+1][h]: # For each unit in h
                    if groupList[t][g].count(theUnit) == 0: newUnitList.append(
                        theUnit) # If in h but not previously in g, count it
```

```
        elif len(groupList[t]) <= g and len(groupList[t+1]) > h: # If real (non-
            virtual) groups exist only at t+1, count all units
            for theUnit in groupList[t+1][h]: newUnitList.append(theUnit) # Count
                everything in h
        if len(newUnitList) > 0: # If new units are added at t+1, count cost
            cost = 0 # Cost of associating g at t with h at time t+1
            for theUnit in newUnitList: cost = cost + globalVars.numberSubUnits[
                theUnit]
            assignmentCosts[g,h,t] = cost
print 'Done.'

# Calculate group assignments
print 'Calculating group assignments over all timeSteps using Munkres / Hungarian
    algorithm...'
groupAssignment = np.empty((numberGroups, numberTimeSteps),dtype=np.int32) #
    Indicates assignment of group g (first index) at time t (second index) to a
    group h at t+1 (the value at [g,t])
groupChannelAssignment = np.empty((numberGroups, numberTimeSteps), dtype=np.int32
    ) # Actual channel number to assign to group g (first index) at time t (
    second index)
totalCost = 0
channel = 0
for group in range(numberGroups): # Assign channel numbers for first timeStep
    groupChannelAssignment[group][0] = group
for g, group in enumerate(groupList[0]): # Assign channel numbers to units for
    first timeStep
    for unit in group: globalVars.channelAssignment[unit][0] =
        groupChannelAssignment[g][0]
globalVars.numberRequiredChannelChanges.append(0) # No channel changes required
    to get to timeStep 0
for t in range(0, numberTimeSteps-1): # Loop through each timeStep to calculate
    cost of going from g (at t) to h (at t+1)
    costMatrix = [] # Reset
    costMatrix = np.copy(assignmentCosts[:,:,t]).tolist() # Create costMatrix for
        Munkres algorithm (Munkres doesn't work with arrays, just lists)
    m = munkres.Munkres() # Create Munkres instance
    outputIndices = m.compute(costMatrix) # Solve assignment problem using
        Munkres
    timeStepCost = 0
    for g, h in outputIndices: # Loop through outputIndices, which indicates
        assignment of g to h
        cost = costMatrix[g][h]
        timeStepCost += cost
        groupAssignment[g][t] = h # Record assignments
        if len(groupList[t+1]) > h: # If h is a real group of units at t+1,
            assign channel number to each unit in group h at t+1 (i.e., 'carry
            forward' channel assignment along path)
            for unit in groupList[t+1][h]: globalVars.channelAssignment[unit][t
                +1] = groupChannelAssignment[g][t]
        groupChannelAssignment[h][t+1] = groupChannelAssignment[g][t] # Save
            channel number for next timeStep
```

```
        print 'Cost from timeStep ' + str(t) + ' to timeStep ' + str(t+1) + ' is: ' +
            str(timeStepCost)
        globalVars.numberRequiredChannelChanges.append(timeStepCost)
        totalCost += timeStepCost

    # Print and save results
    print 'Done.  ' + str(totalCost) + ' radios require channel changes over all
        timesteps.'
    percentageFewer = 0
    if naiveColoringCost == 0: percentageFewer = 0
    else: percentageFewer = (naiveColoringCost - totalCost)/float(naiveColoringCost)
    print 'Naive coloring cost is ' + str(naiveColoringCost) + '.  Optimization
        requires ' + str(percentageFewer) + ' percent fewer channel changes.'
    return 'feasible'
```

Bibliography

Aardal, K., Hipolito, A., Van Hoesel, C., Jansen, B., Roos, C., Terlaky, T. (1996). *A branch-and-cut algorithm for the frequency assignment problem.* METEOR, Maastricht Research School of Economics of Technology and Organizations.

Aardal, K., Hurkens, C., Lenstra, J. K., Tiourine, S. (2002). "Algorithms for radio link frequency assignment: The CALMA project." *Operations Research.* 50(6):968–980.

Aardal, K. I., Van Hoesel, S. P., Koster, A. M., Mannino, C., Sassano, A. (2007). "Models and solution techniques for frequency assignment problems." *Annals of Operations Research.* 153(1):79–129.

Abbasi, A. A., Younis, M. (2007). "A survey on clustering algorithms for wireless sensor networks." *Computer Communications.* 30(14):2826–2841.

Aggelou, G. (2004). *Mobile Ad-hoc Networks: From Wireless LANs to 4G Networks.* (McGraw-Hill Professional).

Ahmadi, M., Pan, J. (2011). "Cognitive wireless mesh networks: A connectivity preserving and interference minimizing channel assignment scheme." *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PacRim)*, 458–463. (IEEE).

Ahuja, R. K., Magnanti, T. L., Orlin, J. B. (1993). *Network Flows: Theory, Algorithms, and Applications.* (Saddle River, NJ).

Akyildiz, I. F., Lee, W.-Y., Vuran, M. C., Mohanty, S. (2006). "NeXt generation/dynamic spectrum access/cognitive radio wireless networks: A survey." *Computer Networks.* 50(13):2127–2159.

Akyildiz, I. F., Lee, W.-Y., Vuran, M. C., Mohanty, S. (2008). "A survey on spectrum management in cognitive radio networks." *IEEE Communications Magazine.* 46(4):40–48.

Alion Science and Technology Corporation (2016). "TIREM – Terrain Integrated Rough Earth Model." URL `http://www.alionscience.com/Technologies/Wireless-Spectrum-Management/TIREM/.`

Alouf, S., Altman, E., Galtier, J., Lalande, J.-F., Touati, C. (2005). "Quasi-optimal bandwidth allocation for multi-spot MFTDMA satellites." *IEEE International Conference on Computer Communications*, volume 1, 560–571.

Analytical Graphics, Inc. (2016). "STK." URL `https://www.agi.com/`.

Anderson, L. G. (1973). "A simulation study of some dynamic channel assignment algorithms in a high capacity mobile telecommunications system." *IEEE Transactions on Vehicular Technology*. 22(4):210–217.

Babadi, B., Tarokh, V. (2010). "GADIA: A greedy asynchronous distributed interference avoidance algorithm." *IEEE Transactions on Information Theory*. 56(12):6228–6252.

Barnhart, C., Johnson, E., Nemhauser, G., Savelsbergh, M., Vance, P. (1998). "Branch-and-price: Column generation for solving huge integer programs." *Operations Research*. 46(3):316–329.

Bastian, M., Heymann, S., Jacomy, M. (2009). "Gephi: An Open Source Software for Exploring and Manipulating Networks." URL `http://www.aaai.org/ocs/index.php/ICWSM/09/paper/view/154`.

Berge, C. (1984). *Hypergraphs: Combinatorics of Finite Sets*, volume 45. (Elsevier).

Berry, L. (1990). "The potential contribution of optimum frequency assignment to efficient use of the spectrum." *IEEE International Symposium on Electromagnetic Compatibility*, 409–412. (IEEE).

Borndörfer, R., Eisenblätter, A., Grötschel, M., Martin, A. (1998). "The orientation model for frequency assignment problems." Technical report, TR 98-01.

Boyinbode, O., Le, H., Takizawa, M. (2011). "A survey on clustering algorithms for wireless sensor networks." *International Journal of Space-Based and Situated Computing*. 1(2):130–136.

Bron, C., Kerbosch, J. (1973). "Algorithm 457: Finding all cliques of an undirected graph." *Communications of the ACM*. 16(9):575–577.

Brown, J. I. (1996). "The complexity of generalized graph colorings." *Discrete Applied Mathematics*. 69(3):257–270.

Capone, A., Trubian, M. (1999). "Channel assignment problem in cellular systems: A new model and a tabu search algorithm." *IEEE Transactions on Vehicular Technology*. 48(4):1252–1260.

Casteigts, A., Flocchini, P., Quattrociocchi, W., Santoro, N. (2012). "Time-varying graphs and dynamic networks." *International Journal of Parallel, Emergent and Distributed Systems*. 27(5):387–408.

Chen, D.-S., Batson, R. G., Dang, Y. (2010). *Applied Integer Programming: Modeling and Solution*. (John Wiley & Sons).

Chu, Y., Xia, Q. (2004). "Generating Benders cuts for a general class of integer programming problems." *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, 127–141. (Springer).

Cichon, D. J., Kürner, T. (1993). "Digital mobile radio towards future generation systems: Cost 231 final report." Technical report, European Cooperation in the Field of Scientific and Technical Research-Action COST.

Coltun, R., Ferguson, D., Moy, J., Lindem, A. (2008). "OSPF for IPv6." *IETF: The Internet Engineering Taskforce*. RFC 5340.

Cooper, M., De Givry, S., Schiex, T. (2007). "Optimal soft arc consistency." *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, 68–73. (Morgan Kaufmann Publishers Inc.).

Correia, L. M. (2001). *Wireless Flexible Personalized Communications*. (John Wiley & Sons, Inc.).

Corson, S., Macker, J. (1998). "Mobile ad hoc networking (MANET): Routing protocol performance issues and evaluation considerations." *IETF: The Internet Engineering Taskforce*. RFC 2501.

Crainic, T. G., Le Cun, B., Roucairol, C. (2006). "Parallel branch-and-bound algorithms." *Parallel Combinatorial Optimization*. 1:1–28.

Cuppini, M. (1994). "A genetic algorithm for channel assignment problems." *European Transactions on Telecommunications*. 5(2):285–294.

Daniels, K., Chandra, K., Liu, S., Widhani, S. (2004). "Dynamic channel assignment with cumulative co-channel interference." *ACM SIGMOBILE Mobile Computing and Communications Review*. 8(4):4–18.

Defense Advanced Research Projects Agency (2015). "Advanced RF Mapping (Radio Map)." URL http://www.darpa.mil/program/advance-rf-mapping.

Defense Information Systems Agency (2013). "Spectrum XXI." URL http://www.disa.mil/Services/Spectrum/Enterprise-Services/Spectrum-XXI.

Department of Defense (2007). *Major Combat Operation-1, Swiftly Defeat the Efforts 2014. Multi-Service Force Deployment, Analytic Agenda.*

Department of Defense (2013). *Integrated Security Construct-B. Multi-Service Force Deployment document, scenario 3.*

Digital Globe (2016). URL https://www.digitalglobe.com/.

Dijkstra, E. W. (1959). "A note on two problems in connexion with graphs." *Numerische mathematik.* 1(1):269–271.

Ding, L., Melodia, T., Batalama, S. N., Matyjas, J. D., Medley, M. J. (2010). "Cross-layer routing and dynamic spectrum allocation in cognitive radio ad hoc networks." *IEEE Transactions on Vehicular Technology.* 59(4):1969–1979.

Dowsland, K. A., Dowsland, W. B. (1992). "Packing problems." *European Journal of Operational Research.* 56(1):2–14.

Drummond, L. M., Uchoa, E., Gonçalves, A. D., Silva, J. M., Santos, M. C., de Castro, M. C. S. (2006). "A grid-enabled distributed branch-and-bound algorithm with application on the Steiner problem in graphs." *Parallel Computing.* 32(9):629–642.

Dunkin, N., Bater, J., Jeavons, P., Cohen, D. (1998). "Towards high order constraint representations for the frequency assignment problem." Technical report, University of London, Egham, Surrey, UK.

Dunkin, N., Jeavons, P. (1997). "Expressiveness of binary constraints for the frequency assignment problem." *Proceedings of IEEE/ACM Workshop Dial M for Mobility.* (Citeseer).

Dupont, A., Vasquez, M., Habet, D. (2005). "Consistent Neighbourhood in a Tabu search." *Meta-heuristics: Progress as Real Problem Solvers.* (17):367–386.

El-Bardan, R., Brahma, S., Varshney, P. K. (2014). "Power control with jammer location uncertainty: A game theoretic perspective." *48th Annual Conference on Information Sciences and Systems*, 1–6. (IEEE).

Eppink, D., Kuebler, W. (1994). "TIREM/SEM Handbook." Technical report, Electromagnetic Compatibility Analysis Center, Department of Defense.

Federal Communications Commission (2003). "Notice of proposed rule making and order, ET Docket 03-322." Technical report.

Federal Communications Commission (2014). "Human exposure to radio frequency fields: Guidelines for cellular and PCS sites." URL `http://transition.fcc.gov/cgb/guides/human-exposure-rf-fields-guidelines-cellular-and-pcs-sites.`

Ferreira, A. (2004). "Building a reference combinatorial model for MANETs." *IEEE Network*. 18(5):24–29.

Ferreira, A., Goldman, A., Monteiro, J. (2010). "Performance evaluation of routing protocols for MANETs with known connectivity patterns using evolving graphs." *Wireless Networks*. 16(3):627–640.

Fischetti, M., Lepschy, C., Minerva, G., Romanin-Jacur, G., Toto, E. (2000). "Frequency assignment in mobile radio systems using branch-and-cut techniques." *European Journal of Operational Research*. 123(2):241–255.

Fishburn, P. C., Kim, J., Lagarias, J., Wright, P. E. (1998). "Interference-minimizing colorings of regular graphs." *SIAM Journal on Discrete Mathematics*. 11(1):15–40.

Freuder, E. C., Wallace, R. J. (1992). "Partial constraint satisfaction." *Artificial Intelligence*. 58(1-3):21–70.

Frieze, A., Jerrum, M. (1995). "Improved approximation algorithms for MAX k-CUT and MAX BISECTION." *International Conference on Integer Programming and Combinatorial Optimization*, 1–13. (Springer).

Garcia Villegas, E., Ferro, R. V., Paradells Aspas, J. (2005). "Implementation of a distributed dynamic channel assignment mechanism for IEEE 802.11 networks." *IEEE International Symposium on Personal, Indoor, and Mobile Radio Communications PIMRC*, volume 3, 1458–1462.

(IEEE).

Gephi Consortium (2016). "Gephi, the Open Graph Viz Platform." URL https://gephi.org.

Goldstein, P. (2013). "Pentagon strikes deal with broadcasters, clearing way for 1755-1780 MHz auction." Fierce Wireless, URL http://www.fiercewireless.com.

Gomes, F. C., Pardalos, P., Oliveira, C. S., Resende, M. G. (2001). "Reactive GRASP with path relinking for channel assignment in mobile phone networks." *Proceedings of the International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, 60–67. (ACM).

Google Earth Pro (2016). URL https://www.google.com/earth/.

Gould, R. (1988). *Graph Theory*. (Cummings).

Goulding, V. J. (2009). "Enhanced MAGTF Operations: Capitalizing on lessons learned." *Marine Corps Gazette*. 93(8):13.

Grötschel, M., Lovász, L. (1995). "Combinatorial optimization." *Handbook of Combinatorics*. 2:1541–1597.

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

Hagberg, A. A., Schult, D. A., Swart, P. J. (2008). "Exploring network structure, dynamics, and function using NetworkX." *Proceedings of the 7th Python in Science Conference (SciPy2008)*, 11–15. (Pasadena, CA USA).

Hale, W. K. (1980). "Frequency assignment: Theory and applications." *Proceedings of the IEEE*. 68(12):1497–1514.

Harris Corporation (2016). "Harris Falcon III AN/PRC-117G(V)1(C) Multiband Networking Manpack Radio." URL https://www.harris.com/sites/default/files/downloads/solutions/harris-falcon-III-an-prc-117g-multiband-networking-manpack-radio.pdf.

Hart, W. E., Laird, C., Watson, J.-P., Woodruff, D. L. (2012). *Pyomo - Optimization modeling in Python*, volume 67. (Springer Science & Business Media).

Hart, W. E., Watson, J.-P., Woodruff, D. L. (2011). "Pyomo: Modeling and solving mathematical programs in Python." *Mathematical Programming Computation*. 3(3):219–260.

Hassan, M. A., Chickadel, A. (2011). "A review of interference reduction in wireless networks using graph coloring methods." *arXiv preprint*. ArXiv:1103.5791.

Hastie, T., Tibshirani, R., Friedman, J. (2001). *The Elements of Statistical Learning*, volume 1. (Springer Series in Statistics).

Hoffman, K. L., Ralphs, T. K. (2013). "Integer and combinatorial optimization." *Encyclopedia of Operations Research and Management Science*, 771–783. (Springer).

Hooker, J. (2011). *Logic-based Methods for Optimization: Combining Optimization and Constraint Satisfaction*, volume 2. (John Wiley & Sons).

Hooker, J. N., Ottosson, G. (2003). "Logic-based Benders decomposition." *Mathematical Programming*. 96(1):33–60.

Hu, J. (2012). *Algorithms for irreducible infeasible subset detection in CSP - Application to frequency planning and graph k-coloring*. Ph.D. thesis, Belfort-Montbéliard.

Hwang, F. K., Richards, D. S., Winter, P. (1992). *The Steiner Tree Problem*. (Elsevier).

IBM (2013a). "CPLEX Optimizer." URL `http://www-01.ibm.com/software/commerce/optimization/CPLEX-optimizer/`.

IBM (2013b). "Numerically Sensitive Data." URL `http://pic.dhe.ibm.com/infocenter/cosinfoc/v12r2/index.jsp`.

IBM (2016). "CPLEX CP Optimizer." URL `http://www-01.ibm.com/software/commerce/optimization/cplex-cp-optimizer/`.

Jaumard, B., Marcotte, O., Meyer, C., Vovor, T. (2002). "Comparison of column generation models for channel assignment in cellular networks." *Discrete Applied Mathematics*. 118(3):299–322.

Johannson, F., et al. (2013). "mpmath: A Python library for arbitrary-precision floating-point arithmetic (version 0.18)." URL `http://mpmath.org`.

Katzela, I., Naghshineh, M. (1996). "Channel assignment schemes for cellular mobile telecommunication systems: A comprehensive survey." *IEEE Personal Communciations*. 3(3):10–31.

Kostakos, V. (2009). "Temporal graphs." *Physica A: Statistical Mechanics and its Applications*. 388(6):1007–1023.

Koster, A. M. C. A. (1999). *Frequency assignment: Models and algorithms*. Ph.D. thesis, Zuse Institute.

Kuhn, H. W. (1955). "The Hungarian method for the assignment problem." *Naval Research Logistics Quarterly*. 2(1-2):83–97.

Lamar, J. (2013). "Northrop Grumman-developed advanced SPEED software released by U.S. Marine Corps." Northrop Grumman press release, URL `http://investor.northropgrumman.com/`.

London, J. P. (2015). "The New Wave of Warfare–Battling to Dominate the Electromagnetic Spectrum." *Journal of Electronic Defense (JED)*. 38(9):68–76.

Longley, A. G., Rice, P. L. (1968). "Prediction of tropospheric radio transmission loss over irregular terrain. A computer method-1968." Technical report, Institute for Telecommunications Sciences.

Mannino, C., Sassano, A. (2003). "An enumerative algorithm for the frequency assignment problem." *Discrete Applied Mathematics*. 129(1):155–169.

Margot, F. (2002). "Pruning in isomorphism in branch-and-cut." *Mathematical Programming*. 94(1):71–90.

Margot, F. (2010). "Symmetry in integer linear programming." *50 Years of Integer Programming 1958-2008*, 647–686. (Springer).

Mehlhorn, K., Thiel, S. (2000). "Faster algorithms for bound-consistency of the sortedness and the alldifferent constraint." *International Conference on Principles and Practice of Constraint Programming*, 306–319. (Springer).

Mehrotra, A., Trick, M. A. (1996). "A column generation approach for graph coloring." *Journal on Computing*. 8(4):344–354.

Metzger, B. (1970). "Spectrum management technique." *38th National ORSA Meeting*.

Modi, P. J., Shen, W.-M., Tambe, M., Yokoo, M. (2005). "ADOPT: Asynchronous distributed constraint optimization with quality guarantees." *Artificial Intelligence*. 161(1):149–180.

Molisch, A. F. (2011). *Wireless Communications*. (John Wiley & Sons).

Monteiro, J., Goldman, A., Ferreira, A. (2006). "Performance evaluation of dynamic networks using an evolving graph combinatorial model." *IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 173–180.

Montemanni, R., Smith, D., Allen, S. M. (2001). "Lower bounds for fixed spectrum frequency assignment." *Annals of Operations Research*. 107(1-4):237–250.

Montemanni, R., Smith, D., Allen, S. M. (2004). "An improved algorithm to determine lower bounds for the fixed spectrum frequency assignment problem." *European Journal of Operational Research*. 156(3):736–751.

Montemanni, R., Smith, D. H., Allen, S. M. (2002). "An ANTS algorithm for the minimum-span frequency-assignment problem with multiple interference." *IEEE Transactions on Vehicular Technology*. 51(5):949–953.

Moy, J. (1998). "Open shortest path first (OSPF) version 2." *IETF: The Internet Engineering Taskforce*. RFC 2328.

Muller, N. J. (2003). *Wireless A to Z*. (McGraw-Hill).

Murphey, R., Pardalos, P., Resende, M. (1999). "Frequency assignment problems." Du, D.-Z., Pardalos, P., eds., *Handbook of Combinatorial Optimization, Supplement Vol. A*. (Kluwer Academic Publishers).

Nicholas, P., Alderson, D. (2012). "Fast, Effective Transmitter Placement in Wireless Mesh Networks." *Military Operations Research*. 17(4):69–84.

Nicholas, P., Tkacheff, J., Kuhns, C. (2016). "Measuring the operational impact of military SATCOM degradation." *Proceedings of the Winter Simulation Conference*, to appear.

Nicholas, P. J. (2016). "Optimal allocation of electromagnetic spectrum to support tactical wideband communications." *Military Operations Research*. 21(1):21–39.

Nicholas, P. J., Hoffman, K. L. (2015). "Computational challenges of dynamic channel assignment for military MANET." *Proceedings of the Military Communications Conference (MILCOM)*, 1150–1157. (IEEE).

Nicholas, P. J., Hoffman, K. L. (2016). "Optimal channel assignment for military MANET using integer optimization and constraint programming." *Proceedings of the Military Communications Conference (MILCOM)*. (IEEE), to appear.

Nicholas, P. J., Pepper, J., Hipsher, M., Bulanow, P. (2013a). "MAGTF Wideband Spectrum Requirement and Allocation Study." Technical report, Marine Corps Combat Development Command, Quantico, VA.

Nicholas, P. J., Pepper, J., Weaver, C., Gibbons, D., Muratore, M. (2013b). "Simulation and Analysis of Mobile Ad hoc Network Technology in the U.S. Marine Corps Infantry Battalion." *Military Operations Research*. 18(4):19–35.

Olexa, R. (2004). *Implementing 802.11, 802.16, and 802.20 Wireless Networks: Planning, Troubleshooting, and Operations*. (Elsevier).

Ostrowski, J., Linderoth, J., Rossi, F., Smriglio, S. (2011). "Orbital branching." *Mathematical Programming*. 126(1):147–178.

Palpant, M., Oliva, C., Artigues, C., Michelon, P., Didi Biha, M. (2008). "Models and methods for frequency assignment with cumulative interference constraints." *International Transactions on Operational Research*. 15(3):307–324.

Park, H.-S., Jun, C.-H. (2009). "A simple and fast algorithm for K-medoids clustering." *Expert Systems with Applications*. 36(2):3336–3341.

Pecora, F., Modi, P., Scerri, P. (2006). "Reasoning about and dynamically posting n-ary constraints in ADOPT." *7th International Workshop on Distributed Constraint Reasoning*, volume 2006.

Pemmasani, G. (2016). "dispy: Python framework for distributed and parallel computing." URL `http://dispy.sourceforge.net`.

Phelps, K. T., Rödl, V. (1984). "On the algorithmic complexity of coloring simple hypergraphs and Steiner triple systems." *Combinatorica*. 4(1):79–88.

Poisel, R. (2011). *Modern Communications Jamming: Principles and Techniques*. (Artech House).

Puget, J.-F. (1998). "A fast algorithm for the bound consistency of alldiff constraints." *Proceedings of the Fifteenth National Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, 359–366.

Ramani, A., Aloul, F. A., Markov, I. L., Sakallah, K. A. (2004). "Breaking instance-independent symmetries in exact graph coloring." *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*, volume 1, 324–329. (IEEE).

Refalo, P. (2004). "Impact-based search strategies for constraint programming." *International Conference on Principles and Practice of Constraint Programming*, 557–571. (Springer).

Régin, J.-C. (1994). "A filtering algorithm for constraints of difference in CSPs." *Proceedings of the Twelfth AAAI National Conference on Artificial Intelligence*, volume 94, 362–367.

Riihijärvi, J., Petrova, M., Mähönen, P. (2005). "Frequency allocation for WLANs using graph colouring techniques.." *Proceedings of the Wireless On-demand Network Systems and Services Conference (WONS)*, volume 5, 216–222.

Rossi, F., Van Beek, P., Walsh, T. (2006). *Handbook of Constraint Programming*. (Elsevier).

Sarzeaud, O., Berny, A. (2003). "Allocation de fréquences par échantillonnage de gibbs, recuit simulé et apprentissage par renforcement." *5ème congrès de la société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF)*. 116–117.

Scellato, S., Leontiadis, I., Mascolo, C., Basu, P., Zafer, M. (2013). "Evaluating temporal robustness of mobile networks." *IEEE Transactions on Mobile Computing*. 12(1):105–117.

Selyukh, A. (2013). "In switch, U.S. military offers to share airwaves with industry." URL `http://www.reuters.com/article/usa-defense-spectrum-idUSL1N0FT0KG20130723`.

Skalli, H., Ghosh, S., Das, S. K., Lenzini, L., Conti, M. (2007). "Channel assignment strategies for multiradio wireless mesh networks: Issues and solutions." *IEEE Commununications Magazine*. 45(11):86–95.

Skiena, S. (1990). *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematics*. (Reading, MA).

Smith, D. H., Taplin, R. K., Hurley, S. (2001). "Frequency assignment with complex co-site constraints." *IEEE Transactions on Electromagnetic Compatibility*. 43(2):210–218.

Sridhar, S., Guo, J., Jha, S. (2009). "Channel assignment in multi-radio wireless mesh networks: A graph-theoretic approach." *2009 First International Communication Systems and Networks and Workshops*, 1–10. (IEEE).

Stine, J., Portigal, D. (2004). "Spectrum 101: An introduction to spectrum management." Technical Report MTR 04W0000048, The MITRE Corporation.

Ståhlberg, M. (2000). "Radio jamming attacks against two popular mobile networks." Technical report, Helsinki University of Technology Seminar on Network Security.

Subramanian, A. P., Gupta, H., Das, S. R., Cao, J. (2008). "Minimum interference channel assignment in multi-radio wireless mesh networks." *IEEE Transactions on Mobile Computing*. 7(12):1459–1473.

Sung, C. W., Wong, W. S. (1997). "Sequential packing algorithm for channel assignment under cochannel and adjacent-channel interference constraint." *IEEE Transactions on Vehicular Technology*. 46(3):676–686.

Tiourine, S., Hurkens, C., Lenstra, J. K. (1995). "An overview of algorithmic approaches to frequency assignment problems." *Proceedings of CALMA Symposium*.

Tomita, E., Tanaka, A., Takahashi, H. (2006). "The worst-case time complexity for generating all maximal cliques and computational experiments." *Theoretical Computer Science*. 363(1):28–42.

Tseng, Y.-C., Chao, C.-M., Wu, S.-L., Sheu, J.-P. (2002). "Dynamic channel allocation with location awareness for multi-hop mobile ad hoc networks." *Computer Communications*. 25(7):676–688.

Van Hentenryck, P. (1999). *The OPL Optimization Programming Language*. (MIT Press).

Vlasak, J., Vasquez, M. (2003). "Résolution du problème d'attribution de fréquences avec sommation de perturbateurs." *5ème congrés de la société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF)*. 118–119.

Voudouris, C., Tsang, E. (1998). "Solving the radio link frequency assignment problem using guided local search." *Proceedings of the NATO Symposium on Radio Length Frequency Assignment*.

Wang, S.-W., Rappaport, S. S. (1989). "Signal-to-interference calculations for balanced channel assignment patterns in cellular communications systems." *IEEE Transactions on Communications*. 37(10):1077–1087.

Whitbeck, J., Dias de Amorim, M., Conan, V., Guillaume, J.-L. (2012). "Temporal reachability graphs." *Proceedings of the International Conference on Mobile Computing and Networking*, 377–388. (ACM).

Wolsey, L. A., Nemhauser, G. L. (2014). *Integer and Combinatorial Optimization*. (John Wiley & Sons).

Wu, Y., Wang, B., Liu, K. R., Clancy, T. C. (2012). "Anti-jamming games in multi-channel cognitive radio networks." *IEEE Journal on Selected Areas in Communications*. 30(1):4–15.

Xu, R., Wunsch, D., et al. (2005). "Survey of clustering algorithms." *IEEE Transactions on Neural Networks*. 16(3):645–678.

Yeoh, W., Felner, A., Koenig, S. (2008). "BnB-ADOPT: An asynchronous branch-and-bound DCOP algorithm." *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*, volume 2, 591–598.

Yu, F., Bar-Noy, A., Basu, P., Ramanathan, R. (2013). "Algorithms for channel assignment in mobile wireless networks using temporal coloring." *Proceedings of 16th ACM International Conference on Modeling, Analysis & Simulation of Wireless and Mobile Systems*, 49–58.

Zhang, Y., Luo, J., Hu, H. (2006). *Wireless Mesh Networking: Architectures, Protocols and Standards*. (CRC Press).

Zhao, J., Zheng, H., Yang, G.-H. (2005). "Distributed coordination in dynamic spectrum allocation networks." *IEEE International Symposium on Dynamic Spectrum Access Networks*, 259–268. (IEEE).

Zhao, Q., Sadler, B. M. (2007). "A survey of dynamic spectrum access." *IEEE Signal Processing Magazine*. 24(3):79–89.

Zhou, B., Xu, K., Gerla, M. (2004). "Group and swarm mobility models for ad hoc network scenarios using virtual tracks." *IEEE Military Communications Conference*, volume 1, 289–294. (IEEE).

Biography

Paul J. Nicholas graduated from the U.S. Naval Academy in 2003 with a Bachelor of Science degree in Systems Engineering, and from the Naval Postgraduate School in 2009 with a Master of Science degree in Operations Research. Paul has served as an active duty and Reserve Marine Corps officer since 2003, and has completed multiple deployments to Iraq and Afghanistan. He currently works as a civilian operations research analyst at the Marine Corps Operations Analysis Directorate, and as the Cyberspace Network Operations Officer at the Marine Corps Information Operations Center, both in Quantico, Virginia. He also teaches a course on analytics and decision analysis at George Mason University.