A SCHEDULING MODEL FOR CENTRALIZED COGNITIVE RADIO NETWORKS

by

Didem Gözüpek

B.S., Telecommunications Engineering, Sabancı University, 2004

M.S., Electrical Engineering, New Jersey Institute of Technology, 2005

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Computer Engineering

Boğaziçi University

2012

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor Assoc. Prof. Fatih Alagöz for his invaluable guidance and encouragement throughout these years. He has been much more than an advisor, more like an elder brother to me. I am also extremely grateful to Assist. Prof. Mordechai Shalom for introducing me to the exciting fields of algorithmic graph theory and approximation algorithms. His clever comments and solid theoretical background have tremendously increased the quality of this thesis. In particular, Chapter 6 of this thesis is an outcome of our collaborative work. I feel extremely lucky to have met him not only because of his contribution to my thesis but also for his role in my future career and research interests. I hope to continue working with him throughout the rest of my professional life.

I would like to thank my jury members Assoc. Prof. Tuna Tuğcu, Assist. Prof. Tınaz Ekim, and Prof. Özgür Barış Akan for their time and insightful comments. Their remarks have significantly improved the quality of this thesis.

I would like to thank Dr. Seyed Buhari for our fruitful research collaboration, which led to Chapter 8 of this thesis. I am grateful to Prof. Shmuel Zaks, Ariella Voloshin, and Assist. Prof. Mordechai Shalom from Technion, Israel for their hospitality and making my short stay in Haifa a very fruitful one.

I was honored and priveleged to receive "ASELSAN PhD Fellowship" award to support my dissertation during the final year of my Ph.D. study. I believe the fellowship constitutes the foundation of a long standing collaboration with ASELSAN for the upcoming years of my professional life.

My PhD years would definitely not be so enjoyable without the wonderful social environment of our department. My deep thanks especially go to the SATLAB community: Gaye & Yunus Emre, Birkan, Derya, Suzan, Şükrü, İlker, Gürkan, Seçkin, and Salim. Our picnics, birthday parties, breakfasts, coffee breaks, and loud chats are defi-

nitely the moments I am really going to miss. Thank you all for your great friendship, especially the SATLAB women Derya, Gaye, and Suzan. I also would like to thank all other members of the CMPE community: Akın, Nadin, Haşim, Çetin, Neşe, Furkan, Pınar, Tekin, Gül, ... I am sure that the list is incomplete.

I am extremely grateful to my mother Şahsine Gözüpek, my father İsmail Gözüpek, and my sister and best friend Sinem Gözüpek for their constant love and support in my life. I always feel like the luckiest person in the world to have such a great family. I would not be at where I am today if it was not for my family. Last but definitely not the least, I am deeply indebted to my husband Erdinç Kocaman for his continuous support and magically changing my life since the day we met. This thesis is dedicated to these great four people.

This thesis has also been supported by the State Planning Organization of Turkey (DPT) under grant number DPT-2007K 120610 and Scientific and Technological Research Council of Turkey (TUBITAK) 2211 National PhD Scholarship.

ABSTRACT

A SCHEDULING MODEL FOR CENTRALIZED COGNITIVE RADIO NETWORKS

In this thesis, we present a scheduling model for centralized cognitive radio networks. Our model consists of a set of schedulers that focus on the data transmission of the secondary users and determine with which frequency, time slot and data rate each secondary user will transmit to the cognitive base station. Common features of the schedulers are that all of them ensure that the primary users in the service area of the cognitive base station are not disturbed, no collisions occur among the secondary users, and reliable communication of the secondary users with the cognitive base station is maintained. Our schedulers differ from each other mainly in terms of their objectives. We propose schedulers that maximize the overall cognitive radio cell throughput, minimize the average scheduling delay of the secondary users, provide max-min, weighted max-min and proportional throughput fairness, maximize the number of secondary users that are satisfied in terms of throughput, and take the different delay costs of switching to different frequency bands into account. In addition to heuristic algorithms and simulation based studies, we also present a graph theoretic approach and prove several NP-hardness and inapproximability results, propose polynomial time graph algorithms as well as approximation algorithms.

ÖZET

MERKEZİ BİLİŞSEL RADYO AĞLARI İÇİN BİR ÇİZELGELEME ÇATISI

Bu tezde merkezi bilişsel radyo ağları için bir çizelgeleme modeli öneriyoruz. Modelimiz ikincil kullanıcıların veri iletimine odaklanan ve merkezi bilişsel baz istasyonuna hangi frekans, zaman dilimi ve veri hızıyla iletim yapacaklarını belirleyen çizelgeleyiciler kümesinden oluşmaktadır. Çizelgeleyicilerin ortak özellikleri merkezi bilişsel baz istasyonunun hizmet alanı içindeki birincil kullanıcıların rahatsız olmamalarını, ikincil kullanıcılar arasında çarpışma olmamasını ve ikincil kullanıcılar ile bilişsel baz istasyonu arasındaki iletişimin itimat edilebilir olmasını garanti etmeleridir. Çizelgeleyicilerimiz birbirlerinden temel olarak amaç fonksiyonlarıyla ayrılmaktadır. Hücredeki ikincil kullanıcıların toplam iş oranını azamileştiren, ikincil kullanıcıların çizelgeleme gecikmesini asgarileştiren, azami-asgari, ağırlıklı azami-asgari ve orantısal açıdan adillik sağlayan, iş oranı açısından tatmin olan ikincil kullanıcı sayısını azamileştiren ve farklı frekans bantlarına geçişin farklı gecikme maliyetlerini dikkate alan çizelgeleyiciler öneriyoruz. Buluşsal algoritmalara ve benzetim çalışmalarına ek olarak aynı zamanda çizge teorisi tabanlı bir yaklaşım öneriyor, NP-zorluk ve yaklaşıklanamama sonuçları ispatlıyor ve polinom zamanlı çizge algoritmaları ile yaklaşıklama algoritmaları öneriyoruz.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

a	Number of antennas of SUs
B	Bandwidth
F	Number of frequencies
$\mathcal{F}$	The set of F frequencies
$\mathbf{I}$	A function associating an interval of natural numbers with each vertex in a graph
N	Number of secondary users
$\mathcal{N}$	The set of N secondary users
M	Number of primary users
R^i_φ	Aggregate average throughput of secondary user i in the last φ scheduling periods
T	Number of time slots in a scheduling period
$\mathcal{T}$	The set of T time slots in a scheduling period
T_s	Time slot length
U_{if}	The maximum number of packets that can be sent by secondary user i using frequency f in every time slot during the scheduling period
V_p	Velocity of primary users
V_s	Velocity of secondary users
β	Switching delay in terms of milliseconds for each unit step in the frequency range
Δ_{iaft}	The absolute difference in terms of the number of frequencies that antenna a of secondary user i has to step to use frequency f in time slot t
η_i	Target throughput proportion (weight) of secondary user i
μ_m	Mutation rate
σ^2	Noise variance

φ	Window size (number of scheduling periods) during which the changes in the network conditions are considered to be important
Φ_{CBS}^{ft}	The set of primary users that are actively utilizing frequency f in the coverage area of the CBS in time slot t
Ω_i^{min}	The minimum throughput requirement of secondary user i

LIST OF ACRONYMS/ABBREVIATIONS

ANOVA	Analysis of Variation
AWGN	Additive White Gaussian Noise
CBS	Cognitive Base Station
CCC	Common Control Channel
CI	Confidence Interval
CR	Cognitive Radio
CRN	Cognitive Radio Network
FCC	Federal Communications Commission
GA	Genetic Algorithm
ILP	Integer Linear Program
IT	Interference Temperature
LBAP	Linear Bottleneck Assignment Problem
LP	Linear Program
MFS	Maximum Frequency Selection
MKP	Multiple Knapsack Problem
MMFS	Max-Min Fair Scheduling
MNSU	Maximizing the Number of Satisfied Users
PFS	Proportionally Fair Scheduler
PLL	Phase Locked Loop
PTAS	Polynomial Time Approximation Scheme
PU	Primary User
RWMM	Random Waypoint Mobility Model
S^2DASA	Spectrum Switching Delay Aware Scheduler
SU	Secondary User
TMS	Throughput Maximizing Scheduling
VCO	Voltage Controlled Oscillator

1. INTRODUCTION

Research studies exhibit that spectrum is sparsely utilized in some frequency bands, whereas it is overcrowded in other frequency bands [1]. The escalating demand for the radio spectrum driven by the continuous growth of wireless technologies and services necessitates new methods to combat the acute shortage of bandwidth by utilizing the spectrum more efficiently. In this respect, dynamic spectrum access (DSA) methods that enable the devices to opportunistically access the licensed frequency bands have been proposed. Cognitive radios, which are computationally intelligent devices that can sense the environment and adapt their communication parameters in accordance with the network and user demands, are able to realize the DSA methodology [2].

A cognitive radio network (CRN) consists of primary users (PU) and secondary users (SU). The former is a licensed user and hence has exclusive rights to access the radio spectrum, whereas the latter is an unlicensed user that can opportunistically access the temporarily unused licensed spectrum bands, provided that it vacates them as soon as the PU of that particular band appears [3]. In the rest of this thesis, we use the terms *cognitive users* and *secondary users* interchangeably. In Chapter 2, we discuss related work corresponding to each chapter of this thesis.

Federal Communications Commission (FCC) has proposed a new model, referred to as interference temperature (IT) model [4], that enables true coexistence between licensed and unlicensed users. In this model, SUs are permitted to simultaneously operate on the same frequencies as the PUs provided that the interference perceived by the PUs is within predefined acceptable limits, quantified by the interference temperature threshold for that particular frequency. In Chapter 3, we formulate throughput and delay optimal schedulers for cognitive radio networks under interference temperature constraints. Furthermore, we also propose two suboptimal schedulers, referred to as *maximum frequency selection (MFS)* and *probabilistic frequency selection (ProbFS)*. To the best of our knowledge, ours is the first study on scheduling in cognitive radio networks meeting the interference temperature constraints of the PUs.

Suboptimal schedulers presented in Chapter 3 have low computational complexity; however, their throughput and delay performance is not satisfactory. Hence, the design of better performing, yet computationally efficient schedulers is important. We address this issue in Chapter 4 and propose two genetic algorithm (GA) based suboptimal schedulers that yield very close performance to the values obtained from the optimization software CPLEX [5]. To the best of our knowledge, the use of GAs for scheduling in cognitive radio networks has not previously been explored.

IT model incurred a lot of debate since its proposition by FCC. Finally, FCC abandoned the IT concept [6]. From Chapter 5 onwards, we distance ourselves from the IT debate and rely on a much simpler physical layer model, which can be actualized using conventional physical and MAC layer spectrum sensing mechanisms in the CRN literature [7, 8]. In this context, we formulate in Chapter 5 throughput maximizing, max-min fair, weighted max-min fair, and proportionally fair schedulers for centralized cognitive radio networks, where the SUs possibly have multiple antennas for data transmission. The major merit of our proposed scheduling scheme is that it is a very general model accomplishing many tasks at the same time such as frequency, time slot, data rate, and power allocation in a heterogeneous multi-user and multi-channel environment. Another distinctive feature of our fair schedulers is that they take the throughput values experienced by the SUs in the recent past into account and use this information in the current scheduling decision. We also propose heuristic algorithms for our fair schedulers and evaluate their performance through simulations.

We present in Chapter 6 a graph theoretic approach to the throughput maximizing, max-min fair, weighted max-min fair, and proportionally fair schedulers formulated in Chapter 5. First, we propose a polynomial time algorithm for the throughput maximizing scheduling problem. We then elaborate on certain special cases of this problem and explore their combinatorial properties. Second, we prove that the max-min fair scheduling problem is NP-Hard in the strong sense. We also prove that the problem cannot be approximated within any constant factor better than 2 unless $P = NP$. Additionally, we propose an approximation algorithm for the max-min fair scheduling problem with approximation ratio depending on the maximum possible data rates of

the secondary users. We then focus on the combinatorial properties of certain special cases and investigate their relation with various problems such as the multiple knapsack, matching, terminal assignment, and Santa Claus problems. We then prove that the proportionally fair scheduling problem is NP-Hard in the strong sense and cannot have an approximation algorithm with absolute approximation ratio smaller than $\log(\frac{4}{3})$. Our graph theoretic approach sheds light on the complexity and combinatorial properties of our proposed throughput maximizing and fair scheduling problems.

We consider in Chapter 7 a scenario where each SU has a possibly different minimum throughput requirement below which the SU is not satisfied. The value of this minimum throughput requirement may depend on the application executed by the SU. For instance, delay sensitive real-time applications usually necessitate a higher minimum data rate. The goal of the scheduler proposed in Chapter 7 is to maximize the number of SUs that are satisfied in terms of throughput. We prove that this problem is NP-Hard in the strong sense and cannot be approximated within any constant factor better than 2 unless $P = NP$. We also prove that this problem is at least as hard as the max-min fair scheduling problem, which has been formulated in Chapter 5 and proved in Chapter 6 to be a computationally very difficult problem. Heuristic algorithms for the throughput satisfaction based scheduling problem in this chapter have been developed in [9].

When a cognitive radio (CR) device changes its operation frequency, it experiences a hardware switching delay to tune to its new frequency before it can fully utilize it. This delay in general depends on the difference between the two frequency bands; i.e., switching from central frequency of 800 MHz to 10 GHz conduces larger delay than switching from 800 MHz to 850 MHz due to the hardware capabilities of the frequency synthesizer. We formulate in Chapter 8 a scheduling problem that takes into account the different hardware delay experienced by the secondary users while switching to different frequency bands and propose a polynomial time suboptimal algorithm. To the best of our knowledge, ours is the first study on scheduling in CRNs that takes into account the hardware switching delay depending on the separation between the current and subsequent frequency bands.

2. RELATED WORK

2.1. Scheduling in Cognitive Radio Networks

Fundamental problems of scheduling schemes have been extensively studied in conventional wireless networks [10–13]; however, the emergence of new concepts like cognitive radio brings this topic into the focus of research again. The cognitive radio paradigm introduces new challenges to the scheduling schemes since the varying channel availability due to coexistence with PUs requires the cognitive users to determine when and on which channel they should tune to in order to exchange data with their neighbors.

Works about scheduling in CRNs can be broadly categorized into two groups: The ones about overlay spectrum sharing paradigm and the ones about underlay spectrum sharing paradigm. Overlay spectrum sharing treats the availability of a frequency band as a binary decision; i.e., either the band is available for SU transmission or not, whereas underlay spectrum sharing is based on the idea of ensuring that the interference experienced by a PU is below a tolerable upper limit. The proposed scheduling model in this thesis falls into the underlay spectrum sharing category.

The work in [14], for instance, concentrates on the spectrum overlay paradigm by modeling the interference as a multi-channel contention graph (MCCG). Authors in [15] propose a spectrum decision framework for centralized CRNs by considering application requirements and current spectrum conditions. Unlike our work, the work in [16] focuses on inter-cell spectrum sharing and propose a joint spectrum and power allocation framework. Both [15] and [16] are based on the spectrum overlay paradigm since they handle spectrum availabilities as a binary decision, i.e., a spectrum band is either available or not.

The integer linear programming (ILP) formulation for the MAC-layer scheduling introduced in [17] minimizes the schedule length in multi-hop cognitive radio networks.

They also propose a distributed heuristic to determine the channels and time slots for the cognitive nodes. However, both in their optimization formulation and the suboptimal heuristic, they do not consider the interference to the PUs.

The optimization problem formulated by the work in [18] considers interference constraints and channel heterogeneity, which implies that different channels support different transmission ranges. Our work in this thesis also has this channel heterogeneity feature since we model the maximum transmission power of different frequencies with respect to the maximum tolerable interference values in the CRN cell. Different maximum transmit power values imply different transmission ranges for the frequencies. However, our work is different from [18] in numerous ways. First, we focus on a *spectrum underlay* model, where SUs transmit at the same time and frequency with the PUs while adhering to the tolerable interference limits. In contrast, authors in [18] focus on a *spectrum overlay* model, where SUs opportunistically utilize the spatio-temporally unoccupied portions of the spectrum. Second, they base their model on an ad hoc cognitive radio network architecture, while we focus on a centralized cognitive radio network. Third, they consider only frequency domain channel assignment, whereas we consider both frequency and time domain channel assignment; i.e., our proposed models determine the assignment of both time slots and frequencies to the SUs. Fourth, unlike the schedulers in this thesis, the objective function in [18] maximizes the total spectrum utilization. Their objective function tries to establish as many links between the SUs as possible. In other words, all frequencies have the same weight in their work, whereas there is a maximum rate constraint for each frequency in our work. Fifth, they use a simple linear expression for the relation between the transmission range and interference range, while we guarantee the reliable communication with the base station and consider the tolerable interference values.

The works in [19–22] also focus on the spectrum overlay paradigm. Our work in this thesis is distinct in principle from all works in the literature about overlay spectrum sharing because we do not consider interference as a binary decision. In particular, we take into account the interference temperature constraints in Chapter 3 and maximum allowed interference power at PUs in Chapter 5. Hence, our work in this thesis falls

into the underlay spectrum sharing category.

2.2. Interference Temperature Concept

Studies on the interference temperature concept mainly revolve around methods that optimize various objectives such as QoS, transmission power allocation or channel capacity subject to the interference temperature constraints. For instance, authors in [23] provide an analysis of the achievable capacity by the interference temperature model. They model the RF environment and derive the probability distributions governing the interference temperature.

The work in [24] formulates a nonlinear social rate optimization problem with QoS and interference temperature constraints. Note that unlike our work, their problem does not consider the frequencies that each SU will be using but only determines the rate and transmission power of the users. Hence, unlike our study, the work in [24] is not a scheduling issue that determines the frequency and time slot allocation of the cognitive users. Moreover, they only consider a single interference temperature measurement point, whereas we satisfy in Chapter 3 the constraints in all the measurement points.

Authors in [25] concentrate on the power control problem in cognitive radio networks under interference temperature constraints. They firstly examine the power control problem without interference temperature constraints. Subsequently, they reformulate the same problem by taking interference temperature constraints into account and model it as a concave minimization problem with linear constraints.

The study in [26] considers the interference temperature model from a different perspective. Binary and transmitter-centric constraints are often used in the literature, where a reuse distance between pair-wise sets of transmitters are considered and the reuse of a set of channels is prohibited within this reuse distance. On the contrary, the work in [26] proposes non-binary and receiver-centric constraints, where the aggregate interference at the receiver is considered and multiple transmitters are allowed to use

the same set of channels as long as they satisfy the interference temperature limit at the receiver.

To the best of our knowledge, our work in Chapter 3 is the first study on scheduling in cognitive radio networks under interference temperature constraints.

2.3. Genetic Algorithms in Cognitive Radio Networks

The usage of genetic algorithms (GA) has been proven to be quite successful for channel assignment schemes in cellular networks [27, 28]. In cognitive radio networks context, on the other hand, the studies about the usage of GAs mainly revolve around the configuration of various cognitive radio parameters such as pulse shape, symbol rate, and modulation. Authors in [29] present GA-based adaptive component of the cognitive radio engine developed at the Virginia Tech Center for Wireless Telecommunications (CWT). They firstly formulate a multi-objective optimization problem, and then evaluate the fitness function for this overall problem as the weighted sum of the fitness values of each objective. In line with this formulation, they define the chromosome structure as consisting of power, frequency, pulse shape, symbol rate, and modulation.

The cognitive radio software testbed discussed in [30] includes a cognitive radio engine based on GAs. The engine executes two separate GAs to select the channel and transmission parameters, each of which is a multi-objective problem similar to the one in [29].

Authors in [31] formulate the channel assignment problem specific to cognitive radio networks and propose an island GA, in which the population is divided into subpopulations called islands and the chromosomes interact through migration to other islands. The channel assignment problem that they consider determines which channel (frequency) to assign for which communication link; hence, it is a static one-shot assignment procedure. In our work, in contrast, we focus on the scheduling problem rather than the channel assignment problem. Therefore, unlike the work in [31], we

also have a time aspect; i.e., the problems we focus on determine both the frequencies and the time slots to assign to the SUs.

The population adaptation technique introduced in [32] again considers cognitive radio engines and devises a method to expedite the convergence of the GAs. Their method is based on having the cognitive engine to utilize the information about the wireless environment learned in the previous cognition cycles and seeding the initial generation of the GA with high scoring chromosomes from the previous run. Their results demonstrate that this approach performs quite well in slowly varying wireless environments but yields poor results when the conditions are changing fast.

2.4. Fair Scheduling Algorithms in Cognitive Radio Networks

Opportunistic scheduling exploits the time-varying channel conditions in wireless networks to increase the overall performance of the system. It has received a lot of attention in the general wireless networking domain [11, 33, 34]. A scheme designed only to maximize the overall throughput can be unfairly biased, especially when there are users with persistently bad channel conditions. Therefore, maintaining some notion of fairness, such as max-min and proportional fairness, is a vital criterion that opportunistic scheduling algorithms should address.

There are various efforts in the literature that address fairness in the general domain of wireless networks. Authors in [35] propose a set of algorithms for the assignment of max-min fair shares to nodes in a wireless ad hoc network. The work in [36] proposes schedulers that provide deterministic and probabilistic fairness by decoupling throughput optimization and fairness guarantees as two distinct blocks. Authors in [37] propose a fair framework for ad hoc wireless networks via a contention resolution algorithm. However, none of these works takes into account the unique features of the CRN concept such as the requirement to ensure PU protection; therefore, they are unsuitable for implementation in CRNs.

The opportunistic nature of the CRN concept lends itself conveniently to the

opportunistic scheduling paradigm. In CRN context, not only the physical channel conditions such as fading and path loss, but also the PU activity is a determinant of the channel quality of an SU. This way, an opportunistic scheduler provides the ability to opportunistically utilize from the time-varying PU activity. Likewise, a purely opportunistic scheduler may cause an SU that persistently has an active PU in its vicinity starve in terms of throughput. Thus, providing fairness again comes into play here to compensate for the throughput losses of the SUs that have active PUs in their surroundings. In addition to time, frequency is also a resource that needs to be shared fairly among the SUs in CRNs. Unlike the works in [11, 33–37], our focus in Chapter 5 is specifically on cognitive radio networks. In particular, we formulate a scheduling scheme that utilizes the opportunistic nature of the CR paradigm as well as taking fairness notions into account.

Most of the scheduling works about underlay spectrum sharing in the literature focus on rate and power allocation without emphasizing other criteria such as frequency allocation, possibly having multiple antennas, ensuring reliable communication, and joint temporal/throughput fairness. The work in [38] proposes a joint scheduling and power control scheme for centralized CRNs. Authors in [39] focus on capacity maximization in multi-hop CRNs. None of the works in [38,39] focus on issues such as fairness or multiple antennas. The major difference of our work in Chapter 5 from these works is that we provide a much more general and complete scheduling model that achieves frequency, time slot, data rate, and power allocation, while at the same time accommodating multiple antennas for data transmission, a heterogenous multiuser and multi-channel environment, taking into account numerous physical layer information such as fading, path loss, mobility, and time-varying channels in addition to ensuring that SUs maintain a reliable communication and PUs are not disturbed by SU transmissions. Moreover, we also have a temporal notion of fairness (by ensuring that each SU is assigned at least one time slot) in addition to throughput fairness. Previous works in the literature [14, 38–41] do not concurrently allow these two different notions of fairness. Furthermore, our schedulers have a windowing mechanism that can be tailored by the CBS at her own discretion to increase the total throughput by allowing a little deviation from target throughput ratios of the SUs in our weighted

max-min fair scheduler. To the best of our knowledge, none of the previous work in the literature encompasses all of these features.

2.5. Graph Theoretic Approaches

Graph theoretic techniques have previously been used for peer-to-peer networks [42] and wavelength division multiplexing (WDM) networks [43]. Some scheduling problems in wireless networks have also been addressed by using graph theoretic techniques [44, 45]. On the other hand, graph theoretic approaches in CRNs are mainly based on simple variants of graph coloring problems. Authors in [46] formulate a spectrum allocation problem considering the different channel availabilities at different nodes in a CRN and show that it is a list coloring problem. The work in [47] also addresses dynamic spectrum allocation problem using list coloring and proposes centralized and decentralized suboptimal algorithms. Moreover, authors in [40] reduce their spectrum allocation problems to a variant of graph coloring problem. The authors in [48] also focus on centralized and distributed DSA problem in CRNs by proving NP-hardness of their problem and presenting an approximation algorithm. To the best of our knowledge, very few works that provide a comprehensive graph theoretic approach for scheduling in CRNs exist in the literature. Moreover, unlike most work in the literature that use graph coloring arguments, we use in Chapter 6 various techniques from matching theory as well as relating our scheduling problems to numerous combinatorial optimization problems such as knapsack, terminal assignment, generalized assignment, partition, and Santa Claus problems.

2.6. Throughput Satisfaction Based Scheduling

Maximizing user satisfaction has been addressed by works about scheduling in various areas of wireless networks. For instance, authors in [49] propose a resource allocation algorithm to maximize user satisfaction in OFDMA systems. Their algorithm assigns subcarriers to the users in an OFDMA system while maximizing the number of satisfied users, where a user is satisfied if the average data rate it experiences is greater than or equal to its average data rate requirement. Their two-step algorithm provides

a suboptimal solution to their formulated problem. Authors in [50] also address users' satisfaction in OFDMA albeit with a different approach. Instead of formulating a different optimization problem, they propose a dynamically configurable framework that combines maximum rate, max-min fairness, and proportional fairness policies to maximize the number of satisfied users. Besides, the scheduling approach proposed by the authors in [51] aims to maximize the number of satisfied users in a multirate wireless system. Since their formulated problem is NP-complete, they present an approximation algorithm based on two phase combinatorial reverse auction.

Users' satisfaction has also been addressed in the realm of CDMA networks. Authors in [52] formulate a resource allocation framework for CDMA cellular networks supporting multimedia services. A user's utility characterizes its degree of satisfaction (happiness) with the received service. By using network utility maximization theory, they propose a method to dynamically adapt the users' utility functions. Additionally, authors in [53] aim at maximizing users' utilities in CDMA networks by considering the probabilistic short term throughput requirements of real-time services and long term throughput requirements of non real-time services. They demonstrate that their joint scheduling approach yields substantial performance improvements.

Maximizing user satisfaction in terms of throughput has received little attention in CRNs. Authors in [54] provide a scheduling mechanism that maximizes the sum of utility functions of the source flow rates of origin-destination pairs in an ad hoc CRN. Unlike our work in Chapter 7, they concentrate on the scheduling of flows rather than packets. Moreover, their focus is on ad hoc CRNs, whereas we concentrate on centralized CRNs. Furthermore, our scheduling mechanism not only determines data rates, but also frequencies and time slots. The work in [55], on the other hand, derives opportunistic scheduling policies with utilitarian fairness for OFDM systems. Their methods are specifically designed for OFDM systems. Furthermore, authors in [56] focus on a cooperation-based spectrum leasing scenario where they maximize the total expected utility while satisfying an average performance constraint for each primary node in the network. Unlike our work, their focus is on cooperation and time slot is the only resource that needs to be shared between SUs and PUs. Furthermore, PUs

and SUs in their scenario communicate with a central entity, whereas in our work only SUs communicate with a central entity (CBS).

To summarize, existing opportunistic scheduling algorithms in CRNs do not aim to maximize the number of SUs whose minimum throughput requirements have been met. Therefore, we address this open research issue in Chapter 7 as part of our scheduling model.

2.7. Spectrum Switching Delay

Some works in the literature use the term *channel switching latency* to refer to the delay encountered while searching for an idle channel [57], whereas some other works [58] use the term to refer to the hardware switching delay of the frequency synthesizer given that the CR device has already determined the idle channel to switch to. Our focus in Chapter 8 is on the latter definition of the term.

Channel switching delay in CRNs is mostly considered in the realm of routing [58–63]. The primary goal in most of these works [60, 63] is to minimize the number of channel switchings along the route; hence, they do not differentiate between switching to different frequencies and assume that all of the channel switchings cause a certain delay irrespective of the frequency separation distance. Only a few of these works about routing [58, 59] consider the possibly different delays depending on the wideness between the frequency bands. Besides, current works about scheduling and channel assignment in CRNs [17, 22, 48, 64, 65] do not take the spectrum switching delay into account. To the best of our knowledge, our work in Chapter 8 is the first study on scheduling in CRNs that takes into account the hardware switching delay depending on the separation between the current and subsequent frequency bands.

3. THROUGHPUT AND DELAY OPTIMAL SCHEDULERS UNDER INTERFERENCE TEMPERATURE CONSTRAINTS

3.1. Problem Formulation

We propose in this chapter throughput and delay optimal schedulers for centralized CRNs under interference temperature constraints. Contents of this chapter appeared in [66].

Interference temperature is the temperature equivalent of the RF power available at a receiving antenna per unit of bandwidth. It is formally defined as [67]:

$$IT(f^c, B) = \frac{P_{IF}(f^c, B)}{kB} \tag{3.1}$$

where $IT(f^c, B)$ is the interference temperature for channel c with central frequency f^c and bandwidth B, $P_{IF}(f^c, B)$ is the average interference power in Watts centered at frequency f^c and covering the bandwidth B in Hertz, and k is Boltzmann's constant (1.38×10^{-23} Joules/Kelvin). Under this model, a channel is available at a cognitive node m if the transmission due to m does not increase the interference temperature at any other primary node in the interference range of m beyond a predefined threshold. This constraint can be conveyed as [67]:

$$IT(f^c, B) + \frac{L^c_{mn} P_m(f^c, B)}{kB} < IT^{th}_c \tag{3.2}$$

In the above formulation, L^c_{mn} refers to the distance dependent path loss in transmission from node m to n on channel c, $P_m(f^c, B)$ is the transmission power of m, and IT^{th}_c is the interference temperature threshold for channel c. The threshold values should be determined by the regulatory bodies for each frequency band in a given geographic region.

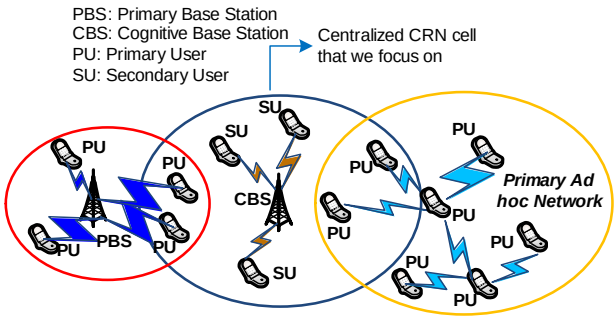


Figure 3.1. The considered centralized CRN architecture.

We consider a time slotted IEEE 802.22 system, where the cognitive devices are managed by the cognitive base station (CBS) [68], which is aware of the DSA concept and has cognitive capabilities. Figure 3.1 illustrates the considered network architecture. The scheduler resides at the CBS and determines how many packets and with which frequency each cognitive user will transmit in each time slot. If we denote the number of packets in the buffer of SU i at the beginning of time slot t by $x_{i,t}$, the number of packets transmitted by SU i in time slot t by $u_{i,t}$, the fading coefficient of the channel between user i and the CBS in time slot t by $A_{i,t}$, the frequency used by user i in time slot t by $f_{i,t}$, the vector of buffer states for a total number of N cognitive nodes as $\overline{x_t} = [x_{1,t}, x_{2,t}, ..., x_{N,t}]$, the vector of transmitted packets as $\overline{u_t} = [u_{1,t}, u_{2,t}, ..., u_{N,t}]$, the vector of channel fading coefficients as $\overline{A_t} = [A_{1,t}, A_{2,t}, ..., A_{N,t}]$, and the vector of transmission frequencies as $\overline{f_t} = [f_{1,t}, f_{2,t}, ..., f_{N,t}]$, then the scheduler's mapping is $\alpha : [\overline{x_t}, \overline{A_t}] \rightarrow [\overline{u_t}, \overline{f_t}]$.

On the other hand, reliable communication can be guaranteed by having the scheduler to choose the power level $P_m(f^c, B)$ in time slot t such that the number of packets transmitted $u_{i,t}$ is equal to the Shannon capacity function for a Gaussian channel [69]. If the noise variance is $\frac{\sigma^2}{|A_{i,t}|^2}$ and the average power is $P_{i,t}$, then

$$u_{i,t} = B \times \frac{T_s}{S} \times \ln(1 + \frac{|A_{i,t}|^2 P_{i,t}}{\sigma^2}) \tag{3.3}$$

where B is the bandwidth, S is the packet size and T_s is the time slot length. For

simplicity, we assume that $S = B \times T_s$. Therefore,

$$P_{i,t} = \frac{\sigma^2(e^{u_{i,t}} - 1)}{|A_{i,t}|^2} \tag{3.4}$$

In line with the above information, we formulate the following scheduling problem, which maximizes the network throughput while satisfying the interference temperature constraints:

$$\max_{\overline{u_t}, \overline{f_t}} \; E\{\sum_{i=1}^{N} u_{i,t}\} \tag{3.5}$$

$$s.t.$$

$$P_{IF}(f_i, B) + L_{ij}^{f_i} \frac{\sigma^2(e^{u_{i,t}} - 1)}{|A_{i,t}|^2} < IT_{f_i}^{th} kB; \forall j \in \Phi_i, \forall i \in \{1, .., N\} \tag{3.6}$$

$$f_{i,t} \neq f_{i',t}; \forall i, i' \in \{1, .., N\}, i \neq i' \tag{3.7}$$

$$u_{i,t} \leq x_{i,t}; \forall i \tag{3.8}$$

where $P_{IF}(f_i, B)$ denotes the average sensed interference power at frequency f_i over bandwidth B, $L_{ij}^{f_i}$ is the distance dependent path loss from node i to node j with frequency f_i, $IT_{f_i}^{th}$ is the interference temperature threshold for frequency f_i, Φ_i is the set of primary nodes in the interference range of cognitive node i, N is the total number of cognitive nodes, and $f_{i,t}$ is the frequency used by node i in time slot t. In the above formulation, Equation 3.5 maximizes the expected value of the total number of packets transmitted by all the cognitive users, Equation 3.6 satisfies the interference temperature constraint, Equation 3.7 ensures that at most one cognitive user can transmit using a certain time slot and frequency combination, and Equation 3.8 represents the fact that a user cannot transmit more than the number of packets in its buffer at the beginning of the time slot.

The way the cognitive nodes learn about the already existing interference temperature in the neighboring primary nodes is an open issue in [4]. Therefore, as in [67], we assume a specialized environment where cognitive radios can locate licensed signals and measure the interference temperature. In other words, cognitive nodes learn about the

interference perceived by their neighboring primary nodes through their local spectrum sensing observations, which we denote here by $P_{IF}(f_i, B)$.

3.2. Proposed Interference Temperature Based Schedulers

3.2.1. Throughput Optimal Scheduler

Our solution to the problem in Equations 3.5-8 consists of two stages. In the first stage, every cognitive node i computes the maximum number of packets that can be transmitted for every frequency by solving the following problem for each frequency f_i:

$$D(f_i) = min(C(f_i, j)); \forall j \in \Phi_i \tag{3.9}$$

$$s.t.$$

$$IT_{f_i}^{th} kB - P_{IF}(f_i, B) > 0 \tag{3.10}$$

$$C(f_i, j) = \ln\left(\frac{(IT_{f_i}^{th} kB - P_{IF}(f_i, B)) \times |A_{i,t}|^2}{\sigma^2 L_{ij}^{f_i}} + 1\right) \tag{3.11}$$

In the above formulation, $\lfloor D(f_i) \rfloor$ equals the maximum number of packets that can be transmitted by cognitive node i using frequency f_i while adhering to the interference temperature constraints for all the primary nodes that are in the interference range of node i. Afterwards, all the cognitive nodes send their $\lfloor D(f_i) \rfloor$ values to the CBS. We assume here that the cognitive nodes have a priori knowledge about the number of primary nodes in their interference range as well as the path loss values to their neighbors. How the nodes acquire this information is beyond the scope of this work.
In the second stage of the algorithm, the CBS constructs a matrix called $\mathbf{U} = [U_{if}]$, where U_{if} is the maximum number of packets that can be transmitted by user i using frequency f, and hence being equal to the $\lfloor D(f_i) \rfloor$ value. The CBS then executes the

following binary integer linear program:

$$max(\sum_{i=1}^{N}\sum_{f=1}^{F}\sum_{t=1}^{T}\frac{U_{if}X_{ift}}{T}) \tag{3.12}$$

$$s.t.$$

$$\sum_{f=1}^{F}\sum_{t=1}^{T}X_{ift} \geq 1; \forall i \in \{1,..,N\} \tag{3.13}$$

$$X_{ift} + X_{i'ft} \leq 1; \forall i, i^{'} \in \{1,..,N\}, i \neq i^{'}, \forall f, \forall t \tag{3.14}$$

$$X_{ift} \in \{0,1\}; \forall i \in \mathcal{N}, \forall f \in \mathcal{F}, \forall t \in \mathcal{T} \tag{3.15}$$

where N is the total number of cognitive nodes, F is the total number of frequencies, T is the total number of time slots, and X_{ift} is a binary variable such that $X_{ift} = 1$ if user i transmits with frequency f in time slot t, and 0 otherwise. In the above formulation, Equation 3.13 ensures that each cognitive user is assigned at least one time slot, whereas Equation 3.14 guarantees that at most one user can transmit in a certain time slot and frequency pair, thereby avoiding collisions among the secondary nodes. Consider the case that a PU is in the interference range of two cognitive users. Since the cognitive users determine their $\lfloor D(f_i) \rfloor$ values, and consequently the U_{if} values by taking only their own transmissions into account, having more than one cognitive user transmit in the same frequency and time slot may increase the aggregate interference perceived at the PU beyond the interference temperature limit. Therefore, in addition to avoiding collisions among the secondary nodes, Equation 3.14 is also necessary to ensure that the aggregate interference temperature at the PUs is within the predetermined limits. Besides, the schedule length T is the duration of time in which the changes in the sensed interference values, denoted by $P_{IF}(f_i, B)$, as well as the path loss values to the PUs in the interference range are small enough not to have any impact on the U_{if} values. Note that because of the floor operator in $\lfloor D(f_i) \rfloor$, the schedule length T does not mandate $P_{IF}(f_i, B)$ and the path loss values to remain constant in that time period, but only requires that the change in their values does not alter U_{if}. The value of T, in general, depends on the characteristics of the spectrum environment. For instance, a slowly varying spectrum environment like the TV broadcast bands utilized by an

IEEE 802.22 network allows T to have a fairly large value. To ensure the feasibility of constraint in Equation 3.13, which prescribes that at least one time slot is assigned to each cognitive user, it is necessary that $F \times T \geq N$. In the simulations part of this chapter, we set $T = N$ because $T = N$ is sufficiently large to ensure the fulfillment of constraint in Equation 3.13.

Once the scheduler determines the U_{if} values, each node i transmits $\min(x_{i,t}, U_{if})$ number of packets in time slot t. We consider traffic in which all flows are continuously backlogged such that the achieved throughput is entirely related to the scheduling process and channel conditions without any variation due to traffic fluctuation.

3.2.2. Delay Optimal Scheduler

The first stage of the scheduler that minimizes the scheduling delay is the same as in the throughput optimal scheduler. However, in the second stage the delay optimal scheduler implements the following nonlinear binary integer program with linear constraints:

$$\min\left(\frac{\sum_{i=1}^{N}\sum_{f=1}^{F}\sum_{t=1}^{T} tU_{if}X_{ift}}{\sum_{i=1}^{N}\sum_{f=1}^{F}\sum_{t=1}^{T} U_{if}X_{ift}}\right) \tag{3.16}$$

$$s.t.$$

$$\sum_{i=1}^{N}\sum_{f=1}^{F}\sum_{t=1}^{T} U_{if}X_{ift} > 0; \tag{3.17}$$

$$(3.13) \quad \text{and} \quad (3.14) \tag{3.18}$$

$$X_{ift} \in \{0,1\}; \forall i \in \mathcal{N}, \forall f \in \mathcal{F}, \forall t \in \mathcal{T} \tag{3.19}$$

In the above formulation, when $X_{ift} = 1$, each one of the U_{if} packets waits for t time slots, starting from the beginning of the schedule. We use the term *scheduling delay* to refer to the number of time slots that a packet has to wait until its transmission, given that the packet is scheduled for transmission in that particular schedule. Therefore,

the total scheduling delay of these U_{if} number of packets is equal to tU_{if}. Hence, $\sum_{i=1}^{N}\sum_{f=1}^{F}\sum_{t=1}^{T} tU_{if}X_{ift}$ denotes the total scheduling delay experienced by all the transmitted packets in the schedule. Because the total number of transmitted packets equals $\sum_{i=1}^{N}\sum_{f=1}^{F}\sum_{t} U_{if}X_{ift}$, the objective function in Equation 3.16 minimizes the scheduling delay in terms of time slots experienced per packet. Moreover, Equation 3.17 is necessary to avoid the situation that the scheduler always selects the frequencies with which the nodes can send zero packets for the sake of reducing the average delay. Without Equation 3.17, the scheduler can arrive at the irrational decision of having none of the nodes being able to transmit any packets, which would result in zero throughput. In order to guarantee a certain throughput value while minimizing the scheduling delay, constraint in Equation 3.17 can be modified as follows: $\sum_{i=1}^{N}\sum_{f=1}^{F}\sum_{t=1}^{T} U_{if}X_{ift} > \Omega$, where Ω is the required minimum total throughput value. The two optimization problems in Equations 3.12-14 and Equations 3.16-18 are binary integer programming problems, which are in general known to be NP-hard. Branch-and-bound algorithms [70, 71] are usually adopted in the literature to address this kind of problems.

3.2.3. Maximum Frequency Selection (MFS) Suboptimal Scheduler

In the first stage of the MFS scheduler, each cognitive node i sends the frequency with which it wishes to transmit to the CBS. The nodes make their selection by finding $\max_{f_i} D(f_i)$. In the second stage, CBS makes the time slot assignments by first grouping the nodes with respect to the frequencies that they wish to transmit with. Among the cognitive nodes in the same frequency group, CBS assigns the node with $\max_{f_i} D(f_i)$ to the first time slot, the second maximum to the second time slot etc. This way, the condition that at most one cognitive node can be assigned a certain time slot and frequency combination is ensured. Furthermore, the scheduling delay is also reduced by doing the time slot assignment in the decreasing order of their allowable number of packets to transmit.

3.2.4. Probabilistic Frequency Selection (ProbFS) Suboptimal Scheduler

As in the MFS scheduler, in the first stage each node sends the frequency with which it wishes to transmit to the CBS. However, the selection of the desired frequency is made probabilistically as follows: A cognitive node i chooses frequency f_i with probability $p_{f_i} = \dfrac{\lfloor D(f_i) \rfloor}{\sum_{f_i} \lfloor D(f_i) \rfloor}$. In order to reduce the average delay, if frequency f_i was selected in the previous schedule and node i waited for T_{f_i} time slots with a schedule length of T, then as a penalty metric p_{f_i} is updated as $p_{f_i} \times (1 - \dfrac{T_{f_i}}{T})$. The selection probability of the frequency that has the highest number of packets among the remaining ones is increased by $\dfrac{p_{f_i} \times T_{f_i}}{T}$, which makes the total probability equal to 1. The second stage of the ProbFS scheduler is the same as MFS.

In order to better comprehend the different behavior of MFS and ProbFS schedulers, suppose that there are three SUs and three frequencies in the network. Assume that $\max_{f_i} D(f_i) = f^3$ for all the three nodes and $\lfloor D(f_i) \rfloor = 22, 21, 20$ for node $1, 2,$ and 3 respectively. Thus, MFS scheduler selects node 1 for the first time slot, node 2 for the second time slot and node 3 for the third time slot. Assume that node 3 can transmit 19 and 18 packets with the other frequencies f^1 and f^2. Note that node 3 could have used f^1 and f^2 in the first and second time slots, while other nodes used f^3. This would decrease the scheduling delay for node 3 because the packets would wait for less number of time slots by being able to transmit earlier. Nevertheless, MFS scheduler does not allow this frequency and time slot usage pattern by always selecting $\max_{f_i} D(f_i)$. In contrast, ProbFS scheduler probabilistically allows node 3 to select frequencies f^1 or f^2 as the frequency that it wishes to transmit with, hence enabling node 3 to transmit in earlier time slots. This way, ProbFS scheduler disperses the selected frequencies and avoids the above situation, which could occur with MFS scheduler. Because of this behavior of the ProbFS scheduler in addition to the penalty metric that it introduces, we intuitively expect ProbFS scheduler to have less scheduling delay on the average. Nevertheless, since ProbFS scheduler probabilistically allows the selection of frequencies whose maximum number of packets for transmission is smaller, we intuitively expect the average throughput of ProbFS scheduler to be smaller than

the throughput of MFS scheduler.

3.2.5. Computational Complexity Comparison

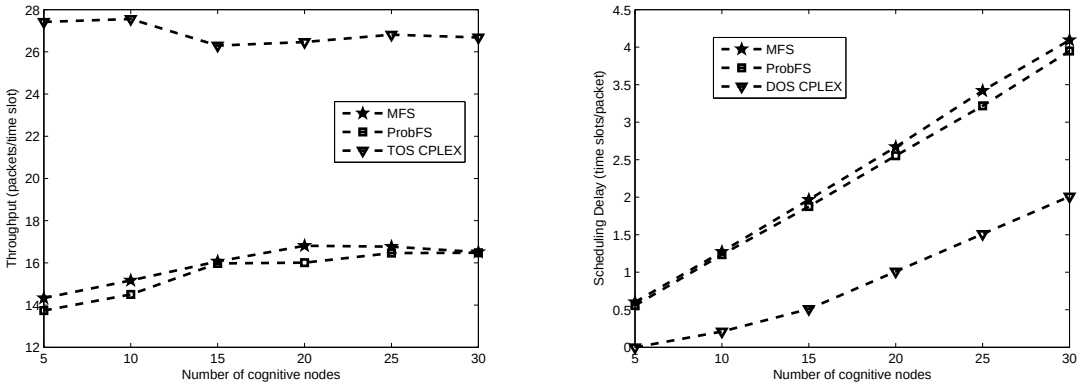
If the number of frequencies is F and each cognitive node has M primary neighbors in its interference range, the first stage of all the schedulers requires $M \times F$ computations at each node. We can assume that M is fixed. Hence, the computational complexity of the first stage is $O(F)$. On the other hand, there is one value for the maximum allowable number of packets to transmit corresponding to a certain frequency for each secondary node in the suboptimal schedulers. Since the number of cognitive nodes is N, the computational complexity of the suboptimal schedulers is $O(NF)$.

3.3. Numerical Evaluation

We simulate the suboptimal schedulers using OPNET Modeler 14.0 [72]. While the first stage of the optimal schedulers is simulated and the U_{if} values are obtained in OPNET, the optimization procedures in the second stages are implemented using CPLEX [5]. First, we consider AWGN channels; i.e., $|A_{i,t}| = 1, \forall i$ and $\forall t$. Second, we evaluate the performance of our proposed schedulers under Gilbert-Elliot fading channel model. Bandwidth is $B = 10$ MHz and the noise variance is $\sigma^2 = 10^{-10}$. PU activity is modeled such that the initially sensed interference for each frequency f_i is uniformly distributed in $[0, 2 \times IT^{th}_{f_i} kB/\sigma^2]$. If the sensed interference at the beginning of a particular scheduling period is $P_{IF}(f_i, B)$, then the sensed interference in the next scheduling period is uniformly distributed in $[P_{IF}(f_i, B) - \delta, P_{IF}(f_i, B) + \delta]$, where δ is selected to be 0.65 mW. Besides, path loss of the cognitive nodes to each primary neighbor is uniformly distributed between 0 and 1. Average values in all of the results were obtained using 10 different seeds and 10000 scheduling periods for each seed. In the simulation results, TOS CPLEX and DOS CPLEX denote the CPLEX results of the throughput optimal and delay optimal scheduler, respectively.

In the first set of simulations, each cognitive node has three primary neighbors

in its interference range. The channel between the SUs and the CBS is AWGN channel. There are three frequencies with interference temperature thresholds of $1000°K$, $2000°K$, and $3000°K$.



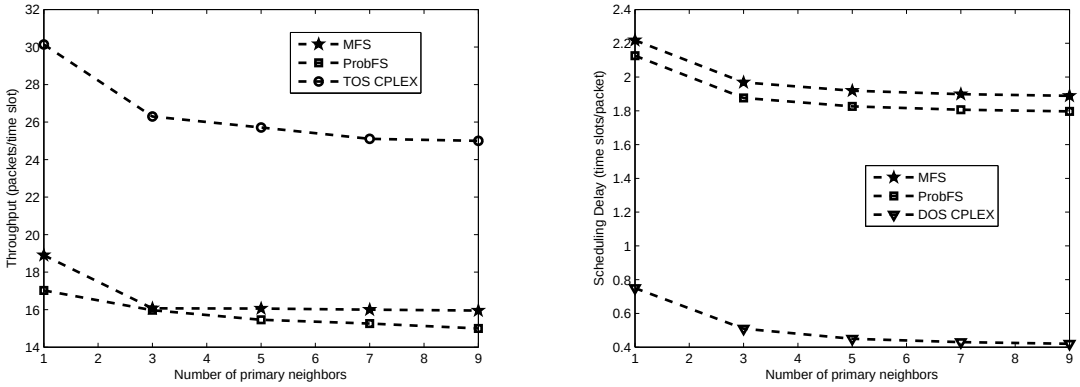
(a) Average network throughput

(b) Average scheduling delay

Figure 3.2. Average network throughput and scheduling delay for varying number of cognitive nodes.

Figures 3.2a and 3.2b illustrate the average network throughput and average scheduling delay values where the number of cognitive nodes varies between 5 and 30. For all three schemes, throughput values remain almost invariant as the number of secondary nodes increases, whereas the average scheduling delay increases almost linearly. Furthermore, ProbFS scheduler has a slightly less average scheduling delay than MFS at the expense of a little decrease in average network throughput compared to MFS. This improvement in delay is due to the penalty metric introduced in ProbFS in order to decrease the scheduling delay. Note here that because of the scale of the graph, the difference between the throughput values of MFS and ProbFS schedulers for 15 and 30 nodes is not visible. The actual average network throughput values for 30 nodes are 16.51 packets/time-slot for MFS and 16.47 packets/time-slot for ProbFS. Similarly, the average throughput for 15 nodes is 16.07 packets/time-slot for MFS and 15.97 packets/time-slot for ProbFS.

In the second set of simulations, again there are three frequencies with interference



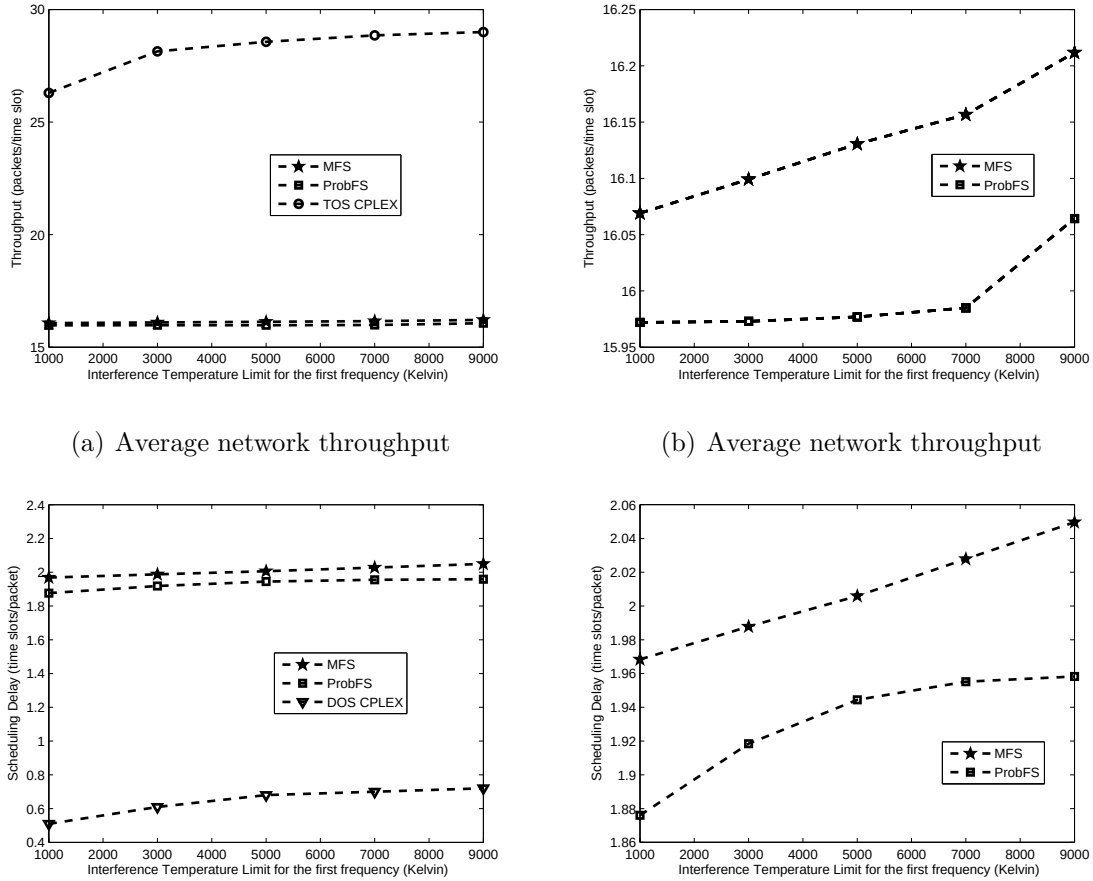
(a) Average network throughput

(b) Average scheduling delay

Figure 3.3. Average network throughput and scheduling delay for varying number of primary neighbors of the cognitive nodes.

temperature thresholds of $1000°K$, $2000°K$, and $3000°K$, and the channel between the SUs and the CBS is AWGN channel. However, this time we vary the number of primary neighbors of each cognitive node while keeping the total number of cognitive nodes constant. Figures 3.3a and 3.3b illustrate the average network throughput and average scheduling delay for 15 cognitive nodes with varying number of primary neighbors for each node. For all the scheduling schemes, throughput decreases as the number of primary neighbors for each secondary node increases. However, the rate of decrease diminishes as the number of primary neighbors increases. The reason for this behavior is that cognitive nodes have more interference temperature constraints as the number of their primary neighbors increases. Therefore, their $\lfloor D(f_i) \rfloor$ values and consequently U_{if} values decrease and hence the average network throughput diminishes. Figure 3.3b illustrates that the average scheduling delay decreases as the number of primary neighbors for each cognitive node increases and the rate of decrease diminishes as the number of primary neighbors increases. This behavior in average scheduling delay is in line with the throughput performance results of Figure 3.3a.

In the third set of simulations, we again model the channel between the SUs and the CBS as an AWGN channel. We vary the interference temperature limits of the frequencies, while keeping the total number of cognitive nodes and the number of primary neighbors of each cognitive node constant. Figure 3.4a illustrates the average network



(a) Average network throughput

(b) Average network throughput

(c) Average scheduling delay

(d) Average scheduling delay

Figure 3.4. Average network throughput and scheduling delay for varying interference temperature limit.

throughput for 15 cognitive nodes, each having 3 primary neighbors in its interference range with increasing values of interference temperature limit. In this figure, values on the x-axis correspond to the interference temperature limit for the first frequency. That is to say, if the value on the x-axis is $A°K$, then the limits for the second and third frequencies are $A + 1000°K$ and $A + 2000°K$ respectively. Figure 3.4a indicates that the network throughput increases as the interference temperature limit increases. The results in Figure 3.4b are essentially the same as the ones in Figure 3.4a with the only difference that we do not show the results of the throughput optimal scheduler in Figure 3.4b in order to provide a better visualization of the throughput increase in the MFS and ProbFS schedulers as the interference temperature limit increases. This increase in average network throughput is due to the fact that the $\lfloor D(f_i) \rfloor$ values and

consequently the U_{if} values of the cognitive nodes increase as the interference temperature limits increase because increasing the interference temperature limit represents the FCC allowing more interference from unlicensed devices. Furthermore, the rate of increase decreases for the optimum scheduler as the interference temperature limit increases. The observed decrease in the rate of increase is consistent with the results in [73], where the network capacity saturates at a certain level after an initial increase as the interference temperature limit increases.

Figure 3.4c shows the average scheduling delay of 15 cognitive nodes, again each having 3 primary neighbors in its interference range, for increasing values of interference temperature limit. This figure illustrates that the average scheduling delay increases as the interference temperature limit increases. Figure 3.4d better illustrates the increase in MFS and ProbFS schedulers. Similar to the situation between Figures 3.4a and 3.4b, the only difference between Figures 3.4c and 3.4d is that we do not show the performance results of the delay optimal scheduler in Figure 3.4d in order to provide a better visualization of the increase in the average scheduling delay as the interference temperature limit increases. Moreover, the results also indicate that the average scheduling delay values of the optimal as well as the MFS scheduler stabilize around $7000°K$. This increase in average scheduling delay is consistent with the increase in average network throughput as the interference temperature limit increases.

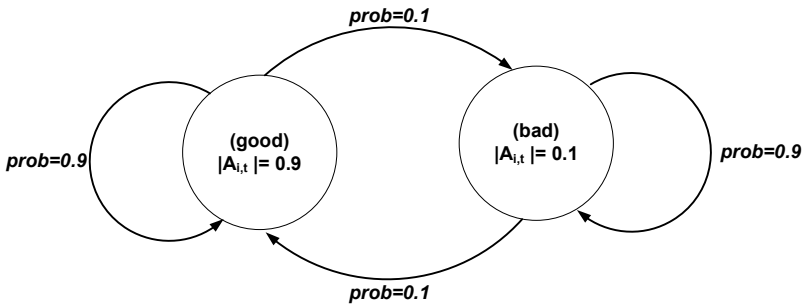
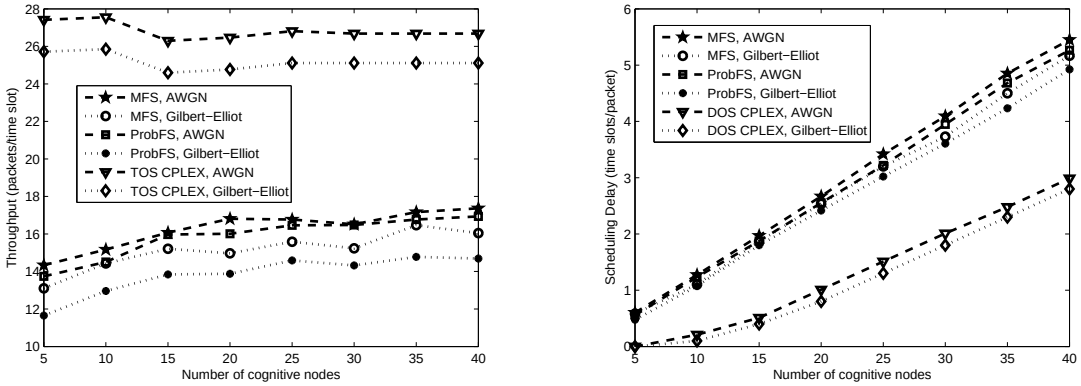


Figure 3.5. Gilbert-Elliot channel model.

The fourth set of simulations investigate the impact of channel fading on the performance of our proposed schedulers. We model the fading process as a Gilbert-

Elliot channel, which we illustrated as a 2-state Markov process in Figure 3.5. The fading coefficient is high when the channel is in the good state, whereas it is low in the bad state. We have chosen the state transition probabilities as 0.1, which is a small number compared to 0.9, the probability of staying in the same state. We have made this selection in order to reflect the slow fading channel process. The rest of the simulation conditions is the same as the ones in the first set of simulations, where we evaluated the AWGN channels.



(a) Average network throughput

(b) Average scheduling delay

Figure 3.6. Average network throughput and scheduling delay with Gilbert-Elliot channel.

Figure 3.6a illustrates the average network throughput with Gilbert-Elliot channel as the number of cognitive nodes increases from 5 to 40 for all three schedulers. To facilitate the visual comparison with the AWGN channel performance, we have also shown in Figure 3.6a the performance results previously displayed in Figure 3.2a. In line with the theoretical expectations, the average network throughput decreases as the fading conditions in the channel deteriorate. Hence, Gilbert-Elliot channel for all three schedulers yields reduced average network throughput when compared to their AWGN channel counterparts.

Figure 3.6b shows the average scheduling delay with Gilbert-Elliot channel as the number of cognitive nodes increases from 5 to 40 for all three schedulers. For better visual comparison with the AWGN channel performance, we have also shown here the

performance results previously illustrated in Figure 3.2b. The reason that the average scheduling delay of the Gilbert-Elliot channel is less than the AWGN channel for all the three schedulers is that $\lfloor D(f_i) \rfloor$ values and consequently U_{if} values decrease as the fading condition of the channel deteriorates. The reasoning here is the same as the one where we varied the number of PUs in the interference range of each SU: The decrease in U_{if} values leads to reduced average scheduling delay due to the formulation in Equation 3.16. As in the preceding simulation results, the decrease in the average scheduling delay is consistent with the decrease in the average network throughput.

To put it in a nutshell, the average network throughput of the MFS scheduler is slightly higher than that of the ProbFS scheduler in all the simulation results. Furthermore, the average scheduling delay of the ProbFS scheduler is slightly lower than that of the MFS scheduler in all the results. These two observations are consistent with our theoretical expectations, which we outlined in Section IV-D. On the one hand, the reason for the superior throughput performance of the MFS scheduler is that MFS scheduler always selects the channel with the maximum allowable number of packets to transmit per time slot, whereas the ProbFS scheduler may select the channel with reduced allowable number of packets to transmit per time slot owing to its probabilistic selection of the channels. On the other hand, the delay performance of the ProbFS scheduler is better than that of MFS scheduler due to the penalty metric introduced in the design of ProbFS. Besides, the throughput and delay performance of the optimal schedulers are significantly better than that of suboptimal schedulers. As a research challenge, simulation results indicate that better performing, yet computationally efficient suboptimal schedulers are needed. We address this open research issue in Chapter 4 by proposing genetic algorithm based schedulers.

4. GENETIC ALGORITHM BASED SCHEDULERS UNDER INTERFERENCE TEMPERATURE CONSTRAINTS

In this chapter, we propose genetic algorithm (GA) based schedulers for the throughput and delay optimal scheduling problems formulated in Chapter 3. Contents of this chapter appeared in [74].

4.1. Overview of Genetic Algorithms

A genetic algorithm (GA) is a biologically inspired heuristic search technique appropriate for problems with large search spaces. It operates by emulating natural processes like reproduction, evolution, and survival. In nature, populations of individuals compete for resources, and the best suited for competition survive, whereas the weakest tends to die out. This phenomenon is referred to as *"survival of the fittest"* in evolution. These ideas were first incorporated into a problem solving algorithm by [75] and they are now used in a multitude of problems.

A GA operates by evolving a *population* of solutions called *chromosomes*. A chromosome is a binary string that represents one sample in the solution space of the problem. The bits of the string are regarded as the *genes* of the chromosome. The *fitness value* assigned to a chromosome exhibits the extent to which the chromosome satisfies the problem requirements. A GA takes a group of chromosomes from a population using a genetic operation called *selection*, and mixes the genes of these chromosomes using *crossover* to produce offspring. These offspring solutions may be further randomly altered using a genetic operation called *mutation*.

The fact that GAs operate on a population of solutions rather than a single solution implies that the algorithm makes parallel searches in the search space. The selection operation, on the other hand, serves the purpose of eliminating the relatively

bad solution candidates and focusing the search operation on a relatively good portion of the solution space. Crossover operation is based on the idea that if two solution candidates that are both good but for different reasons (for instance the first half of the bits in one candidate have desirable qualities, like not violating any problem constraints, and the second half of the bits in the other candidate have good features in a similar way), then we can obtain an even better solution if we take the good parts of both solutions and combine them. Mutation, on the other hand, is like introducing some noise with little magnitude into the system in order to take the search procedure out of a locally optimal region, and enable the search process to possibly delve into a better region of the search space.

Our motivation for utilizing GAs in designing suboptimal schedulers for the throughput and delay optimal scheduling problems formulated in Chapter 3 is manifold. First, GAs are proper for problems with large search spaces. They are equipped with many tools to reduce computational complexity and produce a diverse set of solutions since they can quickly center in on a specific solution and diversify search to develop wide range of solutions to address unknown environments. Considering that the solution space in the throughput and delay optimal scheduling problems is enormous (even for 5 nodes, 3 frequencies and 5 time slots, the size of the solution space for the throughput optimal scheduler is 2^{75}), GAs seem to be a suitable tool. Second, GAs can be implemented on semiconductor devices and enable rapid integration with wireless technologies and leverage economies of scale. Rapid prototyping is possible using digital signal processor (DSP) or field programmable gate array (FPGA). Third, binary decision variables X_{ift} can be easily encoded to a binary string; therefore, GAs can be conveniently implemented.

4.2. GA Based Suboptimal Schedulers

Figure 4.1 illustrates the flowchart of our proposed GA based algorithm for both throughput and delay optimal scheduling schemes.

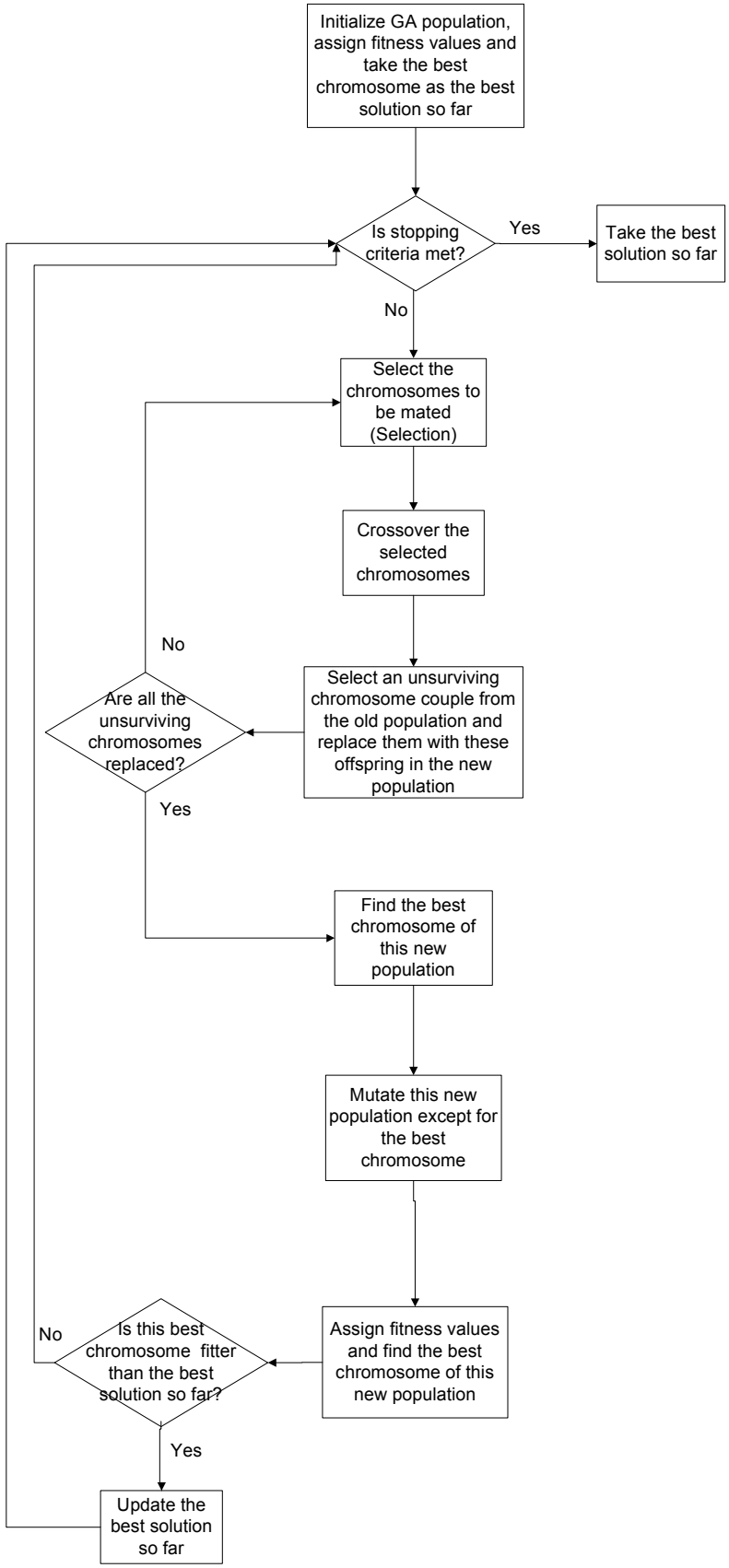


Figure 4.1. Block diagram for our proposed GA based algorithm.

4.2.1. Chromosome Representation (Encoding)

We use binary encoded chromosomes containing the X_{ift} values. Thus, the chromosome size is equal to $N \times F \times T$. The order used in decoding the possible X_{ift} values has an impact on the performance of the algorithm since this order affects the gene pattern that can survive in the subsequent generations. In our analysis, we evaluate the impact of the following two encoding methods. In *Encoding Type-1*, the chromosome structure is $[X_{111}, X_{211}, X_{311}, X_{112}, X_{212}, \cdots, X_{123}, X_{223}, X_{323}]$, while in *Encoding Type-2*, the chromosome structure is $[X_{111}, X_{112}, X_{113}, X_{121}, X_{122}, \cdots, X_{322}, X_{323}]$ for $N = 3, F = 2$, and $T = 3$.

4.2.2. Initial Population Creation

The most common way to generate the initial population is to employ uniformly random generation for each bit of the chromosomes. We call this approach *Rand* in this work. Another alternative, which we refer to as *RandComp*, is to randomly generate half of the chromosomes, and then take the complement of the first half to generate the second half [76]. This approach ensures diversity by requiring every bit to assume both a one and a zero within the population. In both of our suboptimal schedulers, we assess the impact of both approaches on the performance of the algorithm.

4.2.3. Fitness Function Evaluation

The extent to which a chromosome satisfies the problem requirements depends on two factors. The first one is how much it maximizes or minimizes the objective function, and the second one is how many of the problem constraints it violates. We represent the former by a *primary fitness* value and the latter by a *secondary fitness* value. If any constraint is violated, the primary fitness F_p equals zero and the secondary fitness F_s is nonzero. Likewise, if no constraint is violated, F_s equals zero and F_p is nonzero. More formally, we formulate the fitness values for the throughput maximizing

scheduler as follows:

$$F_p = \begin{cases} 0 & ; \text{if } V_1 + V_2 > 0, \\ \sum_{i=1}^{N} \sum_{f=1}^{F} \sum_{t=1}^{T} \frac{U_{if} X_{ift}}{T} & ; \text{otherwise (if } V_1 + V_2 = 0) \end{cases} \tag{4.1}$$

$$F_s = \begin{cases} 0 & ; \text{if } V_1 + V_2 = 0, \\ \frac{1}{V_1 + V_2} & ; \text{otherwise (if } V_1 + V_2 > 0) \end{cases} \tag{4.2}$$

where V_1 is the number of violations of constraint in Equation 3.13, and V_2 is the number of violations of constraint in Equation 3.14. If $V_1 + V_2 > 0$, then it means that some constraint is violated. If $V_1 + V_2 = 0$, on the other hand, it means that none of the constraints are violated.

Specifically, we can express V_1 and V_2 as follows:

$$V_1 = \sum_{i=1}^{N} V_1^i, \text{where } V_1^i = \begin{cases} 1 & ; \text{if } \sum_{f=1}^{F} \sum_{t=1}^{T} X_{ift} < 1, \forall\ i \in \{1, .., N\}, \\ 0 & ; \text{otherwise} \end{cases} \tag{4.3}$$

$$V_2 = \sum_{i=1}^{N} \sum_{i'=i+1}^{N} \sum_{f=1}^{F} \sum_{t=1}^{T} {V_2}^{ii'ft} \tag{4.4}$$

$$\tag{4.5}$$

$${V_2}^{ii'ft} = \begin{cases} 1 & ; \text{if } X_{ift} + X_{i'ft} > 1, \\ 0 & ; \text{otherwise} \end{cases} \tag{4.6}$$

Besides, we define the fitness functions for the delay minimizing scheduler as

follows:

$$F_p = \begin{cases} 0 & ; \text{if } V_1 + V_2 + V_3 > 0, \\ \dfrac{\sum_{i=1}^{N}\sum_{f=1}^{F}\sum_{t=1}^{T} U_{if}X_{ift}}{\sum_{i=1}^{N}\sum_{f=1}^{F}\sum_{t=1}^{T} tU_{if}X_{ift}} & ; \text{otherwise (if } V_1 + V_2 + V_3 = 0) \end{cases} \tag{4.7}$$

$$F_s = \begin{cases} 0 & ; \text{if } V_1 + V_2 + V_3 = 0, \\ \dfrac{1}{V_1 + V_2 + V_3} & ; \text{otherwise (if } V_1 + V_2 + V_3 > 0) \end{cases} \tag{4.8}$$

$$V_3 = \begin{cases} 0 & ; \text{if } \sum_{i=1}^{N}\sum_{f=1}^{F}\sum_{t=1}^{T} U_{if}X_{ift} > 0, \\ 1 & ; \text{otherwise} \end{cases} \tag{4.9}$$

where V_3 is the number of violations of constraint in Equation 3.17, and V_1 and V_2 are the same as in the throughput maximizing scheduler.

```
procedure CompareFitness (F_p^α,F_p^β,F_s^α,F_s^β)
//Return the chromosome having higher fitness value, return 0 if equal
if F_p^α > F_p^β then
   Return α
else if F_p^α < F_p^β then
   Return β
else if F_s^α > F_s^β then
   Return α
else if F_s^α < F_s^β then
   Return β
else
   Return 0
end if
```

Figure 4.2. Pseudocode for fitness comparison.

We demonstrate in Figure 4.2 the method we employ for the overall fitness com-

parison of two chromosomes α and β having primary fitness F_p^α, F_p^β and secondary fitness F_s^α, F_s^β. Primary fitness values have high priority in the comparison procedure; that is to say, the chromosome that has higher primary fitness value than the other chromosome is declared to have the higher fitness value. The reason for this behavior is that a chromosome that does not violate any constraints and hence has positive primary fitness value is fitter than the one that violates some constraint and therefore having a primary fitness value of zero. Likewise, when comparing two chromosomes that both have a positive primary fitness value (i.e., none of them violate any constraint), the one that better satisfies the objective function has the higher fitness value. If both chromosomes have zero primary fitness value; i.e., both chromosomes violate some problem constraint, then the chromosome that violates less number of problem constraints and hence has higher secondary fitness value is declared to have the higher fitness value.

One of the possible approaches employed when handling constraint based problems using GAs is to penalize a constraint violating chromosome by setting its fitness value to zero, and hence basically nullifying its chance to survive to the next generation [29, 77]. Nevertheless, in our scheme, since the probability that constraint in Equation 3.14 is violated in a randomly generated population is quite high (will be discussed in detail in Section 4.3), nullifying the fitness values of the constraint violating chromosomes would not make sense. Some gene pattern might violate a constraint, but another part of the chromosome might be fit in terms of the primary fitness value. Preventing this chromosome from surviving in the subsequent generations would decrease diversity. Diversity helps prevent the algorithm from getting stuck in a local optimum. Therefore, we adopt the mechanisms outlined in (4.1)-(4.9) for the fitness function evaluation.

4.2.4. Selection

After the population is sorted by comparison of the fitness values according to Figure 4.2, the top $N_{pop} \times R_{select}$ number of chromosomes are included in the mating pool, where N_{pop} is the population size and R_{select} is the selection rate, which was chosen

to be 0.5 in our work. The mother and father chromosomes are selected from this pool and mated through crossover mechanisms (explained in detail in Section 4.2.5). Each crossover generates two offspring, which replace two chromosomes from the bottom of the population that is not in the mating pool. This two at a time replacement process continues until all the chromosomes that are not in the mating pool are replaced. This way, chromosomes that have high fitness, i.e. the ones in the mating pool, survive in the subsequent generations. In contrast, chromosomes that have low fitness, i.e. the ones that are not in the mating pool, do not survive and are replaced by the offspring of the mating pool, which potentially have higher fitness.

We evaluate the performance of two different selection schemes. The first one is *Roulette Wheel Selection*, also referred to as *Proportional Selection* or *Weighted Random Pairing with Cost Weighting* [76], whereas the second one is *Tournament Selection*.

procedure *RouletteWheel (population)*
if $N_p > 0$ **then**
  //Select among the chromosomes having $F_p > 0$
  Return α *with probability* $F_p^{\alpha} / \sum_{\alpha'=1}^{N_p} F_p^{\alpha'}$
else
  //Select among the chromosomes having $F_p = 0$
  Return α *with probability* $F_s^{\alpha} / \sum_{\alpha'=1}^{N_s} F_s^{\alpha'}$
end if

Figure 4.3. Pseudocode for Roulette Wheel Selection.

Roulette wheel selection is a way of choosing mother and father chromosomes from the population in a way that is proportional to their fitness. We have made pertinent modifications to the general roulette wheel selection framework to accommodate our fitness function evaluation and comparison techniques. In our version of roulette wheel selection scheme, we choose the mother chromosome according to the algorithm outlined in Figure 4.3, where N_p is the number of chromosomes in the mating pool that have nonzero primary fitness value (that do not violate any problem constraint) and N_s

denotes the number of chromosomes in the mating pool that have nonzero secondary fitness value (that violate some problem constraint). Furthermore, F_p^{α} and $F_p^{\alpha'}$ denote the primary fitness values of the chromosomes α and α', respectively. Likewise, F_s^{α} and $F_s^{\alpha'}$ denote the secondary fitness values of the chromosomes α and α', respectively. In order to create more diversity, we do not allow the mating of a chromosome with itself. Hence, we select the father chromosome in the same way as the mother, but from a population that excludes the mother.

Tournament selection approach picks a small subset of chromosomes (two or three in general, three in our case) from the mating pool in a uniformly random manner. The chromosome with the highest fitness in this subset becomes a parent, where the fitness values of the chromosomes are compared with each other according to our proposed algorithm outlined in Figure 4.2. The tournament repeats for every parent needed. It is computationally simpler than roulette wheel selection because the population does not need to be sorted [76].

4.2.5. Crossover

Crossover is a genetic operator analogous to reproduction and biological crossover. It is used to vary the programming of chromosomes from one generation to the next. In this work, we evaluate the performance of three crossover types, namely, single-point crossover, two-point crossover, and uniform crossover.

In single-point crossover, a single crossover point on both parents' strings is selected. All data beyond that point in either organism string is swapped between the two parent organisms. The resulting organisms are the offspring. Two-point crossover requires that two points are selected on the parent organism strings. Everything between the two points is swapped between the parent organisms, rendering two child organisms. In the uniform crossover scheme, individual bits in the string are compared between two parents. The bits are swapped with a fixed probability [76], which we selected as 0.5.

4.2.6. Mutation

A bit within a chromosome is inverted with probability equal to the mutation rate, denoted as μ_m. In our scheme, we mutate every chromosome of the population except for the best chromosome. The exclusion of the best chromosome in mutations is a common practice in GAs, since the best chromosomes are designated as *elite* solutions destined to propagate unchanged [76].

4.2.7. Stopping Criteria

We stop the execution of the GA when any of the two following conditions are met: Either the same best solution has been found for N_{best} number of iterations or the maximum number of iterations, N_{max}, has been reached.

4.3. Numerical Evaluation

As in Chapter 3, we have simulated the first stages of all schedulers and acquired the U_{if} values in OPNET Modeler [72]. In the second stages, we have implemented the GA-based suboptimal schedulers in OPNET, whereas we solved the optimization problems in CPLEX [5]. Additive white gaussian noise (AWGN) channels are considered. The bandwidth, noise variance, path loss, and PU activity models are as in Chapter 3. In all simulations, each SU has three primary neighbors in its interference range. There are three frequencies with interference temperature thresholds of $1000^{\circ}K$, $2000^{\circ}K$, and $3000^{\circ}K$. For the GA based schedulers, the selection rate $R_{select} = 0.5$ and $N_{max} = 5000$. Average values in all results have been obtained using 10 different seeds and 10000 scheduling periods for each seed.

The methodology we employ for both throughput maximizing and delay minimizing GA based suboptimal schedulers is firstly to evaluate the impact of numerous methods outlined in Table 4.1 in a relatively small cognitive radio network consisting of $N = 5$ cognitive nodes. This first evaluation equips us with the knowledge about which set of methods outlined in Table 4.1 suits best for our problem. We then evaluate the

Table 4.1. Test case parameters.

Case	Initial Population Creation	Encoding	Selection
1	RandComp	Type-1	Roulette Wheel
2	Rand	Type-2	Roulette Wheel
3	Rand	Type-1	Tournament
4	Rand	Type-1	Roulette Wheel

performance of the schedulers using these determined methods for varying number of cognitive nodes; i.e., $N = 5, 10, \ldots, 30$. This sequential experimental design method of employing a series of smaller experiments each with a specific objective is a common method in experimental design [78] because the experimenter can quickly learn crucial information from a small group of runs that can be used to plan the next experiment. Employing a very large experiment directly in the first steps is usually considered to be a waste of time [78]. In line with this design approach, we initially make a series of experiments using $N = 5$ SUs and observe the impact of different method combinations, and then evaluate the scalability of the system with $N = 5, 10, ..., 30$ SUs using the methods that have been found to yield better results in the initial experiments.

Firstly, we consider 5 SUs with $N_{pop} = 100$, $N_{best} = 50$, $\mu_m = 0.01$, and single-point crossover. We evaluate the performance of parameter sets defined in Table 4.1. For the throughput maximizing GA based suboptimal schedulers, the achieved average network throughput and the average number of iterations are stated in Table 4.2.

First of all, when we compare the results of GA-based solutions (Case 1-4), we see that all of our GA based solutions yield almost twice better results than the MFS and ProbFS schedulers proposed in Chapter 3, at the same time being very close to the throughput optimal scheduler's performance.

When we compare the results of Case 1 and 4, where only the initial population creation method is different, we observe that Case 1 has a slightly larger throughput

Table 4.2. Results for throughput maximizing GA scheduler
$N = 5, N_{pop} = 100, N_{best} = 50, \mu_m = 0.01$, and single-point crossover.

Case	Average network throughput	Average number of iterations
1	26.58	155.27
2	26.23	171.24
3	26.43	132.77
4	26.57	154.24
Throughput Optimal	27.41	–
MFS	14.33	–
ProbFS	13.74	–

at the expense of a slightly higher number of iterations, leading to the conclusion that either scheme might be preferred. Since the resulting throughput of both schemes are very close to the optimal one, we prefer to select Case 4 as our candidate parameter set.

When we examine the results of Case 2 and Case 4, where only the encoding type is different, we observe that Case 4 achieves higher throughput as well as a significantly reduced average number of iterations. In order to investigate the reason for the superior performance of *Encoding Type-1* over *Encoding Type-2*, note that *Encoding Type-1* has the ability to recognize a gene pattern that does not violate constraint in Equation 3.14 because the genes that correspond to the same time slot and frequency pair are next to each other in this representation. For instance, if a chromosome starts with the bit string $[010100\ldots]$, then $X_{111} = 0, X_{211} = 1, X_{311} = 0$ and $X_{112} = 1, X_{212} = 0, X_{312} = 0$ for $N = 5, F = 3$, and $T = 5$. This gene pattern does not violate constraint in Equation 3.14 and hence has higher fitness value. In consequence, *Encoding Type-1* enables this kind of a gene pattern to prevail and produce better individuals in the subsequent generations through crossover. Similarly, *Encoding Type-2* is capable of preserving gene patterns that do not violate constraint in Equation 3.13.

Let P_{nv-1} , where *nv* stands for *"not violate"*, denote the probability that constraint in Equation 3.14 is not violated in a randomly generated chromosome for N nodes, F frequencies, and T time slots. If we consider the entire binary string of the chromosome as the concatenation of $F \times T$ string groups, each with N strings encoded according to *Encoding Type-1*, and denote the probability that constraint in Equation 3.14 is not violated in a group by P_{gnv-1}, where *gnv* stands for *"group not violate"*, then $P_{nv-1} = (P_{gnv-1})^{F \times T}$. Besides, probability that constraint in Equation 3.14 is not violated in a group equals the probability that bit '1' occurs at most once in a randomly generated string of length N, which in turn equals $0.5^N + (N \times 0.5^N) = (N+1)/2^N$. Thus, $P_{nv-1} = ((N+1)/2^N)^{F \times T}$.

Likewise, let P_{nv-2} denote the probability that constraint in Equation 3.13 is not violated in a randomly generated chromosome of length $N \times F \times T$. If we consider the whole binary string as the concatenation of N string groups each with $F \times T$ strings encoded according to *Encoding Type-2* and denote the probability that constraint in Equation 3.13 is not violated in a group by P_{gnv-2}, then $P_{nv-2} = (P_{gnv-2})^N$. Note that P_{gnv-2} equals the probability that bit '1' occurs at least once in a randomly generated string of length $F \times T$. Therefore, $P_{gnv-2} = 1 - (0.5)^{F \times T}$ and $P_{nv-2} = (1 - (0.5)^{F \times T})^N$. Even for $N = 5, F = 3$, and $T = 5$, $P_{nv-1} = 1.24 \times 10^{-11}$, whereas $P_{nv-2} = 0.99$. Therefore, we can conclude that the probability that constraint in Equation 3.14 is violated in a randomly generated chromosome is much higher than the probability that constraint in Equation 3.13 is violated. Whenever Equation 3.14 is violated, it will drive the primary fitness value of the chromosome to zero; therefore, being able to recognize and track a non-violating pattern will enable this gene pattern with good traits to prevail in the next generations and thus conduce better fitness values as well as faster convergence. For this reason, *Encoding Type-1* yields superior performance compared to *Encoding Type-2*.

Finally, when we compare the performance of Case 3 and Case 4 in Table 4.2, where only the selection type differs, we observe that the average number of iterations of tournament selection is less than the one of roulette wheel selection at the expense of a little decrease in average network throughput. Since the difference in throughput

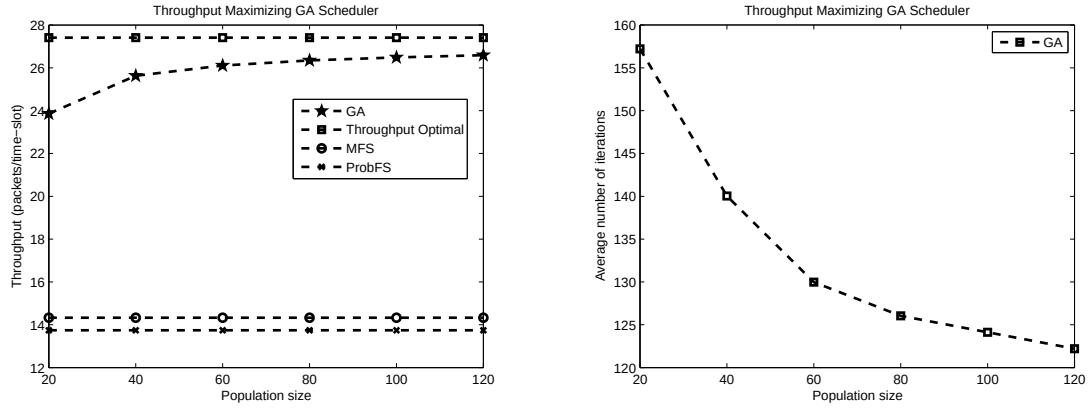
is slight, we conclude that faster convergence is more important and hence Case 3 is superior to Case 4. Therefore, in the following simulations, we evaluate Case 3 with different crossover types.

Table 4.3. Crossover type comparison for throughput maximizing GA scheduler Case 3, $N = 5, N_{pop} = 100, N_{best} = 50, \mu_m = 0.01$.

Crossover Type	Average throughput	Average number of iterations
Single Point	26.43	132.77
Two Point	26.44	130.88
Uniform	26.48	127.20
Throughput Optimal	27.41	–
MFS	14.33	–
ProbFS	13.74	–

Table 4.3 presents the average network throughput and average number of iterations values for Case 3 with single-point, two-point and uniform crossover, where $N = 5, N_{pop} = 100, N_{best} = 50$, and $\mu_m = 0.01$. We observe that uniform crossover outperforms the other two schemes both in terms of average network throughput and average number of iterations. Therefore, in the following, we use Case 3 with uniform crossover.

Figures 4.4a and 4.4b present the average throughput and average number of iterations, respectively, for the throughput optimal, MFS, and ProbFS schedulers as well as the throughput maximizing GA based scheduling scheme with Case 3, uniform crossover, $N = 5, N_{best} = 50$, $\mu_m = 0.01$, and varying N_{pop}. Increasing the population size decreases the average number of iterations and increases the throughput; however, the computational cost of a single iteration increases since more chromosomes have to be processed in each iteration. Moreover, the performance improvement decreases as the population size increases. The results indicate that setting $N_{pop} = 100$ is a reasonable decision in terms of both performance criteria. Furthermore, we can also



(a) Average network throughput

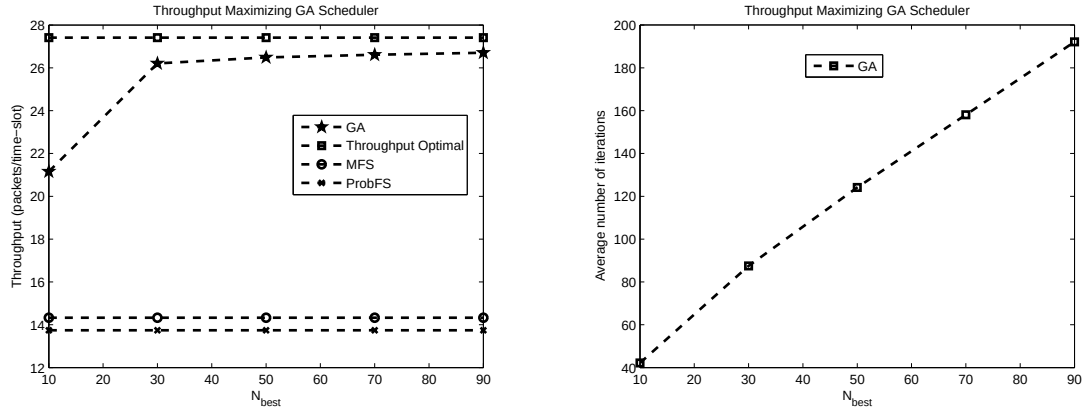
(b) Average number of iterations

Figure 4.4. Average network throughput and average number of iterations for throughput maximizing GA based scheduling scheme with Case 3, $N = 5$, N_{best}=50, μ_m=0.01, uniform crossover, and varying population size.

see that our GA based approach outperforms the other suboptimal schedulers, i.e. MFS and ProbFS, with several orders of magnitude, while at the same time yielding very close performance to the throughput optimal scheduler.

Figures 4.5a and 4.5b show the average network throughput and average number of iterations, respectively, for the throughput optimal, MFS, and ProbFS schedulers as well as the throughput maximizing GA based scheduling scheme with Case 3, uniform crossover, $N = 5$, $N_{pop} = 100$, $\mu_m = 0.01$, and varying N_{best}. As N_{best} increases, the increase in throughput decreases after some point, whereas the average number of iterations increases linearly with increasing N_{best}. The results point out that setting $N_{best} = 50$ is feasible when we take both performance criteria into account.

Figures 4.6a and 4.6b exhibit the average network throughput and average number of iterations for the throughput optimal, MFS, and ProbFS schedulers as well as the throughput maximizing GA based scheduling scheme with Case 3, uniform crossover, $N = 5, N_{pop} = 100$, $N_{best} = 50$, and varying μ_m. Both average throughput and average number of iterations initially increase as μ_m increases; however, they decrease after some point. Setting μ_m to a too high value can result in the introduction of unneces-



(a) Average network throughput

(b) Average number of iterations

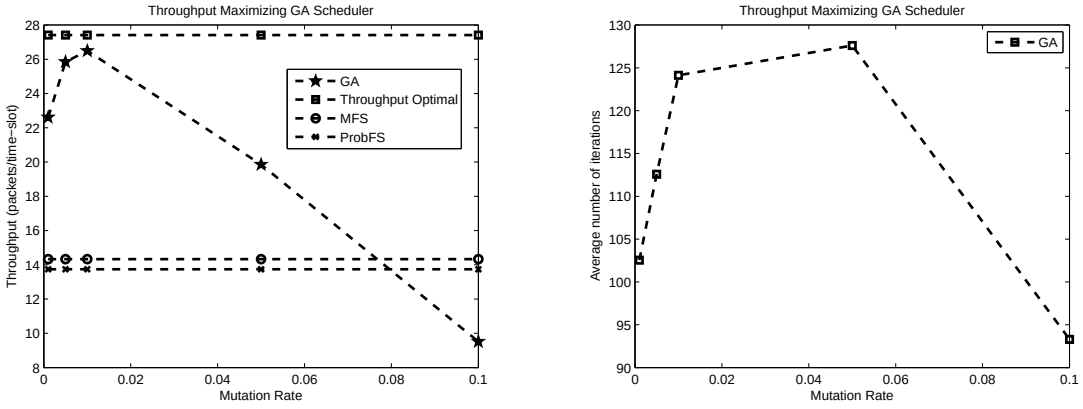
Figure 4.5. Average network throughput and average number of iterations for the GA based throughput maximizing scheduling scheme with Case 3, $N = 5$, N_{pop}=100, μ_m=0.01, uniform crossover, and varying N_{best}.

sarily large noise to the current solution; hence, the solution space can even get further away from the good solution area while trying to get out of the local optimum, and thereby yielding even worse performance than the MFS and ProbFS schedulers. The results indicate that $\mu_m = 0.01$ yields throughput very close to the optimal value with a reasonable number of iterations.

Besides, the parameters N_{pop}, N_{best}, and μ_m influence the performance depending on the number of cognitive nodes N. In Table 4.4, we have outlined the values for these parameters that we empirically found to yield near optimal results for varying values of N. We have used Case 3 and uniform crossover in the simulations of the throughput maximizing GA based scheduling scheme in Table 4.4. We can see that our GA based scheduling scheme yields much better performance than the MFS and ProbFS schedulers, and very close performance to the optimal scheduler for all $N = 5, 10, \ldots, 30$. Note that the parameters in this table are set so that the suboptimal scheduler gives very close results to the optimal one. If less number of iterations are desired at the expense of reduced throughput, then N_{pop}, N_{best}, and μ_m parameters can be adjusted depending on the number of SUs (N), with Table 4.4 serving as a baseline.

Table 4.4. Parameter settings for throughput maximizing GA scheduler with varying number of cognitive nodes.

N	5	10	15
N_{pop}	100	150	200
N_{best}	50	75	200
μ_m	0.01	0.01	0.01
Average throughput	26.48	26.87	25.61
Average number of iterations	127.20	330.99	885.22
Throughput optimal	27.41	27.55	26.29
MFS	14.33	15.16	16.06
ProbFS	13.74	14.50	15.97
N	20	25	30
N_{pop}	300	400	500
N_{best}	300	400	500
μ_m	0.001	0.001	0.001
Average throughput	25.89	26.12	25.99
Average number of iterations	1954.77	3076.82	5084.88
Throughput optimal	26.46	26.81	26.68
MFS	16.81	16.76	16.51
ProbFS	16.01	16.46	16.47



(a) Average network throughput

(b) Average number of iterations

Figure 4.6. Average network throughput and average number of iterations for the GA based scheduling scheme with with $N = 5$, $N_{pop} = 100$, $N_{best} = 50$, and varying μ_m.

We have also evaluated the performance of the GA based suboptimal scheduler that minimizes the scheduling delay. Initially, we have evaluated the performance of the parameter sets defined in Table 4.1. The resulting average scheduling delay and average number of iterations of the delay minimizing GA scheduler for $N = 5, N_{pop} = 100, N_{best} = 75, \mu_m = 0.01$, and single-point crossover are shown in Table 4.5. The results indicate that GA based schedulers yield much better scheduling delay performance than the MFS and ProbFS schedulers, while at the same time providing close to optimal performance. Furthermore, the results also indicate that Case 3 conduces the least scheduling delay and the least number of iterations. Hence, we employ Case 3 in the subsequent simulations.

Table 4.6 presents the average scheduling delay and average number of iterations values for $N = 5, N_{pop} = 100, N_{best} = 75, \mu_m = 0.01$ with Case 3 and single-point, two-point, as well as uniform crossover. The results show that GA based schedulers again have far better performance than the MFS and ProbFS schedulers, at the same time having very close to optimal performance. The results also reveal that uniform crossover outperforms the other two schemes both in terms of the average scheduling delay and the average number of iterations. Therefore, in the sequel, we use Case 3 with uniform crossover.

Table 4.5. Results for delay minimizing GA scheduler for $N = 5, N_{pop} = 100, N_{best} = 75, \mu_m = 0.01$.

Case	Average scheduling delay	Average number of iterations
1	0.039	94.43
2	0.038	92.55
3	0.023	87.24
4	0.040	92.29
Delay optimal	0.00095	–
MFS	0.60	–
ProbFS	0.55	–

Table 4.6. Crossover type comparison for delay minimizing GA scheduler for $N = 5, N_{pop} = 100, N_{best} = 75, \mu_m = 0.01$ with Case 3.

Crossover Type	Average delay	Average number of iterations
Single Point	0.023	87.24
Two Point	0.022	86.68
Uniform	0.0099	82.38
Delay optimal	0.00095	–
MFS	0.60	–
ProbFS	0.55	–

Table 4.7. Average scheduling delay and average number of iterations for the GA based scheme with Case 3, uniform crossover, $N = 5, N_{best} = 75$, $\mu_m = 0.01$, and varying population size.

Scheduler type	Average delay	Average number of iterations
Delay Minimizing GA, $N_{pop} = 20$	0.073	242.29
Delay Minimizing GA, $N_{pop} = 40$	0.056	93.38
Delay Minimizing GA, $N_{pop} = 60$	0.034	88.03
Delay Minimizing GA, $N_{pop} = 80$	0.018	85.37
Delay Minimizing GA, $N_{pop} = 100$	0.009	82.38
Delay Minimizing GA, $N_{pop} = 120$	0.0009	81.51
Delay optimal	0.00095	–
MFS	0.6	–
ProbFS	0.55	–

Table 4.7 presents the average scheduling delay for the delay optimal, MFS, and ProbFS schedulers as well as the delay minimizing GA based scheduling scheme with Case 3, uniform crossover, $N = 5, N_{best} = 75$, $\mu_m = 0.01$, and varying N_{pop}. The table also shows the average number of iterations for the GA based schemes. The results indicate that the GA based scheduling scheme results in much better performance than the MFS and ProbFS schedulers, even with a low population size. Moreover, the scheduling delay and the average number of iterations decrease as N_{pop} increases; however, the computational cost of a single generation also increases. Furthermore, the rate of decrease in the average number of iterations decreases as N_{pop} increases, whereas the average scheduling delay values diminish almost linearly. The results point out that setting $N_{pop} = 100$ is a reasonable decision in terms of both performance criteria.

Table 4.8 shows the average scheduling delay for the delay optimal, MFS, and ProbFS schedulers as well as the delay minimizing GA based scheduling scheme with Case 3, uniform crossover, $N = 5, N_{pop} = 100$, $\mu_m = 0.01$, and varying N_{best}. The

Table 4.8. Average scheduling delay and average number of iterations for the GA based scheduling scheme with Case 3, uniform crossover, $N = 5, N_{pop} = 100$, $\mu_m = 0.01$, and varying N_{best}.

Scheduler type	Average delay	Average number of iterations
Delay Minimizing GA, $N_{best} = 15$	0.0108	23.89
Delay Minimizing GA, $N_{best} = 30$	0.0107	39.00
Delay Minimizing GA, $N_{best} = 45$	0.0106	53.93
Delay Minimizing GA, $N_{best} = 60$	0.0104	68.98
Delay Minimizing GA, $N_{best} = 75$	0.0099	84.12
Delay Minimizing GA, $N_{best} = 90$	0.0096	99.2
Delay optimal	0.00095	–
MFS	0.6	–
ProbFS	0.55	–

table also shows the average number of iterations for the GA based schemes. The results reveal that the GA based scheduling scheme yields far better scheduling delay performance than the MFS and ProbFS schedulers even with small values of N_{best}. We can also see that the scheduling delay decreases as N_{best} increases. We can also see in Table 4.8 that the average number of iterations increases as N_{best} increases. As in the throughput maximizing GA scheduler, the increase in the average number of iterations is linear as N_{best} increases. The results point out that setting $N_{best} = 75$ is feasible when we take both performance criteria into account.

Table 4.9 shows the average scheduling delay for the delay optimal, MFS, and ProbFS schedulers as well as the delay minimizing GA based scheduling scheme with Case 3, uniform crossover, $N = 5, N_{pop} = 100$, $N_{best} = 75$ and varying μ_m. Again, the results show that the scheduling delay performance of the GA based scheme is much better than the ones of MFS and ProbFS, while at the same time being close to the optimal delay performance. Table 4.9 also shows the average number of iterations for

Table 4.9. Average scheduling delay and average number of iterations for the GA based scheduling scheme with Case 3, uniform crossover, $N = 5, N_{pop} = 100$, $N_{best} = 75$, and varying μ_m.

Scheduler type	Average delay	Average number of iterations
Delay Minimizing GA, $\mu_m = 0.001$	0.019	89.82
Delay Minimizing GA, $\mu_m = 0.005$	0.015	84.81
Delay Minimizing GA, $\mu_m = 0.01$	0.009	84.12
Delay Minimizing GA, $\mu_m = 0.05$	0.012	87.74
Delay Minimizing GA, $\mu_m = 0.1$	0.02	94.87
Delay Minimizing GA, $\mu_m = 0.2$	0.0345	98.24
Delay optimal	0.00095	–
MFS	0.6	–
ProbFS	0.55	–

the GA based scheme with the same parameters. We can see that both the average scheduling delay and the average number of iterations initially decrease as μ_m increases; nevertheless, they both increase after some point. The results indicate that $\mu_m = 0.01$ yields reasonable performance in terms of both criteria when we compare it to the other mutation rates.

Besides, as in the throughput maximizing GA based scheduler, the parameters N_{pop}, N_{best}, and μ_m influence the performance depending on the number of cognitive nodes N. In Table 4.10, we have outlined the values for these parameters that we have empirically found to yield satisfactory results with reasonable number of iterations for varying values of N. We have used Case 3 and uniform crossover in the simulations in Table 4.10.

All in all, both of our proposed GA based schedulers achieve very close performance to their optimal scheduler counterparts while at the same time operating with

Table 4.10. Parameter settings for delay minimizing GA scheduler with varying number of cognitive nodes.

N	5	10	15
N_{pop}	100	150	200
N_{best}	75	100	150
μ_m	0.01	0.01	0.01
Average delay	0.0099	0.212	0.538
Average number of iterations	84.12	132.27	316.97
Delay optimal	0.00095	0.203	0.509
MFS	0.60	1.27	1.96
ProbFS	0.55	1.23	1.87
N	20	25	30
N_{pop}	250	300	350
N_{best}	200	250	300
μ_m	0.001	0.001	0.001
Average delay	1.11	1.60	2.11
Average number of iterations	613.34	987.14	1112.45
Delay optimal	1.01	1.51	2.01
MFS	2.66	3.41	4.09
ProbFS	2.55	3.21	3.94

much lower complexity. However, the average number of iterations in the simulation results reveal that our GA based schedulers are computationally more costly than the MFS and ProbFS schedulers explained in Chapter 3. Nevertheless, when they are compared with respect to the throughput and delay performance, we can see that our GA based schedulers are approximately twice better than the MFS and ProbFS schedulers. Moreover, our GA based schedulers are computationally more efficient than the classical branch and bound algorithms that are used to solve binary integer programming problems [70, 71]. Therefore, our GA based schedulers present a very reasonable trade-off between computational complexity and performance, hence addressing the open research issue identified in Chapter 3. Hence, we can conclude that our GA based schedulers are more suitable for slowly varying spectral environments, while MFS and ProbFS schedulers are more suitable for very swiftly changing spectral environments. Considering that IEEE 802.22 networks [68] operate on the TV broadcast bands which are slowly varying, we can confidently conclude that our GA based schedulers can operate in realistic network settings, and provide useful solutions to the open research problem pinpointed in Chapter 3.

5. THROUGHPUT MAXIMIZING AND FAIR SCHEDULERS

The work in Chapters 3 and 4 relies on the interference temperature (IT) model proposed by FCC in [1]. Because the IT model requires the measurement of interference temperature and setting of an upper interference limit on the entire frequency band, it spurred a lot of debate since its inception and received both positive and negative comments. Most of the negative comments were due to the complexity of its practical implementation at the physical layer. Finally, FCC abandoned the IT concept [6].From this chapter onwards, we distance ourselves from the IT debate and rely on a much simpler physical layer model. Instead of measuring the interference temperature at all the measurement points and setting an upper limit for each frequency band, the CBS in our model only needs to determine whether PUs are actively using a particular frequency or not. There is a maximum tolerable interference power for each active PU as opposed to each frequency band in the IT model. This can be accomplished using conventional physical and MAC layer spectrum sensing mechanisms in the CRN literature [7, 8].

The major merit of our proposed scheduling scheme is that it is a very general model accomplishing many tasks at the same time. Specifically, all schedulers proposed in this chapter achieve the following:

(i) Frequency allocation
(ii) Time slot allocation
(iii) Data rate allocation
(iv) Power allocation
(v) Taking into account possibly multiple antennas for data transmission
(vi) Multi-user environment
(vii) Multi-channel environment
(viii) Considering channel heterogeneity, where not only the availability of the channels

(frequencies), but also the information about "how much available a particular frequency is for a particular SU in terms of maximum allowed transmission power and data rate" differ for each SU and channel (frequency) pair

(ix) Taking into account numerous physical layer information such as fading, path loss, mobility, and time-varying channels

(x) Guaranteeing reliable communication of SUs with the CBS

(xi) Ensuring that PUs are not disturbed by SU transmissions

(xii) Ensuring that no collisions occur among SUs

(xiii) A temporal notion of fairness together with throughput fairness (in fair schedulers)

(xiv) Taking into account the recently experienced throughput values during the current scheduling decision and thereby providing flexibility to increase throughput by sacrificing from fairness (in fair schedulers)

To the best of our knowledge, none of the previous work in the literature encompasses all of the above features. Contents of this chapter partially appeared in [79].

5.1. Problem Formulations and Proposed Solutions

We again focus on a time-slotted centralized CRN cell, as in Figure 3.1, where the CBS is responsible for the overall coordination of the SUs. The scheduler resides at the CBS and decides on how many packets and with which frequency each SU will transmit in each time slot.

Figure 5.1 demonstrates our cognitive scheduling method. $SU(f^1)$ and $PU_1(f^1)$ show that SU and PU_1 currently use frequency f^1, while $PU_2(f^2)$ illustrates that PU_2 uses frequency f^2. That is to say, PU_1 and PU_2 will be disturbed if any SU transmits using frequency f^1 and f^2, respectively, and the interference power received by each PU is above its maximum tolerable interference power. We represent the maximum tolerable interference power of PU_j for frequency f by $P^{fj}_{IF_{max}}$, where IF stands for "interference". In other words, the SU in Figure 5.1 does not disrupt PU_2 because their operation frequency is different, while it is possible for PU_1 to be disturbed by

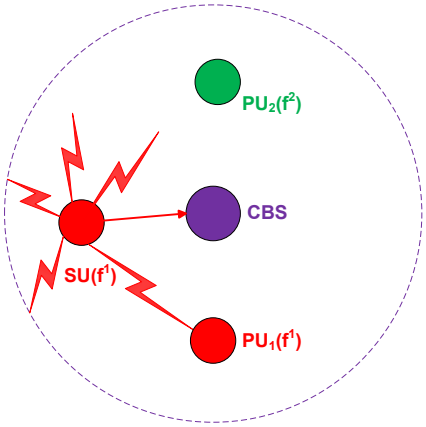


Figure 5.1. Framework for our cognitive scheduling mechanism.

the SU since they currently utilize the same frequency f^1. The goal in this work is to determine the transmission power and consequently the data rate of every SU for every frequency and time slot such that the PUs whose communication are active in that particular frequency are not disrupted.

Let us represent by u_{it} the number of packets transmitted by SU i in time slot t, by x_{it} the number of packets in the buffer of SU i at the beginning of time slot t, and by f_{it} the frequency used by SU i in time slot t. We denote the locations of SU i and PU j in time slot t by L_{it} and L_{jt}, respectively, and the location of the CBS by L_{CBS}. Furthermore, we denote the fading coefficient of the channel between SU i and the CBS in time slot t by h_{i0t} and the fading coefficient of the channel between SU i and PU j in time slot t by h_{ijt}. Consequently, the vector of buffer states for a total number of N cognitive nodes is $\overline{x_t} = [x_{1t}, x_{2t}, \cdots, x_{Nt}]$, the vector of transmitted packets is $\overline{u_t} = [u_{1t}, u_{2t}, \cdots, u_{Nt}]$, and the vector of frequencies used by SUs is $\overline{f_t} = [f_{1t}, f_{2t}, \cdots, f_{Nt}]$. Additionally, the vector of SU locations is $\overline{L_t^{SU}} = [L_{1t}, L_{2t}, \cdots, L_{Nt}]$, and the vector of PU locations is $\overline{L_t^{PU}} = [L_{1t}, L_{2t}, \cdots, L_{Mt}]$, where M is the total number of PUs in the coverage area of the CBS. Moreover, the matrix of fading coefficients for the channels between SUs and PUs is symbolized by $\mathbf{h_t^{SU,PU}} = [h_{ijt}]$, which is an $N \times M$ matrix. Similarly, the vector of fading coefficients for the channels between the SUs and the CBS is $\overline{\mathbf{h_t^{SU,CBS}}} = [h_{10t}, h_{20t}, \cdots, h_{N0t}]$. We presume that the CBS knows $\overline{L_t^{SU}}$, $\overline{L_t^{PU}}$, L_{CBS}, $\mathbf{h_t^{SU,PU}}$, and $\overline{\mathbf{h_t^{SU,CBS}}}$. As a consequence, the scheduler's mapping is $\gamma(t) : [\overline{x_t}, \overline{L_t^{SU}}, \overline{L_t^{PU}}, L_{CBS}, \mathbf{h_t^{SU,PU}}, \overline{\mathbf{h_t^{SU,CBS}}}] \rightarrow [\overline{f_t}, \overline{u_t}]$.

FCC has required CR devices to use geolocation in conjunction with database consultation in [80]. Hence, our assumption that the CBS knows $\overline{L_t^{SU}}$, $\overline{L_t^{PU}}$, and L_{CBS} is in line with this requirement. Authors in [81], for instance, also point out the positive impact of location awareness in CRNs. The knowledge about other physical layer parameters such as $\mathbf{h_t^{SU,PU}}$, and $\overline{\mathbf{h_t^{SU,CBS}}}$ is also the assumption of other works in the literature [82, 83]. For instance, SUs can estimate channel gains between SUs and PUs ($\mathbf{h_t^{SU,PU}}$ in our case) by employing sensors near all receiving points and make them available at the central controller. Values for $\overline{\mathbf{h_t^{SU,CBS}}}$ can also be estimated in a similar way. The fact that our scheduling model is designed for a centralized (infrastructure-based) CRN setting facilitates the implementation for the gathering of such physical layer parameters.

5.1.1. Throughput Maximizing Scheduler (TMS)

We formulate in Equations 5.1-5 the scheduling problem that maximizes the expected value of the total network throughput, while assuring that the PUs in the service area of the CBS are not disrupted, and reliable communication between SUs and the CBS is maintained:

$$\max_{\overline{u_t},\overline{f_t}} E\{\sum_{i=1}^{N} u_{it}\} \tag{5.1}$$

$$s.t.$$

$$P_{r_j}^{ft} \leq P_{IF_{max}}^{fj}; \forall j \in \Phi_{CBS}^{ft}, \forall f \in \mathcal{F} \tag{5.2}$$

$$u_{it} = B \times \frac{T_s}{S} \times \ln(1 + \frac{P_{r_{CBS}}^{ift}}{\zeta}); \forall i \in \mathcal{N} \tag{5.3}$$

$$f_{it} \neq f_{i't}; \forall i, i' \in \mathcal{N}, i \neq i' \tag{5.4}$$

$$u_{it} \leq x_{it}; \forall i \in \mathcal{N} \tag{5.5}$$

where $\mathcal{F} = \{1, 2, \cdots, F\}$ denotes the set of F frequencies, $\mathcal{N} = \{1, 2, \cdots, N\}$ shows the set of N SUs, $P_{IF_{max}}^{fj}$ represents the maximum tolerable interference power of PU j for frequency f, and $P_{r_j}^{ft}$ symbolizes the power received by PU j through frequency f in time slot t. Moreover, Φ_{CBS}^{ft} denotes the set of PUs that are actively utilizing

frequency f in the coverage area of the CBS in time slot t, and $P^{ift}_{r_{CBS}}$ is the power received by the CBS owing to the possible transmission of SU i using frequency f in time slot t. Besides, S is the packet size, B is the bandwidth, ζ is the sum of interference power from the PUs to SU i and noise power, and T_s is the time slot length. Interference power from the PUs to the SUs and to the CBS may depend on many factors such as the spectrum occupancy of the PUs as well as the locations. Even when this interference is modeled through different variables, an expression for the maximum transmission rate U_{if} for each SU i and frequency f can be obtained. Once the U_{if} values are obtained, our ILP formulations in the second stages of our throughput maximizing, (weighted) max-min fair, and proportionally fair schedulers remain the same. Therefore, for notational simplicity, we represent the interference power from the PUs and the noise power by a single variable ζ. On the other hand, units for parameters S, B, and T_s are bits/packet, bits/second, and seconds/time slot, respectively.

In the formulation from Equation 5.1 to Equation 5.5, the objective in Equation 5.1 maximizes the expected value of the total number of packets transmitted by all the cognitive users. Besides, constraint in Equation 5.2 guarantees that interference power values received by PUs because of SU transmissions adhere to the tolerable limits. Note here that constraint in Equation 5.2 imposes a restriction on the interference power received by PUs. The primary goal of SUs is to communicate with the CBS. In doing so, they may increase the interference received by PUs. Therefore, the maximum tolerable interference requirement of the PUs translates to a maximum transmission power constraint on the SUs because in order to ensure that the interference perceived by the PUs is within the tolerable limits, SUs need to adjust their transmission powers while communicating with the CBS. By considering the channel conditions between SUs and PUs, we can impose a maximum transmission power constraint on SUs. We can then convert this maximum transmission power constraint to a maximum data rate constraint on SUs through Shannon's capacity function for Gaussian channels. Constraint in Equation 5.3 serves this purpose via ensuring that reliable communication between SU i and the CBS is achieved by having the scheduler to choose the number of packets transmitted, u_{it}, equal to the Shannon capacity function for a Gaussian channel

[69]. Recent advances in coding (Turbo codes and LDPC) make it feasible to achieve performance close to Shannon capacity using codes over finite block lengths. Besides, the uncoded systems also follow the exponential relationship between transmission rate and transmit power [84]. All the optimization problems we formulate in this work take as input the maximum possible transmission rate of each SU for each frequency. If the relationship between transmission rate and power is not exponential, then the calculation of these maximum possible transmission rates will be different; however, our formulated optimization problems in this work will remain the same. Therefore, even in cases where the relationship between transmission rate and power is not exponential, all of our optimization problems are still valid. Furthermore, Equation 5.4 ensures that at most one SU can transmit using a certain time slot and frequency combination, and Equation 5.5 represents the fact that an SU cannot transmit more than the number of packets in its buffer at the beginning of the time slot.

In the initial step of our solution to the problem in Equations 5.1-5, we find the maximum permissible transmission power for each SU i and frequency f in time slot t, which is represented here by P_{xmt}^{ift}. We use free space path loss and fading in modeling the channels between SUs and PUs, as well as the ones between the CBS and SUs. Therefore, the following relationship holds between P_{xmt}^{ift} and P_{rj}^{ft}:

$$P_{rj}^{ft} = P_{xmt}^{ift} \times |G_{ift}|^2 \tag{5.6}$$

$$G_{ift} = \max_{j \in \Phi_{CBS}^{ft}} (\frac{\lambda_f}{4\pi d_{ijt}} \times |h_{ijt}|) \tag{5.7}$$

where d_{ijt} equals the distance between SU i and PU j in time slot t, λ_f is the wavelength of frequency f, and Φ_{CBS}^{ft} symbolizes the set of PUs that are in the coverage area of the CBS and that are carrying out their communication using frequency f in time slot t. Moreover, h_{ijt} denotes the fading coefficient of the channel between SU i and PU j in time slot t. In particular, $(\frac{\lambda_f}{4\pi d_{ijt}})^2$ refers to the path loss of the channel between SU i and PU j in time slot t as a result of the free space path loss formula. Hence, we denote by $|G_{ift}|$ the maximum channel gain of SU i and the PU that has the highest channel gain with this SU among all PUs which are actively using frequency f in time

slot t. Notice that we consider here the interference from SUs to PUs (not vice versa) because the goal of the scheduler is to govern the transmission of SUs to the CBS without causing any harmful interference to PUs (without disturbing PUs). Recall that we have represented the interference from PUs to SUs and noise power by the variable ζ.

Let us assume, without loss of generality (w.l.o.g), that $P^{fj}_{IF_{max}}$ is constant for all j. Hence, in the sequel, let us use $P^{f}_{IF_{max}}$ in lieu of $P^{fj}_{IF_{max}}$. Moreover, assume for simplicity, and yet w.l.o.g., that $S = B \times T_s$. Recall that P^{ift}_{xmt} denotes the maximum permissible transmission power for SU i and frequency f in time slot t. Therefore, the expressions from Equation 5.8 to Equation 5.12 in the following hold:

$$P^{ift}_{xmt} = \frac{P^{f}_{IF_{max}}}{|G_{ift}|^2} \tag{5.8}$$

$$|G_{i0t}| \triangleq \frac{\lambda_f}{4\pi d_{i0t} \times h_{i0t}} \tag{5.9}$$

$$P^{ift}_{r_{CBS}} = P^{ift}_{xmt} \times |G_{i0t}|^2 \tag{5.10}$$

$$P^{ift}_{r_{CBS}} = P^{f}_{IF_{max}} \times (\frac{G_{i0t}}{G_{ift}})^2 \tag{5.11}$$

$$U_{ift} = \Big\lfloor \ln(1 + P^{f}_{IF_{max}} \times (\frac{G_{i0t}}{G_{ift} \times \sigma})^2) \Big\rfloor \tag{5.12}$$

where d_{i0t} is the distance between SU i and the CBS in time slot t, and U_{ift} is the maximum number of packets that can be transmitted by SU i using frequency f in time slot t. Equation 5.8 converts the maximum tolerable interference power constraint ($P^{f}_{IF_{max}}$) at PUs to maximum transmission power constraint (P^{ift}_{xmt}) at SUs. Equation 5.9 defines the channel gain between SU i and the CBS in time slot t, denoted by G_{i0t}. Equation 5.10 relates the power that would be received by the CBS due to the transmission of SU i using frequency f in time slot t ($P^{ift}_{r_{CBS}}$) if SU i uses its maximum permissible transmission power for frequency f and time slot t (P^{ift}_{xmt}). Equations 5.8-10 are valid since the maximum possible value for $P^{ft}_{r_j}$ is $P^{f}_{IF_{max}}$ because of Equation 5.2. Equation 5.11 follows directly from Equations 5.8, 5.9, and 5.10. Finally, Equation 5.12 is due to Shannon's capacity function for Gaussian channels, where $P^{ift}_{r_{CBS}}$ is replaced by Equation 5.12. The floor operation $\lfloor . \rfloor$ in Equation 5.12 is necessary since U_{ift} can

naturally only take integer values.

Assume that the network conditions, i.e., PU spectrum occupancies, PU and SU locations, and all the channel fading coefficients, are small enough not to have any influence on the U_{ift} values for a duration of T time slots in the considered centralized CRN cell. In other words, assume that the U_{ift} value for SU i and frequency f remain the same for a duration of T time slots, which is equal to the scheduling period during which the network conditions are fairly stable. Due to the floor operation $\lfloor . \rfloor$ in Equation 5.12, the scheduling period length T does not obligate the PU spectrum occupancies as well as the PU and SU locations to remain constant in that time period, but only requires that the change in their values does not alter U_{ift} (related to SU i and frequency f) for a period of T time slots. The value of T, in general, hinges upon the characteristics of the spectral environment. For instance, a slowly varying spectrum environment like the TV bands used by an IEEE 802.22 network, allows T to have a fairly large value. Therefore, instead of U_{ift}, let us use U_{if} , which represents the maximum number of packets that can be transmitted by SU i using frequency f in every time slot for a duration of T time slots. After U_{if} values are obtained through the analysis in Equations 5.6-12, we execute the following binary integer linear program (ILP) to solve our formulated throughput maximizing scheduling problem:

$$\max\left(\sum_{i=1}^{N}\sum_{f=1}^{F}\sum_{t=1}^{T}\frac{U_{if}X_{ift}}{T}\right) \tag{5.13}$$

$$s.t.$$

$$\sum_{f=1}^{F}\sum_{t=1}^{T}X_{ift} \geq 1; \forall i \in \mathcal{N} \tag{5.14}$$

$$\sum_{i=1}^{N}X_{ift} \leq 1; \forall f \in \mathcal{F}, \forall t \in \mathcal{T} \tag{5.15}$$

$$\sum_{f=1}^{F}X_{ift} \leq a_i; \forall i \in \mathcal{N}, \forall t \in \mathcal{T} \tag{5.16}$$

$$X_{ift} \in \{0,1\}; \forall i \in \mathcal{N}, \forall f \in \mathcal{F}, \forall t \in \mathcal{T} \tag{5.17}$$

where $\mathcal{T}$ denotes the set of T time slots in a scheduling period; i.e., $\mathcal{T} = \{1, 2, \cdots, T\}$.

In addition, X_{ift} is a binary decision variable such that $X_{ift} = 1$ if SU i transmits with frequency f in time slot t and 0 otherwise, and a_i is the number of transceivers (antennas) of SU i. In this formulation, Equation 5.14 guarantees that at least one time slot is assigned to every SU and hence provides a temporal notion of fairness, while Equation 5.15 ensures that at most one SU can transmit in a particular time slot and frequency, and hence obviates collisions between SUs. Consider a situation where two SUs transmit with a certain frequency f in a particular time slot t. This implies that two SUs will contribute to the value of $P_{r_j}^{ft}$ in Equation 5.2. However, when we calculate the maximum transmission power for an SU by considering the interference that can occur at PUs, we consider the interference created by only that SU. Therefore, having more than one SU transmit in the same frequency and time slot may increase the aggregate interference experienced by PUs above the maximum tolerable interference limit, which is $P_{IF_{max}}^{f}$. Hence, besides avoiding collisions among SUs, Equation 5.15 serves the purpose of guaranteeing that the aggregate interference at the PUs is within the tolerable threshold. Moreover, Equation 5.16 represents the fact that an SU i cannot transmit at the same time using frequencies more than the number of its transceivers (antennas), a_i, because each transceiver can tune to at most one frequency at a time. MIMO technology can be used to have multiple antennas at the SUs. We assume that channel conditions of each antenna of a particular SU are the same for a particular frequency. We make this assumption to isolate us from the possible impacts of different channel conditions for different antennas and concentrate on the performance impact of the number of antennas. Note here that our formulation does not mandate the SUs to have multiple antennas; in other words, even when each SU has a single antenna, our formulation in Equations 5.13-16 is still valid since $a_i = 1$ $\forall i \in \mathcal{N}$. After the scheduling decisions about U_{if} and X_{ift} values are made, an SU i for which $X_{ift} = 1$ transmits $\min(x_{it}, U_{if})$ number of packets using frequency f in time slot t. As in Chapter 3, we consider traffic in which all flows are continuously backlogged so that the resulting throughput is completely related to the scheduling process and channel conditions without any variation because of the traffic fluctuation. That is to say, in the simulations part of this work, it is always true that $x_{it} \geq U_{if}$, $\forall i \in \mathcal{N}$, $\forall f \in \mathcal{F}$, $\forall t \in \mathcal{T}$; i.e., each SU always has sufficient number of packets waiting in its

buffer to be transmitted to the CBS. This situation is necessary in order to effectively evaluate the performance of the scheduling process by avoiding the possible influence of the traffic arrival process.

5.1.2. Max-Min Fair Scheduler (MMFS)

We formulate in Equations 5.18-19 the scheduling problem that maximizes the throughput of the SU experiencing the minimum throughput among all SUs, while ensuring that the communication of none of the PUs is disturbed, and reliable communication between SUs and CBS is achieved:

$$\max_{\overline{u_t},\overline{f_t}} \min_i E\{u_{it}\} \tag{5.18}$$

$$s.t.$$

$$(5.2), (5.3), (5.4), \text{ and } (5.5) \tag{5.19}$$

To solve the problem in Equations 5.18-19, we firstly implement the analysis in Equations 5.6-12 and obtain the U_{if} values. Secondly, we define R^i_φ, the aggregate average throughput of SU i in the last φ scheduling periods. The unit for R^i_φ is "packets per time slot". All of the R^i_φ values are initialized to 0 for all SUs. Let us define $\overline{R^i_\varphi}$, which is based on an exponentially weighted low pass filter, as follows:

$$\overline{R^i_\varphi} = (1 - \frac{1}{\min(k,\varphi)})R^i_\varphi + \frac{1}{\min(k,\varphi)} \frac{\sum_{f=1}^{F}\sum_{t=1}^{T} U_{if}X_{ift}}{T} \tag{5.20}$$

Here, $\frac{\sum_{f=1}^{F}\sum_{t=1}^{T} U_{if}X_{ift}}{T}$ denotes the throughput of SU i in the current scheduling period k and $\frac{1}{\min(k,\varphi)}$ is the weight given to it. On the other hand, $(1 - \frac{1}{\min(k,\varphi)})$ is the weight given to the value of R^i_φ, i.e. the aggregate average throughput of SU i at the start of the current scheduling period. Using an exponentially weighted low pass filter enables our scheduler to give more importance to the throughput experienced in the

more recent scheduling periods than the distant past. At the end of each scheduling period k, the value of R^i_φ is updated as $R^i_\varphi \leftarrow \overline{R^i_\varphi}$. Since both k and φ are constant in a particular scheduling execution, w.l.o.g we use φ instead of $\min(k, \varphi)$ in the rest of this work.

We then have the CBS execute the following mixed ILP in each scheduling period, which consists of T time slots:

$$\max Z \tag{5.21}$$

$$s.t.$$

$$Z \leq (1 - \frac{1}{\varphi})R^i_\varphi + \frac{1}{\varphi}\frac{\sum_{f=1}^{F}\sum_{t=1}^{T} U_{if}X_{ift}}{T} \tag{5.22}$$

$$(5.14), (5.15), \text{ and } (5.16) \tag{5.23}$$

$$X_{ift} \in \{0, 1\}; \forall i \in \mathcal{N}, \forall f \in \mathcal{F}, \forall t \in \mathcal{T} \tag{5.24}$$

where Equations 5.21 and 5.22 together maximize $\min_{i\in\mathcal{N}} \overline{R^i_\varphi}$. Keeping track of and using the aggregate throughput information R^i_φ in Equations 5.21-24 in lieu of maximizing the minimum throughput only in that scheduling period enables us to provide fairness in a longer time scale. If an SU suffers from low throughput due to PU activity in its vicinity, it can compensate for this loss in the subsequent scheduling periods due to the historical throughput information accumulated in R^i_φ. Embedding the information about the accrued throughput in the scheduling algorithm itself is a vital feature of our schedulers. The variable φ can be regarded as the window size during which the changes in the network conditions are considered to be important. If φ is too large, the scheduler will be too responsive to small changes in the network conditions. If φ is is too small, the scheduler will be inflexible in compensating for the temporary fluctuations in the network conditions. For instance, having $\varphi = 1$ connotes that merely the network conditions in the current scheduling period are taken into consideration without paying any attention to what has happened in the recent past.

Theorem 5.1. *Let Ω^{OPT}_{MaxMin} be the optimal value for the minimum throughput found*

by the max-min fair scheduler in Equations 5.21-24. Let Ω^{UB}_{MaxMin} *be the maximum possible value for* Ω^{OPT}_{MaxMin}*, and* Ω^{OPT}_{ThrMax} *be the total throughput found by the throughput maximizing scheduler in Equations 5.13-16. Then,* $\Omega^{OPT}_{MaxMin} \leq \Omega^{UB}_{MaxMin} = \frac{\Omega^{OPT}_{ThrMax}}{N}$.

Proof. Since Ω^{OPT}_{MaxMin} is the throughput of the SU with minimum throughput, the throughput of any other SU among the remaining $N-1$ SUs is at least Ω^{OPT}_{MaxMin}. Let Ω^{tot}_{MaxMin} denote the total throughput resulting from the max-min fair scheduler execution. Then, $\Omega^{tot}_{MaxMin} \geq \Omega^{OPT}_{MaxMin} + (N-1)\Omega^{OPT}_{MaxMin} = N \times \Omega^{OPT}_{MaxMin}$. Since $\Omega^{tot}_{MaxMin} \leq \Omega^{OPT}_{ThrMax}$, it follows that $\Omega^{OPT}_{MaxMin} \leq \Omega^{UB}_{MaxMin} = \frac{\Omega^{OPT}_{ThrMax}}{N}$. □

5.1.3. Weighted Max-Min Fair Scheduler (Weighted MMFS)

Max-min fair scheduling problem in Equations 5.21-24 maximizes the average throughput of the SU that has the minimum aggregate throughput; therefore, all SUs operate at a similar throughput level. However, in some practical cases, CBS operator may want to differentiate between different SUs and provide them with different levels of service by giving them priorities. We can quantify these priorities by associating a target weight with each SU such that the higher the target weight of an SU is, the more it is favored by the CBS scheduler in the frequency and time slot allocation.

We formulate in Equations 5.25-26 the scheduling problem that makes the throughput ratios of all SUs as close to their target weights as possible, while ensuring that the communication of none of the PUs is disturbed, and reliable communication between the SUs and the CBS is achieved:

$$\max_{\overline{u_t},\overline{f_t}} \min_i \frac{E\{u_{it}\}}{\eta_i} \tag{5.25}$$

$$s.t.$$

$$(5.2), (5.3), (5.4), \text{ and } (5.5) \tag{5.26}$$

where $0 \leq \eta_i \leq 1$ is the target weight of SU i; i.e., the target ratio of the throughput of SU i to the total throughput of all SUs in the CRN cell. Weights of all SUs in the

coverage area of the CBS sum up to 1; that is to say, $\sum_{i=1}^{N} \eta_i = 1$. If it were theoretically possible to assign all SUs their exact target weights, then their $\frac{E\{u_{it}\}}{\eta_i}$ values, which we refer to here by *normalized throughput values*, would all be equal to each other. Accordingly, the objective function in Equation 5.25 aims to maximize the normalized throughput value of the SU that has the minimum normalized throughput value, and thereby makes the throughput ratio of every SU as close to its target weight as possible. To solve the problem in Equations 5.25-26, as in the max-min fair scheduler, we firstly implement the analysis in Equations 5.6-12 and obtain the U_{if} values. Secondly, we calculate R^i_φ $\forall i \in \mathcal{N}$, and update them at the end of every scheduling period as in Equation 5.20. Finally, we have the CBS execute the following mixed ILP in each scheduling period, which consists of T time slots:

$$\max Z' \tag{5.27}$$

$$s.t.$$

$$Z' \leq \frac{(1-\frac{1}{\varphi})R^i_\varphi + \frac{1}{\varphi}\frac{\sum_{f=1}^{F}\sum_{t=1}^{T} U_{if}X_{ift}}{T}}{\eta_i} \tag{5.28}$$

$$(5.14),(5.15), \text{ and } (5.16) \tag{5.29}$$

$$X_{ift} \in \{0,1\}; \forall i \in \mathcal{N}, \forall f \in \mathcal{F}, \forall t \in \mathcal{T} \tag{5.30}$$

where the constraint in Equation 5.28 specifies Z', which is the normalized aggregate throughput of the SU that has the minimum normalized aggregate throughput among all SUs. The objective function in Equation 5.27 maximizes this minimum normalized aggregate throughput (Z'). In other words, Equations 5.27 and 5.28 together maximize $\min_{i\in\mathcal{N}} \frac{\overline{R^i_\varphi}}{\eta_i}$. As in the max-min fair scheduler, having $\varphi = 1$ implies that only the current scheduling period is taken into consideration. Similarly, keeping track of and using the aggregate throughput information R^i_φ in Equations 5.27-30 in lieu of maximizing the minimum normalized aggregate throughput merely in that scheduling period enables us to provide fairness in a longer time scale.

Notice here that in our formulations, max-min fair scheduler turns out to be a special case of the weighted max-min fair scheduler where $\eta_i = \frac{1}{N}\ \forall i \in \mathcal{N}$; i.e., all target weights are equal to each other. This result intuitively makes sense since the goal of the max-min fair scheduler is to essentially make the throughput of each SU as close to each other as possible.

5.1.4. Proportionally Fair Scheduler (PFS)

The notion of proportional fairness aims to provide a tradeoff between users' satisfaction and system revenue [85]. A data rate allocation is said to be proportionally fair if for any other feasible rate allocation, the aggregate of the proportional changes is not positive. Accordingly, proportional fairness is achieved by maximizing the sum of the logarithms of the data rates, which is equivalent to maximizing the product of the data rates [37].

We formulate in Equations 5.31-32 the scheduling problem that maximizes the products the SU throughput values, while ensuring that the communication of none of the PUs is disturbed, and reliable communication between SUs and the CBS is achieved:

$$\max_{\overline{u_t},\overline{f_t}} E\{\prod_{i=1}^{N} u_{it}\} \tag{5.31}$$

$$s.t.$$

$$(5.2), (5.3), (5.4), \text{ and } (5.5) \tag{5.32}$$

To solve the problem in Equations 5.31-32, as in the max-min fair and weighted max-min fair schedulers, we firstly implement the analysis in Equations 5.6-12 and obtain the U_{if} values. Secondly, we calculate $R^i_\varphi\ \forall i \in \mathcal{N}$ and update them at the end of every scheduling period using Equation 5.20. Finally, we have the CBS execute the following

mixed integer program for each scheduling period, which consists of T time slots:

$$\max \left(\sum_{i=1}^{N} \log\left((1-\frac{1}{\varphi})R_{\varphi}^{i} + \frac{1}{\varphi}\frac{\sum_{f=1}^{F}\sum_{t=1}^{T} U_{if}X_{ift}}{T}\right)\right) \tag{5.33}$$

$$s.t.$$

$$(5.14),(5.15), \text{ and } (5.16) \tag{5.34}$$

$$X_{ift} \in \{0,1\}; \forall i \in \mathcal{N}, \forall f \in \mathcal{F}, \forall t \in \mathcal{T} \tag{5.35}$$

where the objective function in Equation 5.33 maximizes the sum of logarithms of the aggregate throughput values of each SU, which would be updated according to Equation 5.20 at the end of the considered scheduling period if those particular X_{ift} values are used. Constraints are the same as in the other schedulers. The integer program in Equations 5.33-35 maximizes a nonlinear concave objective function subject to linear constraints, where the concavity is due to the logarithm operation in the objective function.

To put it in a nutshell, our proposed methods to solve the formulated scheduling problems can be summarized as follows:

Step 1. Find the values for U_{if} $\forall i \in \mathcal{N}$, $\forall f \in \mathcal{F}$ by doing the analysis in Equations 5.6-12.

Step 2. Given the values for N, F, T, φ, η_i, a_i, and U_{if}, find the values for X_{ift} $\forall i \in \mathcal{N}$, $\forall f \in \mathcal{F}$,$\forall t \in \mathcal{T}$ by executing either of the following binary/mixed integer programs:

(i) For throughput maximizing scheduler, execute Equations 5.13-16
(ii) For max-min fair scheduler, execute Equations 5.21-24
(iii) For weighted max-min fair scheduler, execute Equations 5.27-30
(iv) For proportionally fair scheduler, execute Equations 5.33-35

All scheduling problems formulated in this chapter are integer programming problems, which may in general be NP-Hard. However, some certain special cases of integer programming problems may be solvable in polynomial time. For instance, the integer programming formulations for maximum weighted matching and minimum spanning tree problems are solvable in polynomial time. In Chapter 6, we prove that the throughput maximizing scheduling problem is solvable in polynomial time, whereas the max-min fair, weighted max-min fair, and proportionally fair scheduling problems are NP-Hard in the strong sense. In this chapter, we propose a computationally efficient heuristic algorithm for the fair scheduling problems.

5.1.5. Our Proposed Heuristic Algorithm

In this section, we propose a heuristic algorithm for max-min, weighted max-min and proportionally fair schedulers. We outline our proposed heuristic in Figure 5.2 (FAIRSCH).

Ω_i indicates the summation of U_{if} values that have hitherto been assigned to SU i during the execution of the algorithm. Step 1 initializes all Ω_i values to 0. $availSUs[t]$ denotes the list of SUs that have an available (free) antenna for frequency assignment in time slot t. Since antennas of all SUs are available for all time slots at the beginning of the execution of the algorithm, $availSUs[t]$ values are initialized to $\mathcal{N}$ in Step 2. Steps 3-5 update the U_{if} values for the weighted max-min fair scheduler since constraint in Equation 5.28 is equivalent to Equation 5.22 when U_{if} values are scaled to $\frac{U_{if}}{\eta_i}$. FAIRSCH assigns each frequency and time slot pair sequentially to an SU. For the max-min and weighted max-min fair schedulers, Steps 8-9 assign frequency f and time slot t to the SU that has the minimum Ω value so far among the SUs that have an available antenna for time slot t. For the proportionally fair scheduler, Steps 11-13 assign frequency f and time slot t to the SU that gives the maximum value for the product of Ω_i values if frequency f and time slot t are assigned. $newObj[i]$ indicates the new objective function value, i.e., product of Ω_i values, if frequency f and time slot t are assigned to SU $\bar{i}$. In the case of proportionally fair scheduling, our heuristic algorithm selects the SU that gives the maximum value for this new objective

Require: $\mathcal{N}$, $\mathcal{F}$, $\mathcal{T}$, $\mathcal{A}_i$, U_{if}, η_i.

Ensure: X_{ift} values $\forall i \in \mathcal{N}, \forall f \in \mathcal{F}, \forall t \in \mathcal{T}$.

$\Omega_i \leftarrow 0, \forall i \in \mathcal{N}$

$availSUs[t] \leftarrow \mathcal{N}\ \forall i \in \mathcal{N}, \forall t \in \mathcal{T}$

if *Weighted Max-Min* **then**

    $U_{if} \leftarrow \dfrac{U_{if}}{\eta_i}, \forall i \in \mathcal{N}, \forall f \in \mathcal{F}$

end if

for $f = 1$ to F **do**

    for $t = 1$ to T **do**

        if *Max-Min* **or** *Weighted Max-Min* **then**

            $i^* \leftarrow \arg \min_{\bar{i} \in availSUs[t]} \Omega_{\bar{i}}$

        else

            if *Prop-Fair* **then**

                $newObj[\bar{i}] \leftarrow (\Omega_{\bar{i}} + U_{\bar{i}f}) \times \prod_{j \in \mathcal{N} - \{\bar{i}\}} \Omega_j, \forall \bar{i} \in availSUs[t]$

                $i^* \leftarrow \arg \max_{\bar{i} \in availSUs[t]} newObj[\bar{i}]$

            end if

        end if

        $X_{i^*ft} \leftarrow 1$

        $\Omega_{i^*} \leftarrow \Omega_{i^*} + \dfrac{U_{i^*f}}{T}$

        if $\sum_{f=1}^{F} X_{i^*ft} = a_{i^*}$ **then**

            $availSUs[t] \leftarrow availSUs[t] - \{i^*\}$

        end if

    end for

end for

Figure 5.2. FAIRSCH (Our proposed heuristic algorithm).

function. Step 16 makes the assignment to the selected SU i^*. If all antennas of SU i^* are assigned some frequency for time slot t, then steps 17-18 remove this SU from the list of available SUs for time slot t ($availSUs[t]$). The algorithm terminates after all frequency and time slot pairs are assigned to some SU. Note here that FAIRSCH is a greedy algorithm since it selects the SU that yields the best possible objective function value in each iteration. In other words, it aims to increase the throughput of the SU with minimum (normalized) throughput in (weighted) max-min fair scheduling, whereas it aims to maximize the product of the throughput values in proportionally fair scheduling.

Computational Complexity: In the case of (weighted) max-min fair scheduling, FAIRSCH scans the list of SUs in $availSUs[t]$, the size of which is at most N during the assignment of frequency f and time slot t. Since there are F frequencies and T time slots, the complexity of FAIRSCH in the case of (weighted) max-min fair scheduling is $O(NFT)$. In the case of proportionally fair scheduling, $availSUs[t]$ is scanned in the calculation of each value for $\overline{i}$ in Step 12. Therefore, the complexity of Step 12 is $O(N^2)$. Hence, the complexity of FAIRSCH in the case of proportionally fair scheduling is $O(N^2FT)$.

5.2. Numerical Evaluation

We focus on a CRN cell with 600 meters of radius and simulate it using Java. We elicit the U_{if} values in each set of simulations for 5000 scheduling periods. We then solve the optimization problems for throughput maximizing, max-min fair, and weighted max-min fair schedulers using CPLEX [5] and proportionally fair scheduling problem using KNITRO [86]. We compare the performance of these schedulers by using the same set of U_{if} values in each comparison. Every scheduling period consists of $T = 10$ time slots and each time slot lasts for $T_s = 100$ milliseconds. According to the IEEE 802.22 standard [68], it is imperative that SUs vacate a spectrum band within two seconds from the appearance of the licensed owner (PU) of that particular band. Therefore, we take each scheduling period equal to one second (hence consisting of 10 time slots) since it is sufficient for proper operation. Additionally, we examine

our methods for additive white gaussian noise (AWGN) channels; in other words, we take $h_{ijt} = h_{i0t} = 1, \forall i \in \mathcal{I}, \forall j \in \Phi_{CBS}^{ft}$, and $\forall t \in \mathcal{T}$. Furthermore, $\zeta = 10^{-6}$ and the maximum tolerable interference power of active PUs is $P_{IF_{max}}^{f} = 10$ milliwatts $\forall f \in \mathcal{F}$.

Initial locations of the SUs and the PUs in the CRN cell are uniformly randomly distributed. We use random waypoint mobility model to simulate the movement of the SUs and the PUs. In line with this model, each SU/PU selects a target location in the cell in a uniformly random manner and moves towards this point with a constant velocity. After reaching its target location, each node stays there for a certain amount of time and then chooses another target point. This movement pattern continues in this way for each SU/PU until the end of simulation. We set the staying duration between movement periods as 10 seconds. We represent the velocity of the SUs by V_s and the velocity of the PUs by V_p.

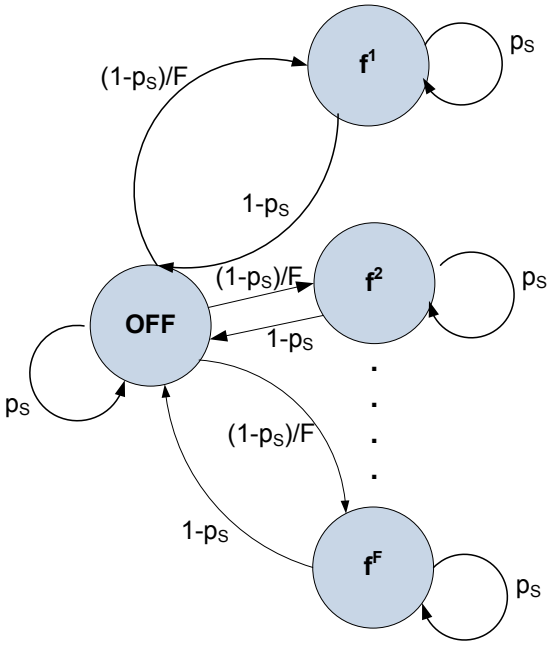


Figure 5.3. PU spectrum occupancy model.

We model the PU spectrum utilization pattern by the finite state model illustrated in Figure 5.3. Each PU is either in the ON or in the OFF state. ON state is comprised of one of the F substates, each of which corresponds to being active using a frequency among a total of F frequencies. The probability of staying in the ON or OFF states

is p_S. The probability of selecting each frequency during switching from OFF to the ON state is equally likely; therefore, the probability of transition from OFF state to any frequency equals $(1 - p_s)/F$. In a slowly varying spectral environment, p_s value is typically high; consequently, we selected p_s as 0.9 in our simulations.

Table 5.1. Parameter names and low/high values for the 2^6 factorial design.

Parameter Name	Low (-) value	High (+) value
N (Number of SUs)	5	30
M (Number of PUs)	5	40
F (Number of frequencies)	3	30
V_p (Velocity of PUs)	1 m/s	25 m/s
V_s (Velocity of SUs)	1 m/s	25 m/s
a (Number of antennas of SUs)	1	5

We initially utilize experimental design methods to evaluate the impacts of six parameters on the throughput maximizing scheduler using analysis of variation (ANOVA) method [87]. We adopt a "2^k factorial" experimental design method, where $k = 6$ since we evaluate six parameters. In other words, we set both *low (-)* and *high (+)* values for the $k = 6$ parameters and run experiments for all the $2^6 = 64$ possible parameter settings. After implementing the ANOVA method, we determine the statistically significant and insignificant terms. We then run detailed experiments with the statistically significant terms. That is to say, we initially implement *factor screening* experiments, and then evaluate the impact of the significant factors. The six parameters we considered together with their low and high values are outlined in Table 5.1. For the velocities of the nodes, we consider the case where all SUs move with the same speed and all PUs move with the same speed. This implies that the speed of the SUs (V_s) and the PUs (V_p) can in general be different from each other; however, the speed of an SU is the same as the speed of the other SUs. Likewise, the speed of a PU is the same as the speed of the other PUs. For the low values of the velocities, we take 1 meter/second as a representative of pedestrian speed, whereas we take 25 meters/second for the high values representing vehicular speed. Additionally, we consider the case where all SUs

have the same number of antennas; i.e., $a_i = a, \forall i \in \mathcal{N}$.

The number of samples that we need to take in the experiments in order to obtain a good estimate of the actual mean depends on the variance of the data. If the variance of the data is little, there is no point in running the experiment with too many samples. Especially considering the fact that we solve integer linear programming problems in CPLEX one after another, it is vital to efficiently use the computational resources. In a data set where the variance is known, the number of samples we need to take to ensure that the sample mean is within $\pm E$ of the actual mean with a $100(1-\alpha)\%$ confidence level is as follows [88]:

$$n = \left\lceil (\frac{z_{\alpha/2}\sigma_{data}}{E})^2 \right\rceil \tag{5.36}$$

Here, $z_{\alpha/2}$ denotes the upper $\frac{\alpha}{2}\%$ percentile of the standard normal distribution, n represents the sample size, and σ_{data} symbolizes the standard deviation of the data. The ceiling operator is necessary because n has to take integer values. Note here that $2E$ denotes the width of the confidence interval (CI). In our experiments, we take $\alpha = 0.05$ and $E = 0.5$; in other words, we can say with 95% confidence level that we are within ± 0.5 of the actual mean in our experiments. Note here that samples in our case correspond to the number of scheduling periods that we run the simulations for.

In our case, however, we do not know the actual standard deviation of our data. Therefore, we make a statistical estimation of the standard deviation (σ_{data}) as we take the samples and use the formula in Equation 5.36 by plugging in our estimated value for σ_{data}. In other words, we employ an iterative method. We firstly take 50 samples because according to the central limit theorem at least 40 samples should be taken in order for the formula in Equation 5.36 to be valid. Afterwards, we calculate the standard deviation of these data with 50 samples and find the value for n using Equation 5.36. Finally, we take the sample mean of these $n+50$ samples and conclude that this estimate is our final estimate for the mean of the data. In order to verify the validity of our method, we calculate the standard deviation of these $n+50$ samples

and find another value for n, which we call n_{new}, by using Equation 5.36. If n_{new} is greater than our actual sample size $n+50$, it implies that there is an undesired feature associated with our data, like the samples not being independent of each other. We do not observe this kind of problem in any of our experiments and hence the validity of our estimation procedure for mean throughput is verified.

Table 5.2. ANOVA results.

Source	Sum Squares	Degrees of Freedom	Mean Square Error	F statistic	P-value
N	10627082.7	1	10627082.7	662798.03	0
M	115397.5	1	115397.5	7197.2	0
F	75508366.3	1	75508366.3	4709363.63	0
V_p	716.6	1	716.6	44.69	0
V_s	5363.4	1	5363.4	334.51	0
a	5654720.5	1	5654720.5	5654720.5	0
NM	6227.3	1	6227.3	388.39	0
NF	10769739.6	1	10769739.6	671695.37	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
MFV_pV_sa	1.4	1	1.4	0.09	0.7693
$NMFV_pV_sa$	4.6	1	4.6	0.29	0.5907
$Error$	296590.8	18498	16	-	-
$Total$	250858751.9	18561	-	-	-

We implement multiway (n-way) ANOVA technique using MATLAB because we have different sample sizes for each of the 64 experiments. We present in Table 5.2 some portion of the resulting ANOVA table. Using a significance level of $\alpha = 0.05$, we conclude that the terms with P-value > 0.05 are statistically insignificant.

Having eliminated some of the statistically insignificant terms using multi-way ANOVA method, we then fit a linear regression model to the rest of the terms, the total number of which is 37. For each term x of these 37 terms, the regression analysis

yields a regression coefficient r_x and a 95% CI $[r_x^l, r_x^h]$, where r_x^l and r_x^h denote the lower and upper bounds, respectively, for the CI of r_x. R^2 statistic for our first regression model is 0.986; i.e., the model explains 98.6% of the variability in the data. The rest of the statistics are F-statistic: 52.6488, P-value: 0, and an estimate of error variance of 182.6994. If the CI for a regression coefficient contains 0; i.e., $r_x^l < 0 < r_x^h$, we conclude that this term is statistically insignificant since there is a good chance that this regression coefficient might be equal to 0. Eliminating these terms, we obtain another regression model with the rest of the terms, the total number of which is 14. The statistics for our second regression model are $R^2 = 0.9675$, F-statistic: 114.4727, P-value: 0, and an error variance estimate of 228.3358. Note that the new model has a significantly reduced number of terms with only a little decrease in the R^2 value. We then apply the same method of eliminating the terms with $r_x^l < 0 < r_x^h$ and fit a third regression model for the rest of the terms, the total number of which is 13. We observe that none of the regression coefficients in this model contains 0 in its CI. Moreover, 13 terms is a sufficiently simple model. The R^2 statistic for this third model equals 0.9649, which is adequate for explaining the variability in the data. Therefore, we conclude that the terms indicated by this final regression model are the statistically most significant ones. We present our final regression model in Table 5.3. The rest of the statistics for this model are F-statistic: 116.8798, P-value: 0, and an error variance estimate of 241.6234. The term β_0 in the model refers to the constant term.

When we analyze this final regression model in Table 5.3, we observe that the most significant 2-way interactions are NM, NF, NV_p, MF, FV_p, and Fa. Therefore, in this chapter, we analyze the impact of NM, NF, MF, and Fa in more detail.

In analyzing each interaction, we run the experiments by setting the values of the parameter pair of the interaction to numerous values between their (-) and (+) values, which are indicated in Table 5.1. This way, we are able to examine in more detail how the average total network throughput is influenced by these parameters. For instance, for the NF interaction case we run the experiments for all possible combinations of the N and F parameters indicated in the value range part of Table 5.4. For the rest of the parameters, i.e., M, V_p, V_s, and a in the NF interaction case, we take their middle

Table 5.3. Final Regression Model.

Term	Coefficient	Lower limit of 95% CI	Upper limit of 95% CI
β_0	139.1349	135.2341	143.0357
N	63.2127	59.3119	67.1135
M	16.669	12.7682	20.5698
F	17.4247	13.5239	21.3255
V_p	6.1122	2.2114	10.013
NM	-14.2436	-18.144	-10.3428
NF	-13.1142	-17.015	-9.2134
NV_p	-4.698	-8.5988	-0.7972
MF	10.9865	7.0857	14.8873
FV_p	9.9355	6.0347	13.8363
Fa	4.2739	0.3731	8.1747
NMF	-4.9622	-8.863	-1.0614
NFV_p	-4.6089	-8.5097	-0.7081

Table 5.4. Parameter Values for Detailed Experiments.

Parameter	Value Range	Middle Value
N	$\{5, 10, 15, \cdots, 30\}$	15
M	$\{5, 10, 15, \cdots, 40\}$	20
F	$\{3, 6, 9, \cdots, 30\}$	15
V_p	$\{1, 4, 7, \cdots, 25\}$	13
V_s	$\{1, 4, 7, \cdots, 25\}$	13
a	$\{1, 2, \cdots, 5\}$	3

value, which is approximately equal to the average of their (-) and (+) values. We outline the values of all the parameters in the detailed experiments in Table 5.4.

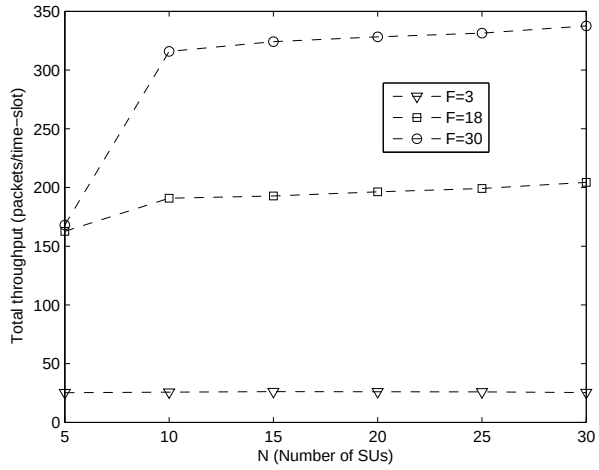


Figure 5.4. Average total throughput for varying number of SUs (N) and frequencies (F).

We present in Figure 5.4 the average total throughput results of the throughput maximizing scheduler for only the low, middle, and high values of F. We can see in this figure that the average total throughput is almost invariant for varying N when F is small. This is because the number of resources in the system (F) is so little that it does not make much difference to have an increasing number of SUs in the system since almost all resources are already occupied by all SUs even when the number of SUs is little. As F increases, increasing the number of SUs increases the average network throughput. This increase continues until some point after which the average network throughput saturates.

We present in Figure 5.5 the results for the NM interaction for the throughput maximizing scheduler, i.e., how the average total network throughput is affected by varying N and M. We can see that the total network throughput decreases as the number of PUs in the CRN cell (M) increases. Increasing the number of PUs implies a decrease in the transmission power of the SUs and consequently their data rate in order not to disturb these PUs. The increase in the total network throughput as the

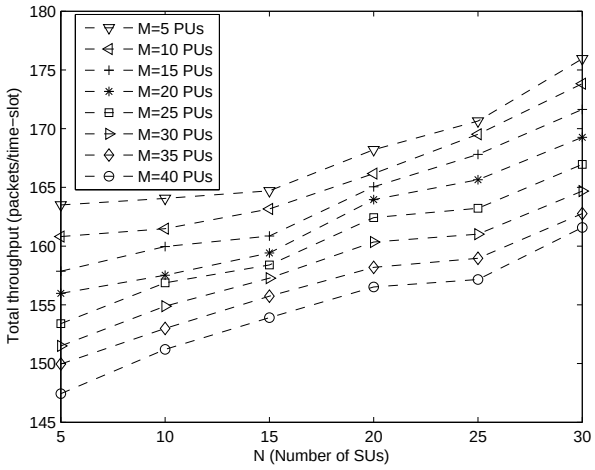


Figure 5.5. Average total network throughput for throughput maximizing scheduler for varying N and M.

number of SUs (N) increases, on the other hand, can be attributed to the opportunistic nature of the throughput maximizing scheduler. A frequency and time slot pair can be regarded as a resource that needs to be assigned to only one SU. Because this resource is most of the time (except in order to comply with the time slot constraint in Equation 5.14 and antenna constraint in Equation 5.16) assigned to the SU that has the best channel conditions (the maximum U_{if} value), the probability that an SU with better channel conditions exists increases as the number of SUs increases.

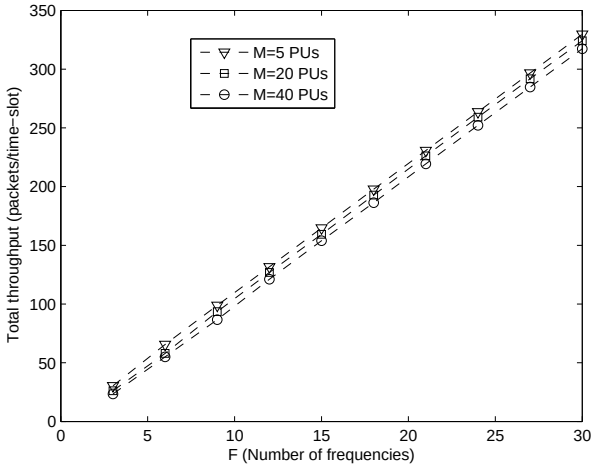


Figure 5.6. Average total network throughput for throughput maximizing scheduler for varying N and F.

Figure 5.6 shows the results for the MF interaction for the throughput maximizing scheduler. Because of the same reasoning as in Figure 5.5, the total average network throughput decreases also here as the number of PUs in the CRN cell increases. Moreover, we can observe that the total network throughput increases almost linearly with the number of frequencies (F) in the CRN cell. This observation is intuitive because the more frequencies there are in the CRN cell, the more resources there are for the SUs to send their data through. Furthermore, we can observe that the number of frequencies is a very important factor on the network throughput. In fact, it is even more important than the number of PUs in the CRN cell since decreasing the number of frequencies in the CRN cell has more impact on decreasing the network throughput in comparison to increasing the number of PUs in the CRN cell.

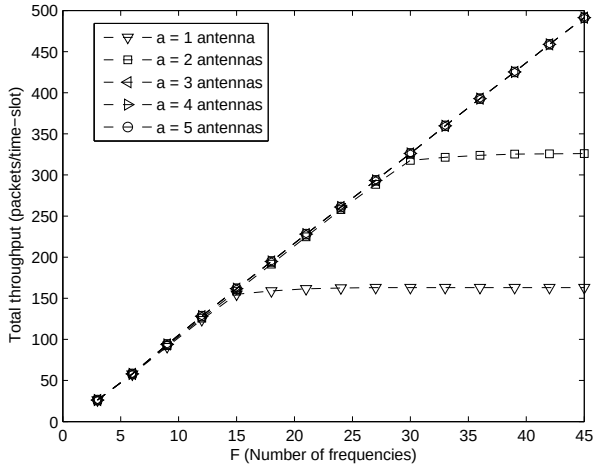
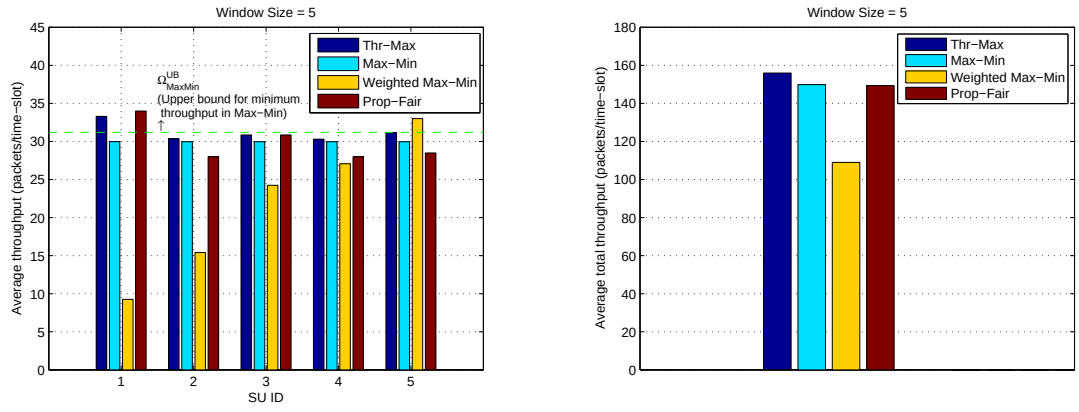


Figure 5.7. Average total network throughput for throughput maximizing scheduler for varying F and a.

We show in Figure 5.7 the results for the Fa interaction for the throughput maximizing scheduler. We see that increasing the number of antennas of the SUs makes sense only when there is a certain number of frequencies in the system. For instance, having one antenna has almost the same effect as having multiple antennas when the number of frequencies is less than 15. On the other hand, when the number of frequencies is between 15 and 30, increasing the number of antennas from 1 to 2 makes a significant difference; nevertheless, having more than two antennas still does not make sense. We see a similar behavior at $F = 30$; i.e., when each SU has 2 antennas,

having $F > 30$ does not increase the total throughput. The reason for this behavior is constraint in Equation 5.14, which ensures that each SU is assigned at least one time slot. In order to comply with this constraint, the scheduler tends to initially assign some frequency to the first antenna of each SU and then continue assigning frequencies to the other antennas. Recall that $N = 15$ in Figure 5.7. Until the point where $N = F$, the scheduler assigns the frequencies to the first antennas of each SU. Increasing the number of antennas has a similar effect on total throughput as increasing the number of SUs. Hence, between $F = N$ and $F = 2N$, the scheduler tends to assign the frequencies to the second antennas of each SU. Bear in mind that this is the average case behavior of the scheduler. Taking this scheduler behavior into consideration, we conclude that a centralized network entity responsible for assigning frequency bands to several CBSs may opt not to assign more frequencies (F) than $a \times N$ to a particular CBS since it does not bring any additional total throughput advantage to this CRN cell. Likewise, the decision about how many antennas the SUs should have can be made by considering the number of frequencies (F) in the CRN cell since having multiple antennas has an additional hardware cost.



(a) Average throughput of all the SUs.

(b) Average total throughput.

Figure 5.8. All schedulers for $N = 5$, window size (φ) $= 5$, and target weights: $\eta_1 = 0.05$, $\eta_2 = 0.1$, $\eta_3 = 0.2$, $\eta_4 = 0.25$, $\eta_5 = 0.4$.

Figure 5.8 displays the behavior of throughput maximizing scheduler (*Thr-Max*), max-min fair scheduler (*Max-Min*), weighted max-min fair scheduler (*Weighted Max-Min*), and proportionally fair scheduler (*Prop-Fair*) for $N = 5$ SUs, window size (φ)

= 5, where the other parameters (M, F, V_p, V_s, and a) are set to their middle values, as shown in Table 5.4. Target weights for the weighted max-min fair scheduler are $\eta_1 = 0.05$, $\eta_2 = 0.1$, $\eta_3 = 0.2$, $\eta_4 = 0.25$, $\eta_5 = 0.4$. We can see here that when compared with the throughput maximizing scheduler, max-min fair scheduler achieves more uniform throughput among the SUs at the expense of a small decrease in the total network throughput, hence achieving fairness. We also present the value for Ω^{UB}_{MaxMin} described and proved in Theorem 5.1, i.e., theoretical upper bound for the minimum throughput in the max-min fair scheduler. We can see that the optimal minimum throughput achieved by the max-min fair scheduler is close to its theoretical upper bound. Throughput values achieved by the weighted max-min fair scheduler, on the other hand, are in line with their target weights at the expense of less total network throughput than the other two schedulers. Besides, proportionally fair scheduler also results in similar throughput values among the SUs; however, the resulting throughput values are not as close to each other as in the max-min fair scheduler. Throughput values of each SU are closer to the throughput values in the throughput maximizing scheduler. In other words, proportionally fair scheduler, as expected, exhibits a tradeoff between users' satisfaction and system revenue.

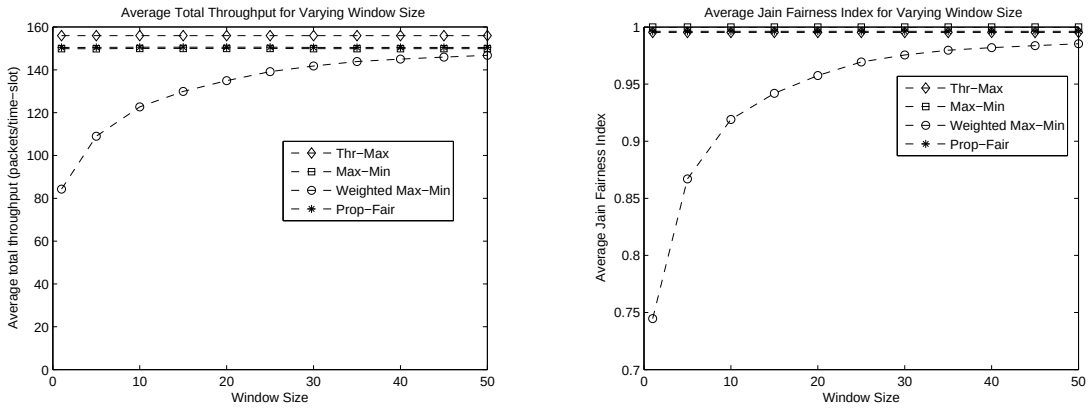
Table 5.5. Achieved and target throughput ratios for weighted max-min fair scheduler.

SU index	$\varphi = 1$	$\varphi = 5$	$\varphi = 10$	$\varphi = 15$	$\varphi = 20$	$\varphi = 25$	$\varphi = 30$	$\varphi = 35$	$\varphi = 40$	$\varphi = 45$	$\varphi = 50$	Target Weight (η)
SU-1	0.057	0.085	0.108	0.121	0.133	0.142	0.148	0.153	0.157	0.16	0.163	0.05
SU-2	0.105	0.141	0.162	0.174	0.179	0.184	0.186	0.188	0.189	0.191	0.192	0.1
SU-3	0.195	0.222	0.221	0.219	0.217	0.215	0.213	0.212	0.211	0.21	0.21	0.2
SU-4	0.249	0.248	0.238	0.231	0.226	0.222	0.22	0.218	0.216	0.215	0.213	0.25
SU-5	0.391	0.302	0.268	0.252	0.242	0.235	0.23	0.227	0.224	0.222	0.22	0.4

We run a similar experiment with window size (φ) = 10 and observe that the total throughput achieved by the weighted max-min fair scheduler with $\varphi = 10$ is more than the one achieved by the weighted max-min fair scheduler with $\varphi = 5$. We also

observe that this increase in total throughput is accomplished at the expense of a small deviation from the target weights. In contrast, performances of max-min fair scheduler and proportionally fair scheduler do not change much with varying window size. Therefore, we have decided to analyze the impact of window size on these schedulers in more detail. We do not explicitly show the results of the experiment with $\varphi = 10$ in a separate figure since we show in the sequel the results of this experiment as part of all the results in Figure 5.9 and Table 5.5.

In Table 5.5, we show the resulting throughput ratios of each SU for varying window size. Apparently, the deviation from target weights increases as the window size increases. When we compare the cases where $\varphi = 1$ and $\varphi = 50$, we can see that the deviation from target weights can become quite large.



(a) Average total throughput.

(b) Average Jain fairness index.

Figure 5.9. All schedulers for varying window size (φ), $N = 5$, and target weights: $\eta_1 = 0.05$, $\eta_2 = 0.1$, $\eta_3 = 0.2$, $\eta_4 = 0.25$, $\eta_5 = 0.4$.

Jain's index is a commonly used fairness index in the literature [89]. It is calculated as $\frac{(\sum_{i=1}^{N} \Omega_i)^2}{N \times \sum_{i=1}^{N} \Omega_i^2}$, where Ω_i is the throughput of SU i and N is the total number of SUs. Jain's index becomes closer to 1 as the throughput values of the SUs become closer to each other. The minimum and maximum values that it can take are $\frac{1}{N}$ and 1, respectively [89]. We present in Figure 5.9a the total average throughput values and in Figure 5.9b the average Jain fairness index values for all schedulers with varying window size (φ). The performance of throughput maximizing scheduler in terms of

both criteria naturally remains constant since window size is not a parameter of this scheduler. The total throughput and Jain index performances of max-min fair and proportionally fair schedulers, on the other hand, are almost invariant of the window size. However, the performance of the weighted max-min fair scheduler is significantly affected by the window size in terms of both performance criteria. The total throughput increases as the window size increases until it saturates at some point and becomes close to the total throughput of max-min fair scheduler. Hence, we observe that our windowing mechanism provides our weighted max-min fair scheduler with the flexibility to provide a tradeoff between maximizing total throughput and adhering to the target throughput proportions.

Throughput and fairness results in Figure 5.9 call for weighted max-min fair schedulers that determine the optimal window size (φ) achieving a certain throughput or fairness objective. First, in line with our observations in Figure 5.9a, we can formulate the following scheduling problem:

$$\max Z' \tag{5.37}$$

$$s.t.$$

$$\sum_{i=1}^{N}\sum_{f=1}^{F}\sum_{t=1}^{T}\frac{U_{if}X_{ift}}{T} \geq \Omega \tag{5.38}$$

$$5.28, \text{ and } 5.29 \tag{5.39}$$

where Ω is the desired minimum total throughput value, which is fed as an input variable to the optimization problem in Equations 5.37-39. Unlike in Equations 5.27-30, the variable φ is a decision variable rather than an input variable.

Second, in line with our observations in Figure 5.9b, we can formulate the follow-

ing scheduling problem:

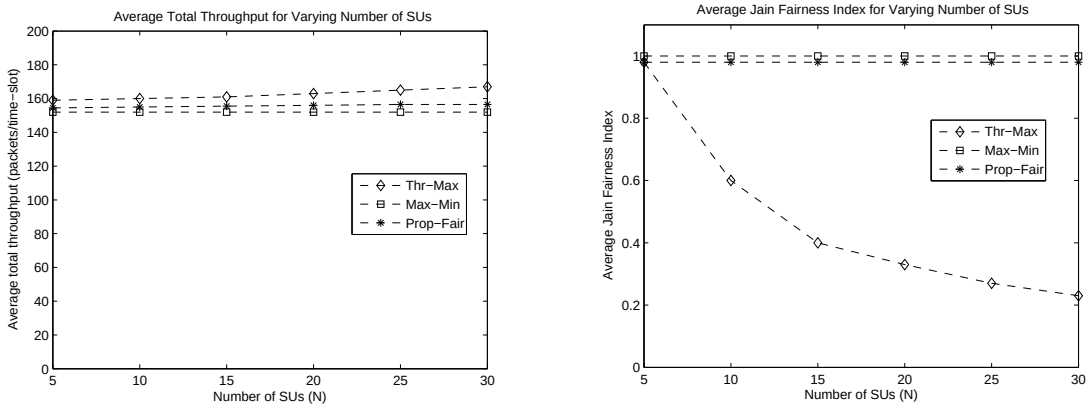
$$\max\ Z' \tag{5.40}$$

$$s.t.$$

$$\frac{(\sum_{i=1}^{N}\sum_{f=1}^{F}\sum_{t=1}^{T}\frac{U_{if}X_{ift}}{T})^2}{N\times\sum_{i=1}^{N}(\sum_{f=1}^{F}\sum_{t=1}^{T}\frac{U_{if}X_{ift}}{T})^2} \geq J \tag{5.41}$$

$$5.28,\ \text{and}\ 5.29 \tag{5.42}$$

where J is the desired Jain's fairness index value, which is fed as an input variable to the optimization problem in Equations 5.40-42. Again unlike in Equations 5.27-30, the variable φ is a decision variable rather than an input variable. Both problems in Equations 5.37-39 and Equations 5.40-42 are nonlinear integer programming problems; therefore, they are computationally difficult. Finding efficient algorithms to address these problems is a research challenge, which is left as future work.



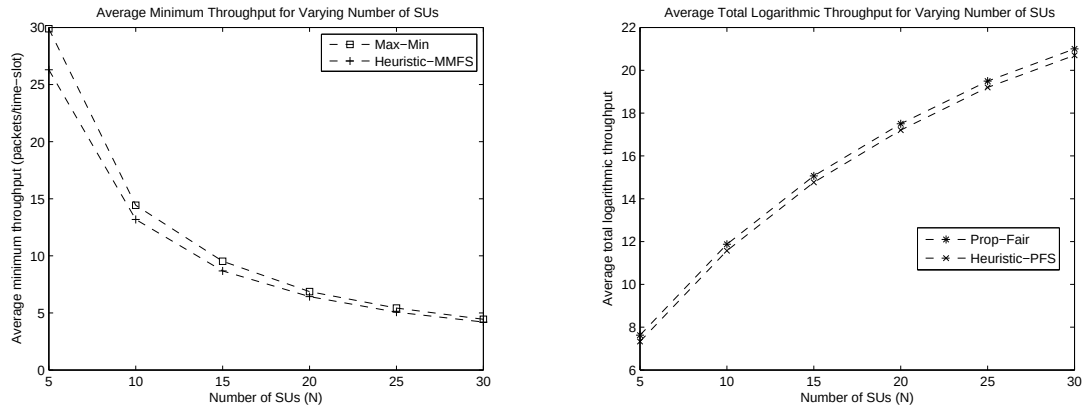
(a) Average total throughput.

(b) Average Jain fairness index.

Figure 5.10. Throughput maximizing, max-min fair, and proportionally fair schedulers for varying number of SUs, $F = 15$, and $\varphi = 5$.

We present in Figure 5.10 the average total throughput and average Jain index values of throughput maximizing, max-min fair, and proportionally fair schedulers for $\varphi = 5$ and varying number of secondary users, where the other parameters (M, F, V_p,

V_s, a) are set to their middle values. We do not present the values for the weighted max-min fair scheduler here because the target weights of each SU have to change as the number of SUs increases and these different simulation scenarios cannot be compared when there are different target weights. Furthermore, target weights become very small as the number of SUs increases and it becomes difficult to truly assess the throughput and fairness performance of the weighted max-min fair scheduler. Recall that the throughput maximizing scheduler assigns the resources (frequencies and time slots) most of the time to the SU that has the best channel conditions and the probability that an SU with better channel conditions exists increases as the number of SUs in the CRN cell increases. This multiuser diversity created by the opportunistic behavior of the throughput maximizing scheduler increases the variation between the SU throughput values and hence decreases Jain's fairness index as the number of SUs increases. Max-min fair scheduler exhibits the highest Jain's fairness index and the lowest total throughput since its objective is to make the throughput values as close to each other as possible. Proportionally fair scheduler, on the other hand, again displays a tradeoff between the overall system revenue (total throughput) and individual throughput values. Its total throughput and Jain's index are between the corresponding values of the throughput maximizing scheduler and the max-min fair scheduler.



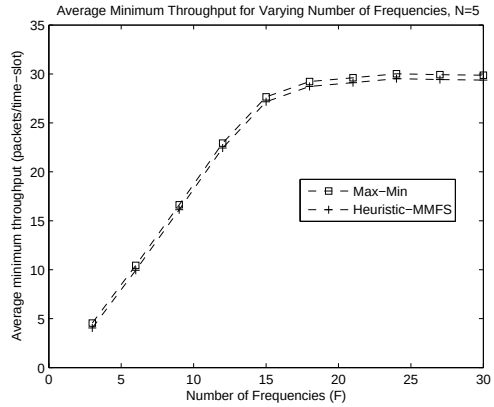
(a) Performance of max-min fair scheduling (b) Performance of proportionally fair scheduling

Figure 5.11. Performance of our heuristic algorithm for varying number of SUs.

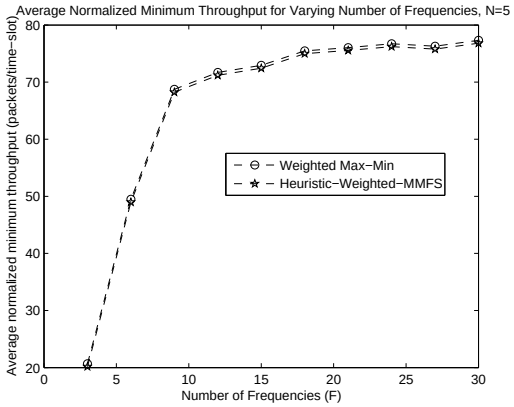
Figure 5.11a shows the average minimum throughput (objective function value)

of CPLEX results (Max-Min) and our heuristic algorithm (Heuristic-MMFS) for $M = 20$, $F = 15$, $V_p = V_s = 13$ m/s, $a_i = 3\ \forall i \in \mathcal{N}$, $\varphi = 5$, and varying number of SUs. Our proposed heuristic yields close performance to the values obtained from CPLEX. Furthermore, we also observe that average minimum throughput decreases as the number of SUs increases since the amount of resources that each SU can receive decreases as there are more SUs competing for the same amount of resources. Figure 5.11b shows the sum of logarithms of SU throughput values (objective function value) of KNITRO results (Prop-Fair) and our heuristic algorithm (Heuristic-PFS) for the same parameters as in Figure 5.11a. Results demonstrate that our proposed heuristic yields very close performance to the values obtained from KNITRO. Objective function value increases as the number of SUs increases; however, they saturate at some point. This behavior is due to the logarithm in the objective function value.

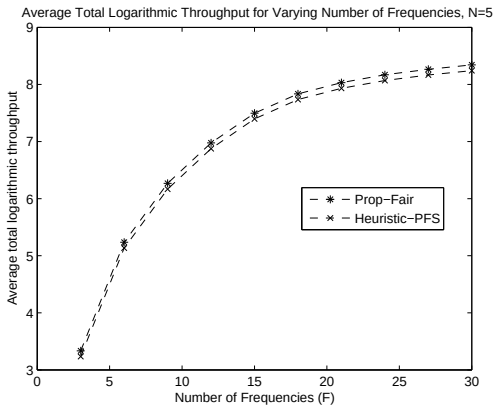
Figure 5.12a shows the average minimum throughput (objective function value) of CPLEX results (Max-Min) and our heuristic algorithm (Heuristic-MMFS) for $N = 5$, $M = 20$, $V_p = V_s = 13$ m/s, $a_i = 3\ \forall i \in \mathcal{N}$, $\varphi = 5$, and varying number of SUs. Our proposed heuristic yields close performance to the values obtained from CPLEX. Furthermore, we also observe that average minimum throughput in general increases as F increases since there are more resources to be shared by the same number of SUs. Figure 5.12b shows the normalized average minimum throughput (objective function value) of CPLEX results (Weighted Max-Min) and our heuristic algorithm (Heuristic-Weighted-MMFS) for the same parameters, where normalized average minimum throughput indicates the minimum throughput obtained after the U_{if} values are updated as $\frac{U_{if}}{\eta_i}$ in Steps 3-5 of Algorithm FAIRSCH in Figure 5.2. Results demonstrate that our heuristic algorithm achieves close results to the ones obtained from CPLEX. Figure 5.12c shows the sum of logarithms of SU throughput values (objective function value) of KNITRO results (Prop-Fair) and our heuristic algorithm (Heuristic-PFS) for the same parameters. Results demonstrate that our proposed heuristic yields very close performance to the values obtained from KNITRO. Objective function value increases as the number of SUs increases; however, they saturate at some point. This behavior is due to the logarithm in the objective function value.



(a) Performance of max-min fair scheduling

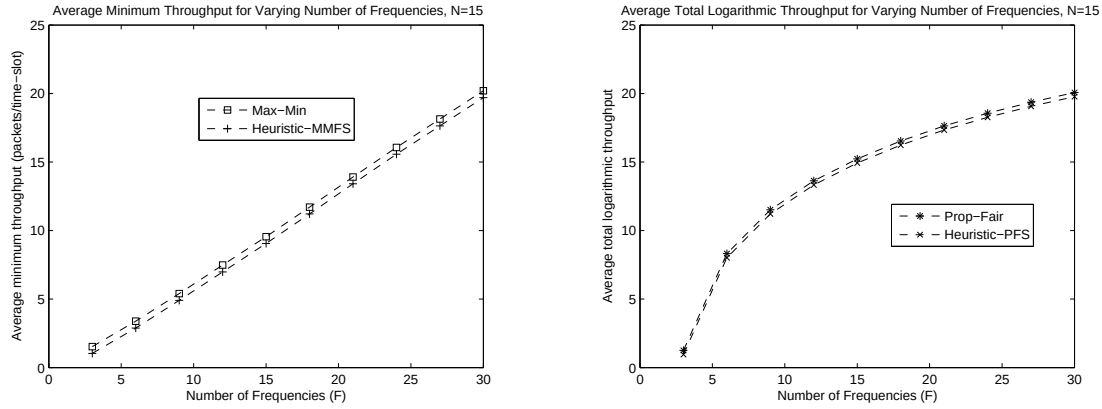


(b) Performance of weighted max-min fair scheduling



(c) Performance of proportionally fair scheduling

Figure 5.12. Performance of our heuristic algorithm for N=5 and varying number of frequencies.



(a) Performance of max-min fair scheduling (b) Performance of proportionally fair scheduling

Figure 5.13. Performance of our heuristic algorithm for N=15 and varying number of frequencies.

Figure 5.13 shows the average minimum throughput (objective function value) of CPLEX results (Max-Min) and our heuristic algorithm (Heuristic-MMFS) for $N = 15$, $M = 20$, $V_p = V_s = 13$ m/s, $a_i = 3\ \forall i \in \mathcal{N}$, $\varphi = 5$, and varying number of frequencies. Our proposed heuristic yields close performance to the values obtained from CPLEX. Figure 5.13b shows the sum of logarithms of SU throughput values (objective function value) of KNITRO results (Prop-Fair) and our heuristic algorithm (Heuristic-PFS) for the same parameters. Results demonstrate that our proposed heuristic yields very close performance to the values obtained from KNITRO.

6. GRAPH THEORETIC APPROACH TO THROUGHPUT MAXIMIZING AND FAIR SCHEDULERS

In Chapter 5, we have formulated throughput maximizing, max-min fair, weighted max-min fair, and proportionally fair scheduling problems, which are referred to as TMS, MMFS, weighted MMFS, and PFS, respectively. In this chapter, we present a graph theoretic approach to these problems. Our contributions can be summarized as follows:

(i) We propose a polynomial time algorithm for the TMS problem
(ii) We investigate the combinatorial properties of certain special cases of the TMS problem and analyze their relations with various combinatorial optimization problems in the literature such as the multiple knapsack and terminal assignment problems
(iii) We investigate the computational hardness of the MMFS problem and some of its special cases in terms of NP-hardness and inapproximability
(iv) We propose an approximation algorithm for the MMFS problem with approximation ratio depending on the maximum possible data rates of the secondary users and evaluate its performance via simulations
(v) We prove that the PFS problem is NP-Hard in the strong sense
(vi) We propose more efficient integer programming formulations for all the three problems

6.1. Proposed Solutions

6.1.1. Preliminaries

Approximation Algorithms: Let Π be a maximization problem and $\rho \geq 1$. A (feasible) solution s of an instance I of Π is a *ρ-approximation* if its objective function value $O_\Pi(s)$ is at least a factor ρ of the optimal objective function value $O^*_\Pi(I)$ of I,

i.e., $O_\Pi(s) \geq \frac{O^*_\Pi(I)}{\rho}$. An algorithm ALG is said to be a *ρ-approximation algorithm* for a maximization problem Π if ALG returns a ρ-approximation for every instance I of Π supplied to it. A problem Π is said to be *ρ-approximable* if there is a polynomial-time ρ-approximation algorithm for it. Π is said to be *ρ-inapproximable* if there is no polynomial-time ρ-approximation algorithm for it unless $P = NP$. An *approximation ratio preserving (polynomial time) reduction* from a maximization problem Π to a maximization problem Π' is a pair of algorithms (f, g) such that (a) f transforms every instance I of Π to an instance $I' = f(I)$ of Π', and (b) g transforms every ρ-approximation s' of $I' = f(I)$ to a ρ-approximation $g(s')$ of I. We denote this fact by $\Pi \preceq_{APX} \Pi'$. Π and Π' are said to be equivalent under approximation preserving reductions if $\Pi \preceq_{APX} \Pi'$ and $\Pi' \preceq_{APX} \Pi$. A polynomial time approximation scheme (PTAS) for a problem Π is an algorithm ALG which takes as input both the instance I and an error bound ϵ, runs in time polynomial in $|I|$ and has approximation ratio $(1 + \epsilon)$. In fact, such an algorithm ALG is a family of algorithms ALG_ϵ that has approximation ratio $(1+\epsilon)$ for any instance I. The running time of a PTAS is required to be polynomial in $|I|$ for every fixed ϵ but can be different for different ϵ.

Matchings: **I**-*matching* and **I**-*factor* are defined as follows [90]: Let $G = (V, E)$ be a (multi)graph with weight function $w : E \to \mathbb{R}$ on its edges, and let $\mathbf{I}$ be a function associating an interval of natural numbers with each vertex in V. We denote by $\delta_G(v)$ the set of incident edges of V in G, i.e. $\delta_G(v) = \{e \in E | v \in e\}$, and $d_G(v) = |\delta_G(v)|$ is the degree of v in G. An **I**-*matching* is a function $\mathbf{m}$: $E \to \mathbb{N}$ such that for $v \in V$, $\sum_{e \in \delta_G(v)} \mathbf{m}(e)$ lies in the interval $\mathbf{I}(v)$. An **I**-*factor* is an **I**-*matching* such that $\mathbf{m}$: $E \to \{0, 1\}$. A *matching* is an **I**-*factor* such that $\mathbf{I}(v) = \{0, 1\}$ for each $v \in V$. In particular if $\mathbf{I}(v) = \{1\}$ for each $v \in V$ it is called a *perfect matching*. A *maximum weighted* **I**-*factor* is an **I**-*factor* $\mathbf{m}$ such that $\sum_{e \in E} \mathbf{m}(e).w(e)$ is maximized. A maximum weighted **I**-*factor* can be found in polynomial time [91,92]. An **I**-*factor* $\mathbf{m}$ corresponds to a sub(multi)graph M of G such that the multiplicity of the edges of G in M is given by the function $\mathbf{m}$. With slight abuse of notation, M will also be called an **I**-*factor*.

Let $G = (U, V, E)$ be an edge weighted bipartite (multi)graph. Let $b(u) = \sum_{e \in \delta_G(u)} \mathbf{m}(e).w(e)$, $\forall u \in U$. A *max-min weighted* **I**-*factor* is an **I**-*factor* $\mathbf{m}$ such that $\min_{u \in U} b(u)$ is maximized. In the rest of this chapter, we use the terms *maximum* **I**-*factor* and *max-min* **I**-*factor* to mean *maximum weighted* **I**-*factor* and *max-min weighted* **I**-*factor*, respectively. Since our focus is on edge weighted graphs, the implication to the *weighted* case is implicit.

The Santa Claus *Problem:* The Santa Claus problem is defined in [93]: Santa Claus has a set of presents that she wants to distribute among a set of kids. Each present has a different value for different kids. The happiness of a kid is the sum of the values of the presents she gets. Santa's goal is to distribute the presents in such a way that the least happy kid is as happy as possible.

In the sequel, we propose equivalent simpler ILP formulations for the TMS, MMFS, and PFS problems.

Lemma 6.1. *Let* Π *be an optimization problem that involves the variables* X_{ift} *only in the form* $\sum_{t=1}^{T} X_{ift}$ *in its objective function* O_π *and also in all its constraints except constraints in Equations 5.15, 5.16 and 5.17. Let* Π' *be the optimization problem obtained from* Π *by substituting*

$$Y_{if} = \sum_{t=1}^{T} X_{ift}, \forall i \in \mathcal{N}, \forall f \in \mathcal{F} \tag{6.1}$$

in O_Π *and all the constraints except Equations 5.15, 5.16, 5.17 and by replacing the constraints in Equations 5.15, 5.16 and 5.17 by the following constraints:*

$$\sum_{i=1}^{N} Y_{if} \leq T; \forall f \in \mathcal{F} \tag{6.2}$$

$$\sum_{f=1}^{F} Y_{if} \leq a_i \cdot T; \forall i \in \mathcal{N} \tag{6.3}$$

$$Y_{if} \in \mathbb{N}; \forall i \in \mathcal{N}, \forall f \in \mathcal{F}. \tag{6.4}$$

Then Π *and* Π' *are equivalent under approximation preserving reductions.*

Proof. It is sufficient to show that any solution X of Π can be converted in polynomial time to a solution Y of Π' with $O_{\Pi'}(Y) = O_{\Pi}(X)$, and vice versa. In particular this holds also for an optimal solution X^*, thus both problems have the same optimum, and a ρ-approximated solution for one problem corresponds to a ρ-approximated solution for the other.

One direction is immediate. Indeed given a solution X of Π, the solution Y defined by Equation 6.1 clearly satisfies all the constraints except Equations 6.2, 6.3 and 6.4 because these constraints were obtained by substituting Equation 6.1 in the original constraints that are satisfied by our assumption. Moreover, $O_{\Pi'}(Y) = O_{\Pi}(X)$ for the same reason. On the other hand, note that the constraints in Equations 6.2, 6.3 and 6.4 are obtained by summing up T inequalities of type Equations 5.15, 5.16 and 5.17, respectively, each of which is satisfied by our assumption.

Now assume that we are given a solution Y of Π'. Any decomposition of Y into $X_{ift}, \forall i \in \mathcal{N}, \forall f \in \mathcal{F}, \forall t \in \mathcal{T}$ satisfying Equation 6.1, clearly satisfies all constraints except Equations 5.15, 5.16 and 5.17 and also $O_{\pi}(X) = O_{\pi'}(Y)$ for the same reason as in the previous paragraph. In the sequel we give a polynomial-time algorithm that finds such a decomposition and also satisfies the constraints in Equations 5.15, 5.16, 5.17. From the vector Y we build a bipartite multi-graph $G = (U, V, E)$ such that $U = \{u_1, \ldots, u_N\}$, $V = \{v_1, \ldots, v_F\}$, and there are Y_{if} parallel edges connecting $u_i \in U$ and $v_f \in V$. The degree of a vertex $v_f \in V$ is at most T by constraint in Equation 6.2 and the degree of a vertex $u_i \in U$ is at most $a_i \cdot T$ by constraint in Equation 6.3. Consider the bipartite graph $G' = (U', V, E')$ obtained from G by replacing each vertex $u_i \in U$ with a_i vertices in U' and dividing the at most $a_i \cdot T$ edges adjacent to u_i to these new vertices such that each vertex receives at most T edges. The degree of each vertex of G' is at most T. Let $G'' = (U'', V'', E'')$ be the graph obtained from G' by adding $||U'| - |V||$ dummy vertices to either U' or V so that $|U''| = |V''|$ and adding dummy edges as long as there are vertices with degree less than T. G'' is a T-regular bipartite graph. A well-known classical result by the works of König, Hall and

Tutte [94–96] implies that such a graph contains a perfect matching and it can be found in polynomial time using the Hungarian algorithm. Removing a perfect matching from G'' one remains with a $(T-1)$-regular bipartite graph. Applying this inductively, G'' is partitioned into T perfect matchings $M''_1, \ldots, M''_T$. By removing all the dummy edges and vertices of G'' from these perfect matchings we obtain T matchings $M'_1, \ldots, M'_T$ of G'. In each matching $M'_t, \forall t \in \mathcal{T}$ we contract back the a_i vertices of U' to the node u_i for every $i \in \mathcal{N}$, to get T bipartite graphs $M_1, \ldots, M_T$ where each vertex $u_i \in U$ has degree at most a_i and each node $v_f \in V$ has degree at most 1. Let $X_{ift} = 1$ if u_i is adjacent to v_f in M_t and 0 otherwise. We conclude that X satisfies Equations 5.15, 5.16 and 5.17. □

Note that the formulation of Π' has less variables and constraints than the formulation of Π; therefore, it is computationally more efficient if we are looking for exact solutions or using optimization software such as CPLEX [5] to find nearly optimal solutions.

Corollary 6.1. *The* TMS *problem is equivalent to find the vector Y given by the following ILP formulation.*

$$\max \sum_{i=1}^{N} \sum_{f=1}^{F} U_{if} Y_{if} \tag{6.5}$$

$$s.t.$$

$$\sum_{f=1}^{F} Y_{if} \geq 1; \forall i \in \mathcal{N} \tag{6.6}$$

$$(6.2), (6.3) \text{ and } (6.4)$$

Corollary 6.2. *The* MMFS *problem is equivalent to find the vector Y given by the*

following ILP formulation.

$$\max\ Z \tag{6.7}$$

$$s.t.$$

$$Z \leq (1-\frac{1}{\varphi})R^i_\varphi + \frac{1}{\varphi}\frac{\sum_{f=1}^{F} U_{if}Y_{if}}{T}; \forall i \in \mathcal{N} \tag{6.8}$$

$$(6.6),(6.2),(6.3)\ \textit{and}\ (6.4)$$

Corollary 6.3. *The* PFS *problem is equivalent to find the vector* Y *given by the following ILP formulation.*

$$\max \prod_{i=1}^{N} \left((1-\frac{1}{\varphi})R^i_\varphi + \frac{1}{\varphi}\frac{\sum_{f=1}^{F} U_{if}Y_{if}}{T} \right) \tag{6.9}$$

$$s.t.$$

$$(6.6),(6.2),(6.3)\ \textit{and}\ (6.4)$$

6.1.2. Algorithms for the TMS Problem

6.1.2.1. A Polynomial-Time Algorithm.

Theorem 6.1. *There exists a polynomial time algorithm for the* TMS *problem.*

Proof. Algorithm THRMAX in Figure 6.1 is an optimal algorithm for the TMS problem. Notice here that the lower bound 1 of $\mathbf{I}(u_i)$ in line 6 is equivalent to constraint in Equation 6.6. Besides, the upper bound T of $\mathbf{I}(v_f)$ in line 7 is equivalent to Equation 6.2, and the upper bound $a_i \cdot T$ of $\mathbf{I}(u_i)$ in line 6 is equivalent to Equation 6.3.

Clearly all the steps except line 8 of the algorithm can be performed in polynomial time. Line 8 calculates a maximum weighted **I**-*factor*. This problem is solvable in polynomial time for general graphs [91]. However, as G is a bipartite graph, its

Require: $\mathcal{N}$, $\mathcal{F}$, a_i, $\forall i \in \mathcal{N}$.

Ensure: Y_{if} values $\forall i \in \mathcal{N}, \forall f \in \mathcal{F}$.

1: Build an edge weighted bipartite (multi)graph $G = (U, V, E)$ as follows:
2: For each $i \in \mathcal{N}$ add a vertex u_i to U.
3: For each $f \in \mathcal{F}$ add a vertex v_f to V.
4: For each pair of vertices $u_i \in U$ and $v_f \in V$, add the edge $\{u_i, v_f\}$ to E with weight U_{if}.
5: Define the following function $\mathbf{I}$, which associates an interval of natural numbers with each vertex in G:
6: $\mathbf{I}(u_i) = [1, a_i \cdot T], \forall i \in \mathcal{N}$
7: $\mathbf{I}(v_f) = [0, T], \forall f \in \mathcal{F}$
8: Find a maximum weighted $\mathbf{I}$-*factor* M of G.
9: For all $i \in \mathcal{N}$ and $f \in \mathcal{F}$, let Y_{if} be equal to the number of edges between vertices u_i and v_f in the $\mathbf{I}$-*factor* M.

Figure 6.1. Algorithm THRMAX.

incidence matrix is totally unimodular. As such, the vertices of the polyhedron corresponding to the linear constraints are integral; in particular, any optimal solution of it is integral. We conclude that the integrality constraints in Equation 6.4 are redundant and thus can be removed. The linear program relaxation obtained in this way can be solved in polynomial time. □

6.1.2.2. Special Cases.

Case 1: Ignore constraint in Equation 6.6 in the TMS formulation. Recall that this constraint ensures that each SU is assigned at least one time slot and therefore achieves temporal fairness. Without this temporal fairness constraint in the problem formulation, some SUs with bad channel conditions may end up with being unable to send any packets for a long time. Some transport layer protocols such as TCP close the connection if no packets are received for a certain amount of time. Constraint in Equation 6.6 gives each SU the opportunity to send at least something and therefore avoids this undesired disconnection situation caused by transport layer protocols. A CBS oper-

ator may prefer to ignore constraint in Equation 6.6 if it does not have any concern with this transport layer behavior or temporal fairness. Ignoring constraint in Equation 6.6 causes the TMS scheduler to behave more opportunistically and increases the possibility to increase the total throughput at the expense of sacrificing from temporal fairness.

In this case, we can solve the following ILP and apply its solution in every time slot $t \in \mathcal{T}$:

$$\max \sum_{i=1}^{N} \sum_{f=1}^{F} U_{if} Y_{if} \tag{6.10}$$

$$s.t.$$

$$\sum_{i=1}^{N} Y_{if} \leq 1; \forall f \in \mathcal{F} \tag{6.11}$$

$$\sum_{f=1}^{F} Y_{if} \leq a_i; \forall i \in \mathcal{N} \tag{6.12}$$

$$Y_{if} \in \{0,1\}; \forall i \in \mathcal{N}, \forall f \in \mathcal{F} \tag{6.13}$$

where Y_{if} is 1 if SU i is assigned to frequency f (in all time slots), and 0 otherwise. Clearly, a method similar to the one in Algorithm THRMAX can be used. The function $\mathbf{I}$ is defined as follows: $\mathbf{I}(u_i) = [0, a_i], \forall i \in \mathcal{N}$, and $\mathbf{I}(v_f) = [0, 1], \forall f \in \mathcal{F}$. However, there are other methods to solve this special case. In the following, we discuss about these alternative ways and the relation of this special case with other combinatorial optimization problems such as the multiple knapsack and terminal assignment problems.

Relation with Other Combinatorial Optimization Problems: This problem is a variant of the knapsack problem, where the frequencies correspond to the items and the SUs correspond to the knapsacks each with capacity a_i. The general case of Equation 6.12 would be $\sum_{f=1}^{F} w_{if} Y_{if} \leq a_i$, where w_{if} values correspond to the weights of the items. Hence, constraint in Equation 6.12 is a special case where $w_{if} = 1, \forall i \in \mathcal{N}, \forall f \in \mathcal{F}$.

In essence, the problem in Equations 6.10-12 is a *multiple knapsack problem (MKP)* where the item profits (U_{if}) vary with knapsacks and all item sizes are identical. The authors of [97] state that the special case of MKP where the item profits vary with bins and all item sizes are identical is solvable in polynomial time. Besides, the work in [98] presents the case with generalized w_{if} as a variant of the *generalized assignment problem (GAP)*, called LEGAP. This problem is also referred to in the operations research literature as the *loading problem*, where the items (frequencies) are loaded into containers (SUs) of different capacities (antennas) such that container capacities are not violated. In the centralized network design literature, this problem is also referred to as the *terminal assignment problem*, where terminals (frequencies) are assigned to the concentrators (SUs). Likewise, the case where all the terminal weights are identical can be solved in polynomial time by alternating chain algorithms [99].

Case 2: Ignore constraints in Equations 6.6 and 6.3 in the TMS formulation. Notice that ignoring constraint in Equation 6.3 is equivalent to having $a_i \geq F, \forall i \in \mathcal{N}$. Then, the optimal solution is to assign each frequency f in every time slot to the SU that has the highest U_{if} value for frequency f, breaking ties arbitrarily. In other words, $X_{ift} = 1$ if $i = \arg\max_i U_{if}$, and 0 otherwise, $\forall f \in \mathcal{F}, \forall t \in \mathcal{T}$.

Case 3: Ignore constraint in Equation 6.6 and assume that $a_i = 1 \; \forall i \in \mathcal{N}$. In this case, a maximum weighted bipartite matching problem between the SUs ($i \in \mathcal{N}$) and the frequencies ($f \in \mathcal{F}$) can be solved and the result of this matching can be applied in every time slot $t \in \mathcal{T}$. Hungarian algorithm [100] can be used to solve the maximum weighted bipartite matching problem. This technique can be extended also to the case where the a_i values are small, by replacing each node u_i corresponding to SU i with a_i nodes each of which corresponds to the antennas of SU i. In this case, the reason for a_i values to be necessarily small is to obtain a polynomial reduction.

6.1.3. Algorithms for the MMFS Problem and its Complexity

We present in this section a graph theoretic approach to the MMFS problem. Recall that we have also formulated the weighted MMFS problem in Chapter 5 in ad-

dition to the MMFS problem. Weighted MMFS is essentially the same computational problem as the MMFS problem, where U_{if} values are replaced by $\frac{U_{if}}{\eta_i}$ and R^i_φ values are replaced by $\frac{R^i_\varphi}{\eta_i}$. All the variables except Z' and Y_{if} are again input variables. The only difference is that U_{if} values in MMFS are integers, whereas the $\frac{U_{if}}{\eta_i}$ values in the weighted MMFS are not necessarily integers. Since they appear only in the objective function, they do not affect the computational hardness of the problem. In the rest of this section, we mainly refer to the unweighted MMFS problem; however the entire analysis is valid for the weighted MMFS problem as well.

6.1.3.1. Hardness Results.

Lemma 6.2. SANTA CLAUS $\preceq_{APX}$ MMFS.

Proof. Consider a special case of the MMFS problem where $T = 1$, $\varphi = 1$, and $a_i \geq F$, $\forall i \in \mathcal{N}$. The optimization in this case corresponds to distributing the frequencies to the SUs in such a way that the SU having the least throughput receives as much throughput as possible. Because $a_i \geq F$ $\forall i \in \mathcal{N}$, there is practically no upper bound on the number of frequencies that an SU can receive. This special case is exactly the SANTA CLAUS problem where kids are replaced by SUs and presents are replaced by frequencies. □

*The Restricted Santa Claus Problem (*R-SANTA CLAUS*):* Due to the difficulty of the SANTA CLAUS problem, the attention in the theoretical computer science community has shifted towards more special cases. One special case is the so-called *restricted Santa Claus* problem (R-SANTA CLAUS), where every present f has the same value U_f for every kid interested in that present, i.e. $U_{if} \in \{U_f, 0\}$. The authors in [101] have shown that it is NP-Hard to approximate the R-SANTA CLAUS problem within any constant factor better than 2. The result of [101] is actually implied by the work of Lenstra *et. al.* [102], which proves that the problem of minimum makespan scheduling in unrelated parallel machines cannot be approximated within any constant factor better than 3/2 unless $P = NP$.

The Degree Two and Symmetric Degree Two Santa Claus *Problems (*D2-Santa Claus *,* SD2-Santa Claus*):*
D2-Santa Claus is the variant of Santa Claus problem when each present has a nonzero value for at most two kids. SD2-Santa Claus is a special case of D2-Santa Claus where every present has the same value for both kids interested in that present. In other words, SD2-Santa Claus is a special case of the R-Santa Claus problem where each present has a nonzero value for at most two kids. Clearly,

Observation 6.1. SD2-Santa Claus $\preceq_{APX}$ D2-Santa Claus $\preceq_{APX}$ Santa Claus.

Observation 6.2. SD2-Santa Claus $\preceq_{APX}$ R-Santa Claus $\preceq_{APX}$ Santa Claus.

Definition 6.1. *For a graph $G = (V, E)$ with non-negative edge weights U, $\alpha(G)$ is the ratio of the maximum edge weight to the minimum non-zero edge weight, i.e.* $\alpha(G) = \dfrac{\max_{e \in E}\{U_e\}}{\min\{U_e | e \in E, U_e > 0\}}$.

Lemma 6.3. *For any $\epsilon > 0$, the* MMFS *problem is $(\alpha(G) - \epsilon)$-inapproximable, even when $T = 1$, $\varphi = 1$, $a_i \geq F$, $U_{if} \in \{U_f, 0\}$ and each frequency f is usable by at most two SUs.*

Proof. This case corresponds to the SD2-Santa Claus problem. The authors in [103] show that SD2-Santa Claus is $(2 - \epsilon)$-inapproximable for any $\epsilon > 0$. The graphs used in their reduction satisfy $\alpha(G) = 2$. Therefore, the lemma holds. □

Note that having $\varphi = 1$ as in Lemma 6.3 implies that the scheduler does not give any importance to what has happened in the recent past and focuses only in the current scheduling period. In other words, $\varphi = 1$ implies that the scheduler does not consider the historical throughput information in its current scheduling decision.

Lemma 6.4. *The* MMFS *problem remains* NP*-Hard in the strong sense even when $T = 1$, $\varphi = 1$, $a_i = 3$, and $U_{if} = U_{jf}$ $\forall i \neq j$ and $i, j \in \mathcal{N}$.*

Proof. Recall that the 3-Partition problem is the problem of deciding whether a given set of integers can be partitioned into triplets all of which have the same sum. More precisely, given a multiset S of $3m$ positive integers $n_1, n_2, \ldots, n_{3m}$ such that

$\sum_{i=1}^{3m} n_i = m \cdot B$, can S be partitioned into m subsets $S_1, S_2, \ldots, S_m$ such that the sum of the integers in each subset is B? This problem is well-known to be NP-Hard in the strong sense. Given such an instance of the 3-PARTITION problem, we can build a complete bipartite graph $G = (U, V, E)$, where $U = \{S_1, S_2, \ldots, S_m\}$, $V = \{b_1, b_2, \ldots, b_{3m}\}$, and each edge $\{S_i, b_j\}$ has a weight equal to n_j, and $a_i = 3, \forall i \in \mathcal{N}$. Every 3-partition corresponds to a feasible solution of this new instance and vice versa. The value of the minimum SU i is equal to B if and only if the answer to the 3-PARTITION problem is *YES*. □

Lemma 6.5. *The* MMFS *problem remains* NP*-Hard even when* $N = 2$*,* $T = 1$*,* $\varphi = 1$*,* $U_{if} = U_{jf}$*, and* $a_i \geq F$*,* $\forall i \neq j$ *and* $i, j \in \mathcal{N}$*.*

Proof. Note that this special case corresponds to the PARTITION problem, i.e. the problem of deciding whether a given set of integers can be partitioned into two subsets that have the same sum. Since the PARTITION problem is NP-Hard in the weak sense, this special case is also NP-Hard in the weak sense. □

6.1.3.2. Algorithms.

Lemma 6.6. MMFS $\preceq_{APX}$ *max-min* ***I****-factor.*

Proof. Since φ and T are both constants, by multiplying both sides of Equation 6.8 by $\varphi \cdot T$ and substituting $K_i = T(\varphi - 1)R^i_\varphi$ and $Z' = Z\varphi T$ we get the following constraint:

$$Z' \leq K_i + \sum_{f=1}^{F} U_{if} X_{if}; \forall i \in \mathcal{N} \tag{6.14}$$

Build a bipartite (multi)graph $G = (U, V, E)$ as follows:

(i) For each SU $i \in \mathcal{N}$, add a vertex u_i to U.
(ii) For each frequency $f \in \mathcal{F}$, add a vertex v_f to V.
(iii) Add a dummy vertex $\overline{v}$ to V.
(iv) For each pair of vertices $u_i \in U$ and $v_f \in V$, add Y_{if} edges $\{u_i, v_f\}$ to E each with weight U_{if}.

(v) For each vertex $v_i \in U$, add the edge $\{v_i, \overline{v}\}$ to E with weight K_i.

We claim that the MMFS problem is equivalent to the max-min **I**-*factor* problem on the (multi)graph G, where the function **I** is as follows: $\mathbf{I}(u_i) = [2, a_i \cdot T + 1]\ \forall v_i \in X$, $\mathbf{I}(v_f) = [0, T]\ \forall v_f \in Y$, and $\mathbf{I}(\overline{v}) = [N, N]$. For an **I**-*factor* H of G, let $Y_{if} = 1$ if and only if the edge $\{u_i, v_f\}$ is in H. $\mathbf{I}(\overline{v})$ implies that $d_H(\overline{v}) = N$. On the other hand $d_G(\overline{v}) = N$ by the construction. Therefore all the incident edges $\delta_G(\overline{v})$ of $\overline{v}$ are also in H. Each node $u_i \in U$ has exactly one incident edge from $\delta_G(\overline{v})$ of utility K_i, and this edge is in H. Therefore $b(u_i) = K_i + \sum_{f=1}^{F} U_{if} \cdot Y_{if}, \forall i \in \mathcal{N}$. In particular this equality holds for the minimum value, thus $\min_i b(u_i) = \min_i (K_i + \sum_{f=1}^{F} U_{if} \cdot Y_{if})$. We conclude that the value of the objective function is equal to the value of the **I**-*factor*. It remains to show that an **I**-*factor* H corresponds to a feasible solution Y and vice versa. Recall that a node $u_i \in U$ has one incident edge from $\delta_G(\overline{v})$, thus $d_H(u_i) - 1$ incident edges connecting it to nodes $v_f, \forall f \in \mathcal{F}$. $\mathbf{I}(u_i)$ implies that $2 \leq d_H(u_i) \leq a_i \cdot T + 1$, thus $1 \leq d_H(u_i) - 1 \leq a_i \cdot T$. We conclude that constraints in Equations 6.6 and 6.2 are satisfied by Y. On the other hand, $\mathbf{I}(v_f)$ implies that constraint in Equation 6.3 is satisfied. The opposite direction is shown similarly. □

Theorem 6.2. *There is an $\alpha(G)$-approximation algorithm for the max-min **I**-factor problem.*

Proof. We show in the following that Algorithm MAXMINEQ in Figure 6.2 is an $\alpha(G)$-approximation algorithm for the max-min **I**-*factor* problem. Consider an optimal max-min **I**-*matching* H^* and the vertex $u_i \in U$ with minimum degree in H^*. Then $d_{H^*}(u_i) \leq D$, because otherwise all the nodes $u_i \in U$ have $d_{H^*}(u_i) \geq D + 1$. Then H^* constitutes an **I**-*factor* for $\mathbf{I}(u_i) = [D + 1, a_i \cdot T], \forall u_i \in U$ and $\mathbf{I}(v_f) = [0, T], \forall v_f \in V$ contradicting the fact that D is the maximum possible value leading to a feasible solution for the steps (7)-(9). Therefore $b(u_i) = \sum_{e \in \delta_{H^*}(u_i)} U_e \leq d_{H^*}(u_i) \cdot \max_{e \in E}\{U_e\} \leq D \cdot \max_{e \in E}\{U_e\}$. Then

$$O(H^*) = \min_{i \in \mathcal{N}} b(u_i) \leq D \cdot \max_{e \in E}\{U_e\} \tag{6.15}$$

On the other hand, our algorithm MAXMINEQ returns an **I**-*factor* H such that

Require: $\mathcal{N}$, $\mathcal{F}$, a_i, $\forall i \in \mathcal{N}$.

Ensure: Y_{if} values $\forall i \in \mathcal{N}, \forall f \in \mathcal{F}$.

1: Build an edge weighted bipartite (multi)graph $G = (U, V, E)$ as follows:
2: For each SU $i \in \mathcal{N}$, add a vertex u_i to U.
3: For each frequency $f \in \mathcal{F}$, add a vertex v_f to V.
4: For each pair of vertices $u_i \in U$ and $v_f \in V$, add the edge $\{u_i, v_f\}$ to E with weight U_{if}.
5: $D \leftarrow \min_i\{a_i\} \cdot T$.
6: Execute lines (7)-(9) iteratively by employing a binary search on D to find the maximum possible value of D for which the below steps (7)-(9) return a feasible solution:
7: Find an **I**-*factor* for:
8: $\mathbf{I}(u_i) = [D, a_i \cdot T], \forall u_i \in U$.
9: $\mathbf{I}(v_f) = [0, T], \forall v_f \in V$.

Figure 6.2. Algorithm MAXMINEQ.

$d_H(u_i) \geq D, \forall i \in \mathcal{N}$. Therefore for every $u_i \in U$, $b(u_i) = \sum_{e \in \delta_H(u_i)} U_e \geq \min\{U_e | U_e > 0, e \in E\} \cdot d_H(u_i) \geq \min\{U_e | U_e > 0, e \in E\} \cdot D$. We conclude that our solution H has value at least $O(H) = \min_{i \in \mathcal{N}} b(u_i) \geq \min\{U_e | U_e > 0, e \in E\} \cdot D$. Combining with inequality in Equation 6.15 we conclude that $\frac{O(H)}{O(H^*)} \geq \alpha(G)$. □

Corollary 6.4. *If $\alpha(G) = 1$, i.e., all nonzero U_{if} values are equal to each other, then max-min **I**-matching is solvable in polynomial time.*

Therefore we have shown the following:

Theorem 6.3. *$\alpha(G)$ is a tight bound for the approximability of all the problems mentioned in this section, i.e. the max-min **I**-factor,* MMFS, Santa Claus, R-Santa Claus , D2-Santa Claus *and* SD2-Santa Claus *problems.*

We proceed with a few observations about special cases.

Lemma 6.7. *There is a* 4*-approximation algorithm for the special case of the* MMFS

problem with $T = 1$, $K_i = 0$, $a_i \geq F$ $\forall i \in \mathcal{N}$ *and each frequency* $f \in \mathcal{F}$ *has nonzero* U_{if} *value for at most two SUs* $i \in \mathcal{N}$.

Proof. This special case corresponds to D2-Santa Claus, for which there exists a 4-approximation algorithm [103]. Therefore, the lemma holds. □

Lemma 6.8. *There exists a polynomial time approximation scheme (PTAS) for the special case of the* MMFS *problem with* $T = 1$, $K_i = 0$, $U_{if} = U_{jf}$, *and* $a_i \geq F$, $\forall i \neq j$ *and* $i, j \in \mathcal{N}$.

Proof. The frequencies and SUs can be regarded as jobs and machines, respectively, and this special case of the problem becomes the problem of maximizing the minimum machine completion time on identical machines [104], which is the dual problem to the well-known problem of makespan minimization. There exists a PTAS for the problem of maximizing the minimum machine completion time [104] and hence the lemma holds. □

6.1.3.3. Related work on the Santa Claus Problem. Bansal and Sviridenko have shown in [93] that the natural LP formulation of the Santa Claus problem has a big integrality gap. Therefore, instead of the natural LP formulation, Bansal and Sviridenko have considered the so-called *configuration LP* and showed how to round it so that the resulting value is at least OPT/N, where N is the number of kids (SUs). They also showed that the integrality gap is in the order of $1/\sqrt{N}$. Asadpour and Saberi have shown in [105] that it is possible to round the configuration LP such that the objective function value is at least $OPT/\sqrt{N}\log^3 N$.

The work of Bansal and Sviridenko [93] proposes a method of rounding the configuration LP for the R-Santa Claus problem such that a factor of at most $\log\log N/\log\log\log N$ is lost. Afterwards, Asadpour, Feige and Saberi [106] have shown an integrality gap of 5 for the R-Santa Claus problem, which was later improved to 4 by the same authors [107]. Note here that this is an *estimation ratio* rather than an approximation ratio. In other words, they have proved that the gap can be at most 4; however, they failed to provide a polynomial time 4-approximation algo-

rithm. They provided a local search heuristic that returns a solution of value at least $OPT/4$; nevertheless, their method is not known to run in polynomial time. Recently, Haeupler, Saha and Srinivasan demonstrated in [108] the first constant factor approximation algorithm for the R-Santa Claus problem and hence solved an important question. Their result returns a c-approximation algorithm for some constant c; however, an explicit value for c is not provided. Therefore, there is still a gap between the 2-inapproximability result of [101] and this constant c-approximability result of [108].

Recall that the $\alpha(G)$-approximation algorithm that we propose in this chapter works for the max-min **I**-*factor* problem, which is a much more generalized version of the general (not restricted) Santa Claus problem. Therefore, our algorithm also works for the general Santa Claus problem, for which very few approximation results have been found [93], [105]. The additive approximation ratio of $\max_{if} U_{if}$ in [101] can be arbitrarily bad since it gives a very bad guarantee even when a single U_{if} value is large and the others are small. The $\sqrt{N}(log^3 N)$ approximation ratio of [105] can still be bad when the number of kids (SUs) is large. On the other hand, our $\alpha(G)$-approximation algorithm gives a good approximation guarantee when the U_{if} values are close to each other; i.e., it works well in a fairly uniform spectral environment where the availabilities of all frequencies are similar for every SU. Even if the number of SUs is very large, our algorithm gives a good approximation ratio as long as the nonzero U_{if} values are close to each other; i.e., $\alpha(G)$ value is small. In other words, it gives a better result than the one in [105] when there are many SUs but a uniform spectral environment. However, it fails to provide a good approximation guarantee for highly non-uniform spectral environments. On the other hand, when the number of kids (SUs) is small and the values of the presents to the kids (spectral environment) are highly non-uniform, the algorithm in [105] gives a better approximation ratio. To sum up, our algorithm is strong in terms of two different criteria: Firstly, unlike [93] and [105], our algorithm works for a much more general case than the Santa Claus problem. Secondly, it gives a better approximation guarantee than the previous work when there are a large number of SUs and the spectral environment is fairly uniform. In other words, it provides an alternative solution so that whichever algorithm provides a better approximation guarantee (either ours or the ones in [93] and [105]) can be chosen in a

practical implementation.

Authors in [101] have shown a method that gives an approximation guarantee of $OPT - \max_{if} U_{if}$ for the general SANTA CLAUS problem. Although their method performs badly for high values of U_{if}, its performance guarantee is good for low values of U_{if}. Therefore, the approximation algorithm in [93] gives good results for the special case of the MMFS problem where U_{if} values are small, $T = 1$, $K_i = 0$, and $a_i \geq F$ $\forall i \in \mathcal{N}$.

6.1.3.4. Special Cases.

Case 1: Assume that $a_i = 1$ $\forall i \in \mathcal{N}$ and $T = 1$. In this case, the problem is equivalent to the max-min version of the *linear bottleneck assignment problem (LBAP)*, where the workers (frequencies) are assigned to the workstations (SUs) such that the completion time of the job with the latest completion time is minimized. The authors in [109] develop threshold algorithms, a dual method, and a shortest augmenting path method for solving LBAP in polynomial time. The work in [100] also develops thresholding methods. In particular, min-max version of LBAP is equivalent to its max-min version [100, 109]. Note that having $K_i = 0$ $\forall i \in \mathcal{N}$ is not necessary here. Even when $\exists i \in \mathcal{N}$ such that $K_i \neq 0$, thresholding method in [100] can still be applied. However, as in the proof of Lemma 6.6, we need a dummy vertex called $\overline{v}$ and dummy edges between each SU i and $\overline{v}$ with weight K_i, and then we need to implement the thresholding method in this new graph.

Case 2: Recall that if we decide to neglect the past performance, then $K_i = 0, \forall i \in \mathcal{N}$. In this case constraint in Equation 6.6 can be eliminated. The reason for this is that the objective function already aims to assign every SU at least one time slot because it tries to make the throughput of every SU as high as possible. If it is possible to assign each SU at least one time slot, objective function will do it anyway. If it is not possible, then the only difference is that the case with constraint in Equation 6.6 declares that no feasible solution can be found, whereas the case without

constraint in Equation 6.6 declares that a feasible solution has been found but the resulting objective function value is zero. Therefore, both cases are essentially the same and hence constraint in Equation 6.6 can safely be eliminated in this special case. This elimination reduces the number of constraints in the problem formulation hence enables more efficient running time if optimization software such as CPLEX [5] is used to find close to optimal solutions.

Remark 6.1. Assume that $a_i = 1, \forall i \in \mathcal{N}$ and ignore constraint in Equation 6.6. In this special case, the method of solving the problem for one time slot and applying the same solution for all time slots (akin to the method used in special case 3 of the TMS problem in Section 6.1.2.2) does not work. To see this, consider the following example: There are 2 SUs and 2 frequencies. Let $U_{11} = U_{21} = 3$ and $U_{21} = U_{22} = 0$. If we solve the problem in one time slot and apply the same solution to the other time slots, then the result equals zero. However, we can achieve a nonzero result by assigning different frequencies to an SU in different time slots. Assigning frequency 1 to SU 1 and frequency 2 to SU 2 in the first time slot, and assigning frequency 1 to SU 2 and frequency 2 to SU 1 in the second time slot achieves a nonzero throughput value for the minimum throughput.

6.1.4. Results about the PFS Problem

Theorem 6.4. *The* PFS *problem remains* NP*-Hard in the strong sense even when* $T = 1$*,* $\varphi = 1$*,* $a_i = 3$*, and* $U_{if} = U_{jf}$ $\forall i, j \in \mathcal{N}$*.*

Proof. Recall that the 3-Partition problem is to decide whether a given set of integers can be partitioned into triplets that all have the same sum. Assume that there is a polynomial time algorithm for the PFS problem. Then we can use this algorithm to solve the 3-Partition problem as follows: Consider the special case of the PFS problem where there are m SUs, $3m$ frequencies, $T = 1$ time slot, $U_{if} = U_{jf}$ $\forall i, j \in \mathcal{N}$ and $a_i = 3$ $\forall i \in \mathcal{N}$. Every 3-partition corresponds to a feasible solution of this special case and vice versa. Then, S_i equals the sum of all the U_{if} values assigned to this SU i. The values of each S_i are equal to each other if and only if the answer to the 3-Partition problem is *YES*. Notice that maximizing the sum of

logarithms of a set of numbers is equivalent to maximizing their product. Furthermore, this product is maximized when all numbers are equal to each other. In other words, if it is theoretically possible to make the sum of integers in each group equal to each other (if the answer to the 3-PARTITION problem is *YES*), then the polynomial time algorithm for the PFS problem will yield a solution where the sum of integers in each group (the sum of U_{if} values assigned to each SU i) is equal to each other. If the answer is *NO*, then as a result of the PFS execution, the sum of integers in each group will not all be equal to each other; i.e., the sum in at least one group will differ from another sum (in another group). Since the 3-PARTITION problem is NP-Hard in the strong sense, PFS problem is also NP-Hard in the strong sense even when $T = 1$, $\varphi = 1$, $a_i = 3$, and $U_{if} = U_{jf}$ $\forall i, j \in \mathcal{N}$. □

*The Max-Log Santa Claus Problem (*MAX-LOG-SANTA CLAUS*):* This problem is the same as the SANTA CLAUS problem except that the goal of Santa Claus is to maximize the sum of logarithms of the happiness of each kid.

Lemma 6.9. MAX-LOG-SANTA CLAUS $\preceq_{APX}$ PFS.

Proof. Consider a special case of the PFS problem where $T = 1$, $\varphi = 1$, and $a_i \geq F$, $\forall i \in \mathcal{N}$. The optimization in this case corresponds to distributing the frequencies to the SUs in such a way that the sum of logarithms of the throughput values of each SU is as high as possible. Because $a_i \geq F$ $\forall i \in \mathcal{N}$, there is practically no upper bound on the number of frequencies that an SU can receive. This special case is exactly the MAX-LOG-SANTA CLAUS problem where kids are replaced by SUs and presents are replaced by frequencies. □

Theorem 6.5. *The* MAX-LOG-SANTA CLAUS *problem cannot have an approximation algorithm with absolute approximation ratio better than* $\log(\frac{4}{3})$.

Proof. This result is implied by the work of Lenstra *et. al.* [102], which proves that the problem of minimum makespan scheduling in unrelated parallel machines cannot be approximated within any constant factor better than 3/2 unless $P = NP$. In particular, the work in [102] proves that for the minimum makespan problem on unrelated parallel

machines, the question of deciding if there exists a schedule with makespan at most 2 is NP-complete. Let us consider the version of the SANTA CLAUS problem where the goal is to maximize the product of the happiness values of the kids. The work in [102] implies that this version of SANTA CLAUS cannot be approximated within any constant factor better than 4/3 because if an assignment where each kid has a happiness value of 2 cannot exist, then at least one kid has to have a happiness value of 3 and another kid with happiness 1. This result implies that MAX-LOG-SANTA CLAUS cannot have an approximation algorithm with absolute approximation ratio better than $\log(\frac{4}{3})$. □

Corollary 6.5. *Due to Lemma 6.9 and Theorem 6.5,* PFS *problem cannot have an approximation algorithm with absolute approximation ratio better than* $\log(\frac{4}{3})$ *even when* $T = 1$, $\varphi = 1$, *and* $a_i \geq F$, $\forall i \in \mathcal{N}$.

6.2. Practical Implications

Our findings in this chapter indicate that the MMFS problem has high computational complexity even in very special cases. Therefore, this chapter shows that providing throughput fairness to the SUs is a computationally challenging task for CBS operators. To better observe the conceptual computational difficulty of providing throughput fairness, consider the following scheduling problem:

$$\max \sum_{i=1}^{N} \sum_{f=1}^{F} U_{if} Y_{if} \tag{6.16}$$

$$s.t.$$

$$\sum_{f=1}^{F} U_{if} Y_{if} \geq \Omega, \forall i \in \mathcal{N} \tag{6.17}$$

$$(6.6), (6.2), (6.3) \text{ and } (6.4) \tag{6.18}$$

where Ω is a prespecified throughput value. In this problem, the goal is to maximize the total throughput such that each SU is guaranteed a prespecified throughput value of Ω. If we had a polynomial time solution to this problem, we would be able to use this solution iteratively by updating the value of Ω in each iteration and therefore find

the maximum value of Ω for which the above problem has a feasible solution; in other words, we would be able to solve the MMFS problem in polynomial time. Hence, the problem in Equations 6.16-18 is also NP-Hard in the strong sense. That is to say, even checking whether each SU can be guaranteed a certain throughput value is a computationally hard problem.

In contrast, the TMS problem, which maximizes total throughput while at the same time providing temporal fairness, is solvable in polynomial time. Taking into account the fact that scheduling decisions have to be made in real time, a CBS operator may opt to provide temporal fairness (by executing the TMS formulation) instead of throughput fairness. If the CBS operator prefers to use the MMFS formulation in spite of its computational difficulties, it may check whether the special cases we have outlined in this chapter are valid in that particular scheduling period and use the relatively efficient methods we discussed. For instance, we have shown in this chapter that the special case of Case-1 in Section 6.1.3.4 is solvable in polynomial time. Otherwise, the CBS operator can check the $\alpha(G)$ value in that particular scheduling period. If $\alpha(G)$ is low enough; i.e., if the spectral conditions are fairly uniform, the CBS operator can use our $\alpha(G)$-approximation algorithm, with a performance guarantee of $\frac{OPT}{\alpha(G)}$ even when there are a large number of SUs and frequencies.

6.3. Numerical Evaluation

In Section 6.1.3 we have analytically investigated the worst case behavior of our algorithm (MAXMINEQ) for the MMFS problem. In this section, we evaluate the average case behavior of our $\alpha(G)$-approximation algorithm (MAXMINEQ) through simulations. We have used the same simulation conditions as in Chapter 5. The simulated centralized CRN cell has 600 meters of radius. Moreover, $\zeta = 10^{-6}$, $P_{IF_{max}} =$ 10 milliwatts for each frequency, $T = 10$ time slots, and $T_s = 100$ ms. Furthermore, AWGN channels and the same random waypoint mobility model as in Chapter 5 are used.

We take $M = 20$, $F = 15$, $V_p = V_s = 13$ m/s, and $a_i = 3\ \forall i \in \mathcal{N}$. We compare the

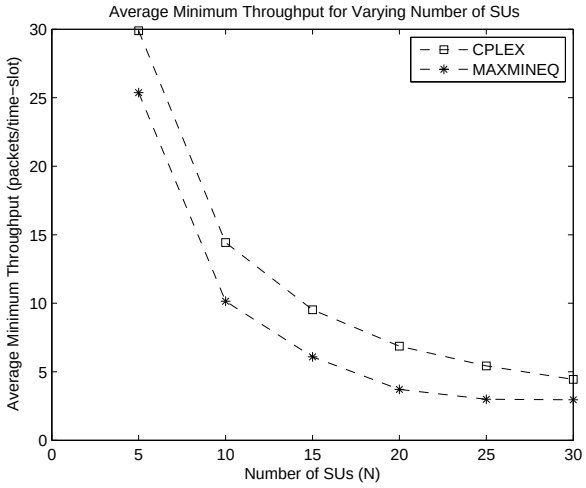


Figure 6.3. Average minimum throughput for varying number of SUs (N).

Table 6.1. Average minimum throughput and CPLEX gap values for varying number of SUs (N).

Number of SUs (N)	5	10	15	20	25	30
MAXMINEQ result	25.36	10.14	6.09	3.71	2.99	2.95
CPLEX result	29.89	14.42	9.52	6.86	5.43	4.44
CPLEX gap value	0.01% (default)	0.58%	0.6%	1.5%	1.5%	1.5%
$\alpha(G)$ value	2.55	4.06	4.3	4.92	5.42	6.62

average minimum throughput performance of MAXMINEQ with the results obtained from CPLEX. Recall that a statistical method is used in Chapter 5 to calculate the number of samples to take (the number of scheduling periods to run the simulations for) so that the sample mean of all the samples are ± 0.5 of the actual mean with a 95% confidence level. We use the same statistical method to determine the number of scheduling periods for each CPLEX experiment. The number of samples we take for MAXMINEQ in all the experiments is the same as the corresponding ones obtained via CPLEX.

Figure 6.3 shows the average minimum throughput of CPLEX and MAXMINEQ

where the number of SUs varies between 5 and 30. As the number of SUs increases, the minimum throughput value resulting from both algorithms decreases. This behavior is natural since the resources assigned per user decrease when more users share the same amount of resources. Table 6.1 provides the numerical minimum throughput values obtained from CPLEX and MAXMINEQ, which are essentially the same as the values shown in Figure 6.3. The minimum throughput performance of MAXMINEQ is close to the one of CPLEX. While obtaining the CPLEX values, we have experimentally determined the appropriate gap value in order to obtain the results in a reasonable amount of time. The default gap value used by CPLEX is 0.01%. As the number of SUs increases, simulations take much longer time. Therefore, we have increased the value of the gap parameter (*epgap*) to obtain satisfactory results in a reasonable amount of time. The gap values for each experiment are shown in Table 6.1. The table also shows the average $\alpha(G)$ values resulting from each experiment. As the number of SUs increases, the average $\alpha(G)$ values also increase due to the increasing diversity in the network; i.e., the probability that there exists an SU with better/worse U_{if} values increases as the number of SUs increases. However, experimental results show that $\alpha(G)$ values do not increase too much in practice. Furthermore, recall that $\alpha(G)$ is the worst case bound of the algorithm MAXMINEQ. Experimental results show that the average case behavior of MAXMINEQ in practice is much better than its worst case guarantee. Although the $\alpha(G)$ value increases as the number of SUs (N) increases, we do not observe an increase in the average performance difference of MAXMINEQ and CPLEX; in other words, the average case performance of MAXMINEQ does not deteriorate.

7. THROUGHPUT SATISFACTION BASED SCHEDULER

7.1. Problem Formulation

The throughput maximizing scheduler formulation in Equations 5.13-17 of the work in Chapter 5 maximizes the total throughput of all SUs in the CRN cell regardless of the minimum throughput requirements of the SUs. However, in a specific time period, each SU may be executing different applications with various requirements and priorities. For instance, a real-time application may require a relatively high minimum data rate (throughput) for proper operation. Unless the resources are allocated to this SU such that its minimum throughput requirement is met, the allocated resources may be useless for this SU. For example, an SU may require at least 1 Mbps data rate; that is to say, data rates between 10 Kbps and 100 Kbps may be indifferent from each other and may both be useless for this SU. If it is theoretically impossible to allocate its required minimum data rate to a particular SU, it makes more sense not to allocate any resources to this SU at all and allocate the resources instead to other SUs whose minimum data rate requirements can be met. On the other hand, an SU with a non-real-time application may be satisfied with a lower minimum data rate requirement. An SU with an audio or video application may need a particular data rate in a specific time period. In contrast, an SU with an e-mail application may be satisfied with a lower data rate in that same time period. Furthermore, allocation of data rate higher than the minimum requirement of the SU may not make that SU happier. For instance, a data rate more than 100 Mbps may not have an additional advantage for that SU; i.e., 101 Mbps and 200 Mbps may be equally good. Therefore, instead of allocating these excess resources to this SU, it makes more sense to allocate some of these resources to another SU which actually needs them. Hence, in a real centralized CRN system, the major goal of the CBS operator is to maximize the number of SUs which are satisfied in terms of throughput.

We formulate in Equations 7.1-4 the scheduling problem that maximizes the number of SUs which are satisfied in terms of throughput while at the same time assuring

that the PUs in the service area of the CBS are not disrupted, reliable communication between the SUs and the CBS is maintained, and no collisions occur between the SUs:

$$\max(\sum_{i=1}^{N} g(\Omega_i - \Omega_i^{min})) \tag{7.1}$$

$$s.t.$$

$$\Omega_i = \sum_{f=1}^{F}\sum_{t=1}^{T} \frac{U_{if} X_{ift}}{T} \tag{7.2}$$

$$(5.14),(5.15), \text{ and } (5.16) \tag{7.3}$$

$$X_{ift} \in \{0,1\}; \forall i \in \mathcal{N}, \forall f \in \mathcal{F}, \forall t \in \mathcal{T} \tag{7.4}$$

where Ω_i in constraint in Equation 7.2 denotes the throughput of SU i in this scheduling period and Ω_i^{min} denotes the minimum throughput requirement of SU i. Besides, $g(\cdot)$ in Equation 7.1 is the step function that indicates the utility of SU i; i.e., the satisfaction of SU i from the throughput Ω_i equals 1 if and only if $\Omega_i \geq \Omega_i^{min}$ and 0 otherwise. These types of utility functions are referred to in the literature as *inelastic* utility functions since the users have *hard* QoS requirements; i.e., utility equals zero when the QoS (throughput in our case) is lower than a prespecified threshold [110]. In the rest of this chapter, we refer to our formulated problem in Equations 7.1-3 as the MNSU (Maximum Number of Satisfied Users) problem.

Remark 7.1. We can convert the MNSU problem to a binary ILP as follows: Since the utility function $g((\Omega_i - \Omega_i^{min}))$ in Equation 7.1 produces either 1 or 0, we can introduce a variable s_i which equals 1 if SU i is satisfied in terms of throughput and 0 otherwise. Therefore, solution of MNSU formulated in Equations 7.1-3 is equivalent to the solution of the following binary ILP:

$$max(\sum_{i=1}^{N} s_i) \tag{7.5}$$

$$s.t.$$

$$\sum_{f=1}^{F}\sum_{t=1}^{T}\frac{U_{if}X_{ift}}{T} \geq \Omega_i^{min} \times s_i; \forall i \in N \tag{7.6}$$

$$(5.14),(5.15),(5.16), \text{ and } (7.4) \tag{7.7}$$

Constraint in Equation 7.6 models the behavior of the function $g(\Omega_i - \Omega_i^{min}))$. More precisely, if an SU i is satisfied; i.e., if $s_i = 1$, then its throughput (Ω_i) is greater than or equal to its minimum throughput requirement (Ω_i^{min}).

7.2. Computational Complexity

The Bin Covering *Problem:* Bin Covering problem is basically the covering version of the bin packing problem: Given n items with sizes $c_1, \cdots, c_n \in (0, 1]$, maximize the number of bins opened so that each bin has items summing to at least 1 [111]. Bin Covering problem cannot be approximated within any constant factor better than 2 unless $P = NP$. This result is based on the observation that a 2-approximation algorithm for the Bin Covering problem applied to instances in which the total size of the items is 2 would solve the Partition problem [112].

Theorem 7.1. Bin Covering $\preceq_{APX} MNSU$.

Proof. We can show that Bin Covering is a special case of MNSU as follows: Let $U_{if'} = U_{if''}$, $\forall f' \neq f''$ and $f', f'' \in \mathcal{F}$; i.e., let U_{if} values be the same for all f corresponding to a particular i. In other words, let us use U_i in lieu of U_{if}. Let us set $c_i = \frac{U_i}{\Omega_i} \leq 1$. Moreover, let us set $T = 1$ and $a_i \geq F, \forall i \in \mathcal{N}$. This special case corresponds to Bin Covering where c_i values correspond to the item sizes and the SUs corresponds to the bins. □

Corollary 7.1. *Since* Bin Covering *is* NP*-Hard in the strong sense [111],* MNSU

is also NP*-Hard in the strong sense.*

Corollary 7.2. *Since* Bin Covering *is* 2*-inapproximable [112],* MNSU *also cannot be approximated within any constant factor better than* 2 *unless* $P = NP$.

Theorem 7.2. *MMFS reduces to MNSU in polynomial time.*

Proof. Assume that we have a polynomial-time algorithm A for MNSU. We can use algorithm A to solve MMFS in polynomial time as follows: Let Ω^{UB}_{MMFS} be an upper bound for the result of MMFS. We can set our initial guess equal to this upper bound; i.e., we set $\Omega_i = \Omega^{UB}_{MMFS}\ \forall i \in \mathcal{N}$ and execute algorithm A. If the result equals N, then it means that the result for MMFS equals Ω^{UB}_{MMFS}. Otherwise, we can do a binary search between 0 and Ω^{UB}_{MMFS}, and execute algorithm A to check whether our guess was true or not. This way, we can find the optimum value for MMFS in polynomial time using algorithm A. □

Corollary 7.3. MNSU *is at least as hard as* MMFS.

Corollaries 7.1, 7.2, and 7.3 corroborate that MNSU is a computationally very difficult problem. Therefore, designing heuristic algorithms for MNSU is of paramount importance. To this end, heuristic algorithms for the MNSU problem have been developed in [9].

8. SPECTRUM SWITCHING DELAY AWARE SCHEDULER

8.1. Motivation

Switching the frequency of a wireless transceiver requires changing the input voltage of the voltage-controlled oscillator (VCO), which operates in a phase locked loop (PLL), to achieve the desired output frequency. Frequency switching speed is regarded as a critical performance parameter in many modern communication systems [113]. It is an important factor since it reduces the time available for data transmission. The frequency switching delay depends on the relative positions of the two channels on the radio spectrum [58, 59, 63, 113, 114] because as the difference between the reference and the final frequency is high, any small frequency drift in the reference oscillator is significantly magnified in the final synthesized frequency [113]. To alleviate this limitation, devices make this conversion in a step by step manner, which increases the time to switch to far away frequencies.

Hardware switching delay is a device dependent parameter. For instance, the TCI Model 7234 wideband SHF tuner has a tuning speed of 1 ms for each 500 MHz step [115], whereas the tuning speed of TCI 715 is 1 ms for each 10 MHz step [116]. As it was also pointed out by [117], some device specifications such as [118] do not explicitly report the dependence of the switching delay on the separation distance between the frequencies. This is because the operational frequency ranges of these devices are narrower, and hence the difference depending on the frequency separation distance is negligible. For instance, TCI 735 has two operation modes, one between 20 and 3000 MHz and the other between 3000 and 8000 MHz. In both cases, the typical switching delay is 1 ms and the maximum delay is 5 ms, which still implies that the switching delay is not constant. Both of these operational modes are narrower compared to 0.5-40 GHz range of the TCI 7234. The CRs of the future are envisioned to operate at a wide range of frequencies; therefore, frequency separation distance is inevitably

an important factor for the spectrum switching delay of the future CRs. Even if the CR device operates in a narrower frequency band and the spectrum switching delay is assumed to be constant, the entire mathematical analysis in this chapter remains valid except for a few modifications about how the switching delay is calculated. In essence, our proposed algorithm can be used with any switching delay model; the only difference is the part where the switching delay is calculated. We explain this fact in detail in Section 8.3.

Besides the requirement that CRs of the future have to operate in a wide range of frequencies, there are numerous other factors that render our spectrum switching delay aware scheduler vital:

(i) Smaller hardware spectrum switching delay means more expensive CR equipment. Our proposed scheduler obviates the need to have smaller hardware switching delay, and hence plays a role in decreasing the cost of the CR devices. Since cost will be an important factor when CRs appear in the market, having a more intelligent scheduling algorithm such as our proposed algorithm in this chapter allows us to utilize less expensive devices. This way, CR equipment manufacturers can increase their profits. Since a cheaper CR device will enable more people to start using CR devices, our proposed algorithm will also have an impact on increasing the market penetration of CRs when they first appear in the market.

(ii) Smaller spectrum switching delay means heavier CR equipment because more filters are needed in the hardware to achieve smaller delay. For instance, the spectrum processor of TCI 745 [119] is 30 kg, whereas the one of TCI 7234 [115] is 6 kg. These weights are impractical for future CRs since they cannot be, for instance, a mobile handset. Achieving small spectrum switching delay with reasonable equipment weights remains a challenge. Since the performance of our proposed algorithm is robust to changes in the spectrum switching delay, it obviates the need to have small spectrum switching delay. Therefore, by using our proposed algorithm in the software, equipment manufacturers can use less number of filters in the hardware and still achieve high throughput performance. Therefore, our algorithm also helps decrease CR device weights by obviating the

need to accomplish smaller spectrum switching delay and hence enabling the usage of less number of filters in the hardware.

(iii) Smaller spectrum switching delay means larger device dimensions because filters occupy space. For instance, the latest TCI 745 [119] has dimensions of $8U$ high, rack mount (14"Hx19"Wx22"D), which is far from being a handheld CR device. Hence, our proposed spectrum switching delay aware scheduling algorithm in this chapter helps achieve more reasonable device sizes for the future CRs by obviating the need to use large number of filters for less switching delay.

(iv) In wireless networks, scheduling decisions are typically made for a duration where the network conditions are fairly stable. In CRNs, this duration also depends on PU activities since SUs are obliged not to disturb PUs. A swiftly changing PU activity and spectral environment makes the scheduling periods shorter since the CR devices have to adapt their transmission parameters such as rate and power more quickly in order not to disturb PUs. In the time-slotted scheduling model in this thesis, we take a time slot length as 100 ms, which is appropriate for slowly varying spectral environments like the TV bands. For instance, TCI 7234 [115] has a switching delay of 1 ms per 500 MHz. Switching the operation frequency from 0.5 GHz to 40 GHz requires 79 ms delay. Considering that the time slot length is 100 ms, 79 ms of switching delay implies that only 21% of the time slot can actually be utilized for data transmission. In other words, the impact of switching delay can be significant even in a slowly varying spectral environment where a 100 ms of time slot length is sufficient. As the spectral environment varies more frequently, time slot lengths have to decrease and hence, the impact of the spectrum switching delay intensifies.

(v) Different CR users may opt for CR devices with different spectrum switching delay. The entity that is responsible for executing the scheduling algorithm usually cannot mandate the CR users to use a specific device. For instance, in a centralized CRN where the cognitive base station (CBS) is responsible for the management of the SUs in its service area, the scheduler resides at the CBS. If some CR users have less expensive devices with larger spectrum switching delay and if the CBS operator does not consider switching delay in its scheduling algo-

rithm, it may unwittingly end up with providing coarse QoS to these CR users, which are the customers of the CBS operator. Some customers may even starve in terms of throughput without the CBS operator having intended to do so.

(vi) Our simulation results indicate that high throughput savings are achieved even when the number of CR users is as small as 10. Furthermore, the throughput savings of our proposed algorithm increase as the number of frequencies increases. The increased impact of switching delay on the throughput performance as the number of frequencies increases indicates that our proposed algorithm will become increasingly useful as the CR paradigm proliferates in real applications.

(vii) The range of frequencies that a centralized CBS cell operates in is not static. A central entity called "spectrum broker" or "spectrum policy server" coordinates the spectrum usage of several base stations by periodically (usually in a medium term) allocating frequency pools to the base stations [3,16,120–123]. To this end, auction theoretic or game theoretic techniques are usually employed [121–123]. Therefore, it is not feasible for the hardware providers of the mobile terminals to design the cognitive radio hardware customized for the frequency range of each and every CBS cell. Furthermore, CR devices may need to handover from one CBS cell to another. Having a CR device specifically designed for operating in a narrow range of frequencies hinders the handover mechanisms between different CBS cells. Moreover, the ultimate goal of the CR technology is to provide an interoperable "universal wireless device" that can seamlessly handle a wide range of frequencies [124]. To this end, CR devices should be able to operate in a wide range of frequencies.

8.2. Problem Formulation

The throughput maximizing scheduler formulation in Equations 5.13-17 of the work in Chapter 5 assumes that no delay occurs when an SU switches from a frequency to another frequency. However, in reality, some portion of the subsequent time slot is inevitably wasted to tune to the new frequency; therefore, only the remaining portion of the next time slot can be used for actual data transmission. It may even be the

case that the time it takes to switch to the new frequency is greater than or equal to the time slot length, which means that no packets can actually be sent using the new frequency. Since the scheduling decisions are known in advance by SUs, they should not waste time and energy in vain to switch to the new frequency; they should instead stay in the same frequency. On the other hand, the new frequency band might be more advantageous in terms of throughput by having a higher U_{if} value. The question is whether the delay incurred while switching to the new frequency band offsets this throughput advantage or not. If the throughput advantage of the new frequency band outweighs the disadvantage of throughput losses due to switching delay, then the SU may still prefer switching to the new frequency. Therefore, there is a tradeoff here; i.e., switching to the new frequency band may or may not be advantageous depending on the circumstances (switching delay and the channel conditions (U_{if} values) of the old and new frequencies). Furthermore, we also need to keep track of the information about which interface is assigned to which frequency since each interface experiences different switching delays depending on the frequency that it was assigned to in the previous time slot. In this chapter, we extend the work in Chapter 5 to account for the spectrum switching delay, which depends on the distance between the used frequencies.

Let us denote by C_{iat} the frequency that interface a of SU i is assigned to in time slot t. Then, $C_{iat} = \sum_{f=1}^{F} f\, X_{iaft}$, where X_{iaft} is a binary variable that equals 1 if frequency f is assigned to the interface a of SU i in time slot t, and 0 otherwise. Let us denote by Δ_{iaft} the absolute difference in terms of the number of frequencies that interface a of SU i has to step to use frequency f in time slot t. Note that the interfaces do not have to be assigned some frequency in every time slot; in other words, it is possible for an interface not to be assigned any frequency in some time slot. If interface a of SU i has not been assigned some frequency for time slot t, then we say that t is a *silent time slot* for the interface a of SU i. Otherwise, we say that t is a *busy time slot* for the interface a of SU i. A time slot t may be a silent time slot for some interface but a busy time slot for another interface. Let us denote by m_{iat} the index of the busy time slot before time slot t. If t is the first busy time slot in the scheduling period, then $m_{iat} = 0$. In other words, $m_{iat} = \max_{\substack{\exists f s.t. X_{iafj}=1 \\ j<t}} \{j, 0\}$. Then the number

of silent time slots between the current time slot t and the previous busy time slot for interface a of SU i equals $t - m_{iat} - 1$. If $m_{iat} = 0$, i.e., if t is the first busy time slot for interface a of SU i in the scheduling period, then in this case, as in [125], we assume that $\Delta_{iaft} = 0$. In other words, we assume for simplicity that the interfaces are pretuned to the firstly used frequency. In practice, this delay in the firstly used time slot may depend on various other factors such as MAC protocol. If a single interface is used for both data transmission and control traffic, the interface may have to tune to the frequency band of the common control channel (CCC) during the time between consecutive scheduling periods. How frequently the tuning to the CCC is performed and which frequency the CCC uses depends on the protocol implementation. To isolate us from the possible influence of these factors, as in [125], we assume that the interfaces are pretuned to the firstly used frequency.

On the other hand, if a frequency f is not the firstly used frequency for interface a of SU i and there are silent time slots preceding time slot t, i.e., $0 < m_{iat} < t-1$, then interface a uses these silent time slots to switch to the new frequency f. Scheduling decisions are made by the CBS for the duration of a scheduling period, which consists of T time slots. This scheduling information is then sent to the SUs through the CCC. Therefore, SUs know the scheduling decisions (which frequencies are assigned to them in which time slots) before the beginning of the first time slot of the scheduling period. Because the scheduling decisions are known by SUs in advance, they can use these silent time slots to switch to the new frequency. If the number of silent time slots are enough to achieve the entire frequency switching, SU becomes ready to use the new frequency in the upcoming busy time slot. In this case, SU does not waste any portion of the busy time slot for frequency switching and hence it can use the entire busy time slot for data transmission. Otherwise, SU utilizes the silent time slots to achieve some portion of the frequency switching. The remaining switching is completed at the beginning of the next busy time slot. If the silent time slots and portions of the busy time slot are still not enough to achieve the frequency switching and no available time remains in the busy time slot for data transmission, then it means that no packets can be sent by the SU using the new frequency in the busy time slot.

Let us denote by β the switching delay in terms of milliseconds for each unit step in the frequency range. The value for β is hardware dependent, for instance the delay is taken as 1 ms/10 MHz in [116]. In this case, if the frequencies are separated with a 10 MHz difference (for instance $f = 1$ corresponds to 800 MHz, $f = 2$ corresponds to 810 MHz etc.), then $\beta = 1\, ms$. If $m_{iat} \neq 0$, then $\left|f - C_{iam_{iat}}\right|$ is the number of frequencies that SU i needs to sweep to tune to the new frequency f in time slot t. Since there are $(t - m_{iat} - 1)$ number of silent time slots, each one of which has a length of T_s milliseconds, then $\dfrac{(t - m_{iat} - 1)T_s}{\beta}$ portion of the frequency band is switched to during the silent time slots. The remaining portion that needs to be switched during the busy time slot t is equal to $(\left|f - C_{iam_{iat}}\right| - \dfrac{(t - m_{iat} - 1)T_s}{\beta})^+$, where $(x)^+ = \max(0, x)$. Recall that Δ_{iaft} denotes the absolute difference in terms of the number of frequencies that interface a of SU i has to step to use frequency f in time slot t. Then, $\Delta_{iaft} = 0$ if $m_{iat} = 0$. Otherwise, $\Delta_{iaft} = (\left|f - C_{iam_{iat}}\right| - \dfrac{(t - m_{iat} - 1)T_s}{\beta})^+$. Similar to the works in [58, 59], we use in the calculation of Δ_{iaft} a linear relationship between the switching delay and the wideness of the difference in the assigned frequencies. In essence, any switching delay model is applicable to our proposed algorithm with a slight modification. For instance, if it is assumed that the switching delay is constant, then $\Delta_{iaft} = 0$ if $m_{iat} = 0$ or $f = C_{iam_{iat}}$, and $\Delta_{iaft} = \big(1 - \dfrac{(t - m_{iat} - 1)T_s}{\beta}\big)^+$ otherwise. The rest of the analysis remains the same. We elaborate on this in detail in Section 8.3. In the rest of this work, we consider the case where the spectrum switching delay is not constant and depends on the frequency separation distance. Finally, if we denote by B_{iaft} the maximum number of packets that can be sent by interface a of SU i if it is tuned to frequency f in time slot t, then $B_{iaft} = \left\lfloor \big(1 - \dfrac{\beta \times \Delta_{iaft}}{T_s}\big)^+ V_{if} \right\rfloor$.

We then extend the throughput maximizing scheduler formulation in Chapter 5 to account for the spectrum switching delay and formulate the following binary integer

program:

$$\max(\sum_{i=1}^{N}\sum_{a=1}^{a_i}\sum_{f=1}^{F}\sum_{t=1}^{T}\frac{B_{iaft}X_{iaft}}{T}) \tag{8.1}$$

$$s.t.$$

$$\sum_{a=1}^{a_i}\sum_{f=1}^{F}\sum_{t=1}^{T}X_{iaft} \geq 1; \forall i \in \mathcal{N} \tag{8.2}$$

$$\sum_{i=1}^{N}\sum_{a=1}^{a_i}X_{iaft} \leq 1; \forall f \in \mathcal{F}, \forall t \in \mathcal{T} \tag{8.3}$$

$$\sum_{f=1}^{F}X_{iaft} \leq 1; \forall a \in \mathcal{A}_i, \forall i \in \mathcal{N}, \forall t \in \mathcal{T} \tag{8.4}$$

$$C_{iat} = \sum_{f=1}^{F} f\ X_{iaft}; \forall i \in \mathcal{N}, \forall a \in \mathcal{A}_i, \forall t \in \mathcal{T} \tag{8.5}$$

$$m_{iat} = \max_{\substack{\exists f s.t. X_{iafj}=1 \\ j<t}}\{j, 0\} \tag{8.6}$$

$$\Delta_{iaft} = \begin{cases} 0, \text{ if } m_{iat} = 0 \\ (\left|f - C_{iam_{iat}}\right| - \frac{(t - m_{iat} - 1)T_s}{\beta})^+, \text{ o.w.} \end{cases} \tag{8.7}$$

$$B_{iaft} = \left\lfloor (1 - \frac{\beta \times \Delta_{iaft}}{T_s})^+ V_{if} \right\rfloor \tag{8.8}$$

$$X_{iaft} \in \{0, 1\}; \forall i \in \mathcal{N}, \forall f \in \mathcal{F}, \forall t \in \mathcal{T} \tag{8.9}$$

where $\mathcal{N}$, $\mathcal{A}_i$, a_i, $\mathcal{F}$ and $\mathcal{T}$ are as defined previously. The objective function in Equation 8.1 maximizes the total average throughput of all SUs in the CRN cell, whereas constraint in Equation 8.2 guarantees that each SU is assigned at least one time slot. Moreover, Equation 8.3 represents the constraint that at most one interface can use a frequency in a specific time slot, and Equation 8.4 denotes that each interface can tune to at most one frequency in a time slot. In other words, purposes of the constraints in Equations 8.2, 8.3, and 8.4 are the same as purposes of the constraints in Equations 5.14, 5.15, and 5.16, respectively. The reason for having the same constraints is because our goal is to evaluate the impact of the spectrum switching delay awareness more effectively by comparing our spectrum switching delay aware scheduler with the scheduler in Equations 5.13-17 of Chapter 5. In order to be able to make an effective comparison,

all other features of both schedulers except switching delay awareness have to be the same. Therefore, we have constraints in Equations 8.2, 8.3, and 8.4. Furthermore, the constraints in Equations 8.5, 8.6, 8.7, and 8.8 are as explained previously. Note here that because B_{iaft} values depend on the frequency assignments in the previous time slots, the objective function in Equation 8.1 is nonlinear. Nonlinear binary integer programming is in general known to be computationally hard in the literature [126].

8.3. Proposed Algorithm

In this section, we propose a polynomial time heuristic algorithm to address the problem in Equations 8.1-9. In the rest of this chapter, we refer to our proposed algorithm by S^{2}DASA (Spectrum Switching Delay Aware Scheduling Algorithm).

We outline the main steps of S^{2}DASA in Figure 8.1. The set $\Phi \subseteq \mathcal{N}$ symbolizes the set of SUs which have not yet been assigned any time slot during the execution of the algorithm. B'_{iaf} represents the benefit (in terms of the maximum number of packets that can be transmitted) that interface a of SU i receives for using frequency f in that particular time slot. In Step 1, S^{2}DASA initializes the set Φ to $\mathcal{N}$ since none of the SUs have been assigned a time slot at the beginning of the algorithm. Moreover, Step 1 initializes the benefit values B'_{iaf} for the first time slot to $\lfloor V_{if} \rfloor$ since no switching delay occurs in the first time slot. S^{2}DASA makes the frequency assignment sequentially for each time slot. At the beginning of each time slot, the algorithm chooses the set Ψ, which is the set of SUs that have to be assigned at least one frequency for that particular time slot. $B^{'}_{iaf}$ indicates the maximum number of packets that can be sent by interface a of SU i if it is tuned to frequency f in that particular time slot. To determine the set Ψ, we introduce a metric called Γ_i to select the SUs with relatively good $B^{'}_{iaf}$ values averaged over all of their interfaces. Step 3 of S^{2}DASA assigns the Γ_i value by determining the average benefit value per interface for each SU in the set Φ. In each time slot, the set $\Psi \subseteq \Phi$ with relatively high Γ_i values is selected, and every SU in this set Ψ is guaranteed to be assigned with a frequency in that time slot. Steps 4 and 5 of S^{2}DASA assign all the SUs in the set Φ to the set Ψ if the number of SUs in the set Φ is less than or equal to $\lceil \frac{N}{T} \rceil$. Otherwise, in Step 7, $\lceil \frac{N}{T} \rceil$ number of SUs that

have the largest Γ_i values are selected and added to the set Ψ. In Step 9, S^{2}DASA runs the following ILP:

$$\max(\sum_{i=1}^{N}\sum_{a=1}^{a_i}\sum_{f=1}^{F} B'_{iaf}X'_{iaf}) \tag{8.10}$$

$$s.t. \quad \sum_{a=1}^{a_i}\sum_{f=1}^{F} X'_{iaf} \geq 1; \forall i \in \Psi \tag{8.11}$$

$$\sum_{i=1}^{N}\sum_{a=1}^{a_i} X'_{iaf} \leq 1; \forall f \in \mathcal{F} \tag{8.12}$$

$$\sum_{f=1}^{F} X'_{iaf} \leq 1; \forall a \in \mathcal{A}_i, \forall i \in \mathcal{N} \tag{8.13}$$

$$X'_{iaf} \in \{0,1\}; \forall i \in \mathcal{N}, \forall f \in \mathcal{F}, \forall t \in \mathcal{T} \tag{8.14}$$

where X'_{iaf} is a binary decision variable that equals 1 if interface a of SU i transmits using frequency f, and 0 otherwise. At the end of the decision for every time slot, B'_{iaf} values are calculated and updated (in Steps 15-17) for the subsequent time slot. Notice here that B'_{iaf} values are input variables rather than decision variables in the optimization problem in Equations 8.10-13. Constraint in Equation 8.11 ensures that the SUs in the set $\Psi \subseteq \Phi$ are assigned at least one frequency, which meets the constraint in Equation 8.2 for the SUs in Ψ. As in Equations 8.3 and 8.4, constraints in Equations 8.12 and 8.13 ensure that at most one interface can use a frequency, and each interface can tune to at most one frequency (in that time slot).

Lemma 8.1. *If the problem in Equations 8.1-9 has a feasible solution, then S^{2}DASA gives a feasible solution.*

Proof. For all the SUs in the set Φ, Step 11 of S^{2}DASA checks if the ILP execution in Step 9 has assigned at least one frequency to that SU. If yes, then Step 12 removes this particular SU i from the set Φ since it is no longer an SU that is not assigned any time slot. Note here that not only the SUs in the set Ψ, but also the other SUs that are assigned some frequency by the ILP execution in Step 9 are removed from the set Φ. In the worst case, it happens that S^{2}DASA assigns some frequency to only the SUs in Ψ in every time slot. If N is a multiple of T, then N/T number of SUs are assigned

Require: $\mathcal{N}$, $\mathcal{F}$, $\mathcal{T}$, $\mathcal{A}_i$, V_{if}.

Ensure: X_{iaft} values $\forall i \in \mathcal{N}, \forall a \in \mathcal{A}_i, \forall f \in \mathcal{F}, \forall t \in \mathcal{T}$.

$\Phi \leftarrow \mathcal{N}$, $B'_{iaf} \leftarrow \lfloor V_{if} \rfloor, \forall i \in \mathcal{N}, \forall a \in \mathcal{A}_i, \forall f \in \mathcal{F}$
for $t = 1$ to T **do**
    $\Gamma_i \leftarrow \sum_{a=1}^{a_i} \sum_{f=1}^{F} B'_{iaf}/a_i$, $\forall i \in \Phi$
    if $|\Phi| \leq \lceil \frac{N}{T} \rceil$ **then**
        $\Psi \leftarrow \Phi$
    else
        Sort all $i \in \Phi$ wrt. Γ_i. Add $\lceil \frac{N}{T} \rceil$ number of i's that have the largest Γ_i values to Ψ; i.e., $|\Psi| = \lceil \frac{N}{T} \rceil$
    end if
    Run ILP in Equations 8.10-13 with $\Psi, B'_{iaf}, \mathcal{N}, \mathcal{F}, \mathcal{A}_i$, and obtain $X'_{iaf}, \forall i \in \mathcal{N}, \forall a \in \mathcal{A}_i, \forall f \in \mathcal{F}$
    for all $i \in \Phi$ **do**
        if $\sum_{a=1}^{a_i} \sum_{f=1}^{F} X'_{iaf} \geq 1$ **then**
            $\Phi \leftarrow \Phi - \{i\}$
        end if
    end for
    $X_{iaft} \leftarrow X'_{iaf}, \forall i \in \mathcal{N}, \forall a \in \mathcal{A}_i, \forall f \in \mathcal{F}$
    Calculate $B_{iaf(t+1)}$ by using X_{iaft}
    $B'_{iaf} \leftarrow B_{iaf(t+1)}, \Psi \leftarrow \emptyset$
end for
return $X_{iaft}, \forall i \in \mathcal{N}, \forall a \in \mathcal{A}_i, \forall f \in \mathcal{F}, \forall t \in \mathcal{T}$

Figure 8.1. S^{2}DASA (Algorithm for the problem in Equations 8.1-9).

a frequency in every time slot, and clearly all the N number of SUs are guaranteed to satisfy constraint in Equation 8.2 at the end of the algorithm. If N is not a multiple of T, then $|\Psi| = \lceil \frac{N}{T} \rceil$ in the first $\left\lfloor \frac{N}{\lceil \frac{N}{T} \rceil} \right\rfloor$ time slots, and condition in line 4 of the algorithm holds true in the next time slot. Therefore, constraint in Equation 8.2 is met for all SUs after $\left\lfloor \frac{N}{\lceil \frac{N}{T} \rceil} \right\rfloor + 1 = \left\lceil \frac{N}{\lceil \frac{N}{T} \rceil} \right\rceil$ number of time slots. Since $\left\lceil \frac{N}{\lceil \frac{N}{T} \rceil} \right\rceil \leq T$, S^{2}DASA returns a feasible solution to the problem in Equations 8.1-9 as long as the original problem has a feasible solution. Notice here that the problem in Equations 8.1-9 has a feasible solution as long as $F \times T \geq N$ because otherwise it is impossible to assign every SU at least one time slot. Hence, S^{2}DASA returns a feasible solution as long as $F \times T \geq N$. In other words, S^{2}DASA guarantees that each SU is assigned at least one time slot. □

Steps 15-17 serve the purpose of calculation and update of the B'_{iaf} values for the next time slot. More precisely, Step 15 of S^{2}DASA sets the X_{iaft} values for that particular time slot t according to the values of X'_{iaf} as a result of the ILP execution in Step 9. Step 16 calculates $B_{iaf(t+1)}$ values for the subsequent time slot $(t+1)$ according to the X_{iaft} values of the current time slot t and possibly the previous time slots by using the formulas specified in Equations 8.5-8. Step 17 updates the values for B'_{iaf} to be used in the iteration of the subsequent time slot.

Every switching delay model will produce a switching delay value, which may depend on the frequency assignments in the previous time slot. If a switching delay model different from the linear model (either constant switching delay or some other model) needs to be used, the only part that needs to be modified in our algorithm is Step 16. In other words, when another switching delay model is used, only the resulting switching delay value used by the calculation of $B_{iaf(t+1)}$ in Step 16 changes; the rest of the algorithm remains the same. In the rest of this chapter, when we mention S^{2}DASA, we implicitly refer to the varying switching delay case with linear model. If a constant delay or some other delay model is used, we explicitly state the name of the switching delay model.

Moreover, Step 17 sets the value of Ψ to the empty set since the SUs in the current Ψ have already been assigned at least one time slot and the Ψ values need to be reconstructed in the next time slot. The algorithm terminates after assignments are made for all time slots.

Theorem 8.1. *ILP in Equations 8.10-14 is solvable in polynomial time.*

Proof. Build an edge weighted bipartite (multi)graph $G = (\mathscr{N}, \mathscr{F}, \mathscr{E})$ as follows: Add a vertex v_i to $\mathscr{N}$, $\forall i \in \mathcal{N}$. Add a vertex v_f to $\mathscr{F}$, $\forall f \in \mathcal{F}$. For each $B'_{iaf} \neq 0$, add an edge $\{v_i, v_f\}$ to $\mathscr{E}$ with weight $B'_{iaf}, \forall i \in \mathcal{N}, \forall a \in \mathcal{A}_i, \forall f \in \mathcal{F}$. Let $\mathbf{I}$ be the following function associating an interval of natural numbers with each vertex in G: $\mathbf{I}(v_i)$=[1,a_i] $\forall i \in \Psi$, $\mathbf{I}(v_i)$=[0,a_i] $\forall i \notin \Psi$, $\mathbf{I}(v_f)$=[0,1] $\forall f \in \mathcal{F}$. The problem of finding a sub(multi)graph that maximizes the total edge weights while respecting the constraints about the interval of allowed degrees for each vertex is known to be solvable for any (multi)graph in polynomial time [91, 92]. In particular, if the (multi)graph is bipartite, then the solution for the ILP representing this problem is equal to the solution of its linear program (LP) because the incidence matrix of a bipartite graph is totally unimodular [91]. If an edge with weight B'_{iaf} is selected in the resulting sub(multi)graph, then X'_{iaf} is set to 1; otherwise, it is set to 0. Hence, the theorem holds. □

Since the other steps of S^{2}DASA are clearly solvable in polynomial time, Theorem 8.1 implies that our proposed heuristic (S^{2}DASA) is solvable in polynomial time. Furthermore, S^{2}DASA is based on running an LP for each time slot. Ellipsoid algorithm [127] can be used to solve LP in polynomial time. In practice, simplex algorithm [127] is widely known to be an efficient algorithm to solve LPs in spite of its exponential complexity.

8.4. Simulation Results

We employ the same simulation conditions as in Chapter 5. The dynamicity of the spectral environment stems from two factors: Physical mobility of the SUs and

PUs and the changing spectrum occupancy behavior of the PUs. U_{if} values for each scheduling period are possibly different due to the changes in the physical mobility and PU spectrum occupancies. If an SU becomes closer to another PU due to physical mobility, it may need to change its operation frequency in order not to disturb the PU. Likewise, if a PU in the vicinity of the SU starts using a frequency that SU was using, the SU needs to switch to another band.

As in Chapter 5, we select the p_S value in the PU spectrum occupancy model as 0.9. Hence, our simulation results demonstrate that switching delay is an important factor even in a slowly varying spectral environment.

The original simulation results in Chapter 5 for the throughput maximizing scheduler, which we refer to here by *upper bound THR_MAX*, do not consider the spectrum switching delay. If we calculate the throughput values of the resulting frequency and time slot assignments in Chapter 5 according to the spectrum switching delay model outlined in Section 8.2, which depends on the difference between the current and the previously assigned frequency, we obtain the actual throughput values of the results in Chapter 5, which we refer to here by *conventional THR_MAX*.

As in Chapter 5, we assume that all the PUs and SUs move with a constant velocity of V_p and V_s, respectively. We denote the number of PUs in the cell by M. In all experiments in this chapter, we take $M = 20$, $V_p = V_s = 13$ m/s, and $a_i = 3$ $\forall i \in \mathcal{N}$. In Chapter 5, a statistical method is used to calculate the number of samples to be taken so that the sample mean of all the samples are ± 0.5 of the actual mean with a 95% confidence level. The number of samples we take for S^{2}DASA in all the experiments is the same as the corresponding ones in upper bound THR_MAX, as calculated in Chapter 5.

We evaluate the impact of F and β in two different sets of experiments. In the first one, we vary the value of β and plot the results for F=18 and F=30. We set N=30 and compare the results of upper bound THR_MAX, conventional THR_MAX, and S^{2}DASA in Figure 8.2. The results show that for both F=18 and F=30, S^{2}DASA

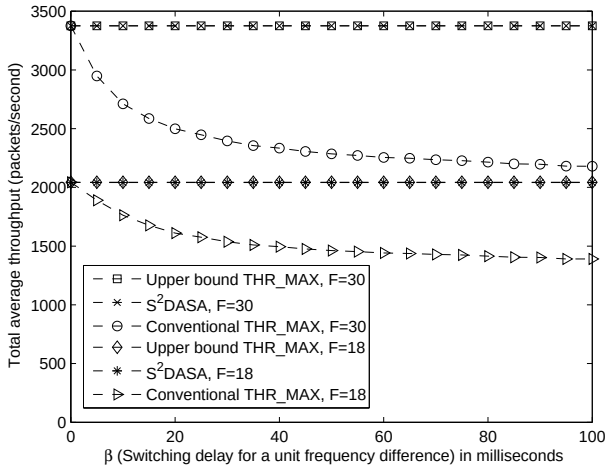


Figure 8.2. Average total throughput of all schedulers for varying β.

yields very close performance to upper bound THR_MAX, which can be regarded as an upper bound to the optimization problem in Equations 8.1-8. Since the results are very close, they look like almost the same in the figure; however, there are subtle differences in reality. For instance, for F=30, the throughput that upper bound THR_MAX yields is 3376.6 packets/time slot, whereas the throughput that S^{2}DASA yields for $\beta = 50$ is 3375.6 packets/time slot. In all of the simulation results presented in this paper, the throughput performance of S^{2}DASA is only slightly less than the one of upper bound THR_MAX; therefore, they cannot be visually differentiated in the figures. We also observe in Figure 8.2 that the decrease in conventional THR_MAX as β increases for F=30 is larger than the decrease for F=18. This implies that S^{2}DASA results in higher savings from throughput as F and β increase. Furthermore, we can also observe that the performance difference between S^{2}DASA and upper bound THR_MAX remains very little as β increases. These results demonstrate that the performance of our proposed algorithm S^{2}DASA is not only very close to its upper bound, but also robust to changes in the hardware switching delay.

In the second set of experiments, we set N=30 and vary the value of F. We then plot the results in Figure 8.3 for $\beta = 1$, $\beta = 5$, and $\beta = 10$ ms. We again observe that the performance of S^{2}DASA is very close to upper bound THR_MAX, in addition to being robust to changes in the switching delay. As in Figure 8.2, the

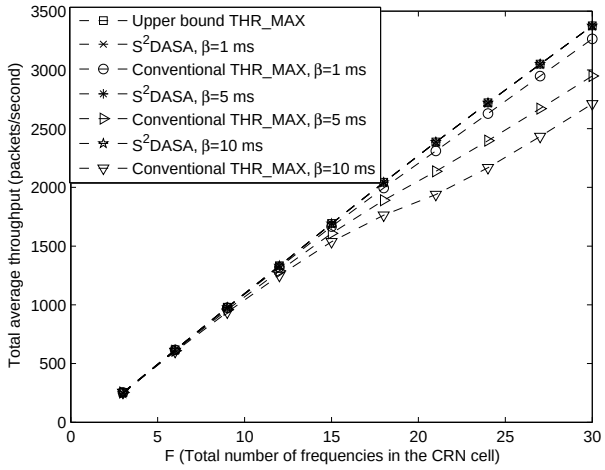
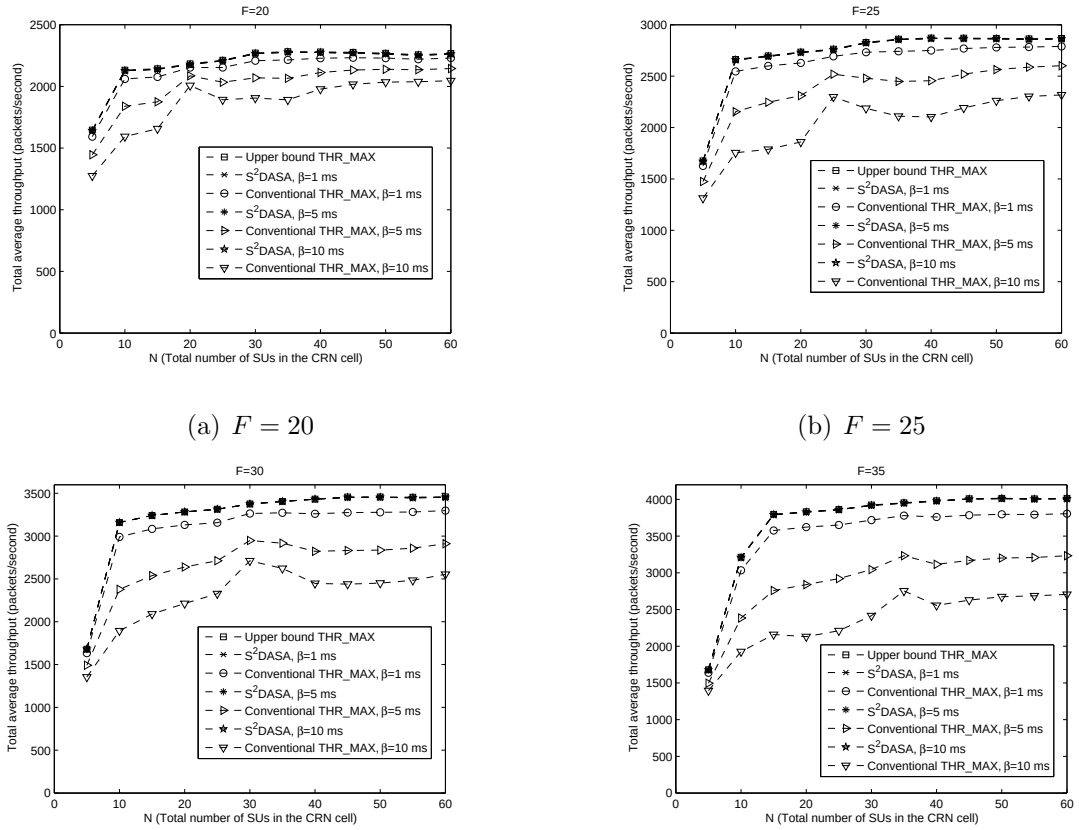


Figure 8.3. Average total throughput of all schedulers for varying number of frequencies (F).

throughput difference between S^{2}DASA and upper bound THR_MAX is so small that the two results cannot be visually differentiated in Figure 8.3. We see in Figure 8.3 that throughput increases linearly with the number of frequencies in all cases. Furthermore, the performance difference between S^{2}DASA and conventional THR_MAX widens as the number of frequencies increases; hence, throughput savings achieved by our proposed algorithm S^{2}DASA increase as the number of frequencies increases. This result demonstrates that switching delay becomes an even more important factor as the CRNs proliferate by operating in a wider range of frequencies. Therefore, our proposed algorithm S^{2}DASA is promising to be even more useful in the CRNs of the future.

In the third set of experiments, we evaluate the impact of varying number of SUs (N). Figure 8.4 shows the total average throughput for different values of F. Figures 8.4a, 8.4b, 8.4c, and 8.4d show the results for F=20, F=25, F=30, and F=35, respectively. We observe in these figures that the throughput savings achieved by our algorithm are significant even when N=10 and they remain significant as N increases. We also observe that there is a ripple effect in conventional THR_MAX at the point where $N = F$; i.e., the real throughput at $N = F$ is higher than the real throughput at $N = F - 5$ and $N = F + 5$. The ripple is more evident for the $\beta = 10$ case. Recall that in all the simulations, $a_i = 3\ \forall i \in \mathcal{N}$ and there is a constraint that each SU has to be



(a) $F = 20$ (b) $F = 25$

(c) $F = 30$ (d) $F = 35$

Figure 8.4. Average total throughput of all schedulers for varying number of secondary users (N).

assigned at least one time slot. Until the point where $N = F$, it gets more difficult to assign each SU at least one time slot as the number of SUs increases and the scheduler mostly uses more than one interface. It happens most of the time that an SU is assigned a frequency for a particular time slot but not assigned any frequency in the subsequent time slot because the other SUs need to be assigned some frequency in order to satisfy the constraint that each SU is assigned at least one time slot. Notice here that the scheduler does not mandate each SU to be assigned a frequency in each time slot, but only mandates that each SU is assigned at least one time slot during the course of the entire scheduling period. When an SU is not assigned a frequency in a particular time slot but assigned a frequency in the subsequent time slot, the hardware switching delay has less impact on the throughput performance since there is more time available to achieve the frequency switching, which is represented by our formulation for Δ_{iaft} in

Equation 8.7. Since this situation occurs more frequently as N value approaches F, we see an increase in the throughput values of conventional THR_MAX. When $N > F$, less number of interfaces are used and the impact of switching delay increases. Let us call an interface *active* if it is used at least once for data transmission during the entire scheduling period. Recall that time slot t is called a silent time slot for an active interface a of SU i if it is not assigned any frequency in time slot t. Recall also that time slot t is otherwise called a busy time slot for interface a of SU i. Note here that this time slot might be in use by some other interface of this SU or of another SU. When we divide the total number of silent time slots in a scheduling period by the number of active interfaces, we find the *"average number of silent time slots per active interface"*. Then we find the sample mean of this *"average number of silent time slots per active interface"* over all iterations (all samples). The average number of silent time slots for $F = 20$ are 4.51%, 10.39%, 11.62%, 10.5%, and 10.28% and the average number of active interfaces are 2.48, 2.34, 1.6, 1.5, and 10.28 for $N = 10$, $N = 15$, $N = 20$, $N = 25$, and $N = 30$, respectively. The ripple effect at $F = N$ can also be observed for the average number of silent time slots per active interface. We have observed the same situation for $F = 25$, $F = 30$, and $F = 35$. The similarity of this behavior to the throughput performance corroborates our explanation for the ripple effect observed at the real throughput performance at $F = N$.

In the fourth set of experiments in Figure 8.5, we compare our algorithm S^{2}DASA with upper bound THR_MAX, conventional THR_MAX and S^{2}DASA with the constant delay scenario. Figures 8.5a, 8.5b, and 8.5c show the results for $\beta = 1$, $\beta = 5$, and $\beta = 10$ cases, respectively. We can see that S^{2}DASA (varying delay) performs better than S^{2}DASA with constant delay case in all scenarios. Furthermore, its performance is close to upper bound THR_MAX. The results corroborate that the assumption of constant switching delay in a CRN setting can lead to poor throughput performance. In some cases, assuming that the switching delay is constant can lead to even lower throughput than conventional THR_MAX. The superiority of S^{2}DASA with varying switching delay over S^{2}DASA with constant switching delay becomes more evident as β (hardware switching delay) increases. To sum up, Figure 8.5 demonstrates that the gist of our paper, which is to take into account in the scheduling algorithm for CRNs

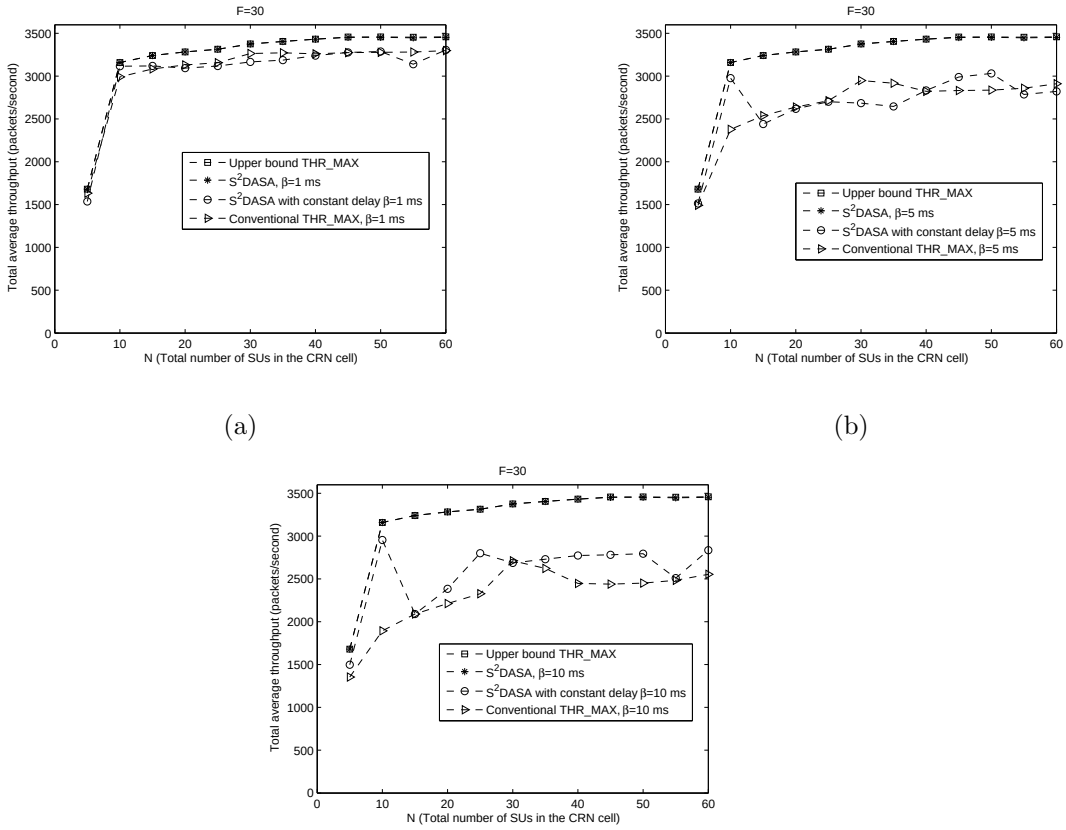


Figure 8.5. Average total throughput comparison with constant switching delay.

the increasing spectrum switching delay due to switching to further away frequencies, is vital in CRNs.

9. CONCLUSIONS AND FUTURE WORK

9.1. Summary of Contributions

In this thesis, we propose a scheduling model for centralized CRNs managed by a CBS. Our model consists of a set of schedulers with different features. All scheduling problems in this thesis concentrate on the data transmission of SUs to the CBS and make the frequency, time slot, and data rate assignments to the SUs. All schedulers ensure that PUs in the service area of the CBS are not disturbed, no collisions occur among the SUs, and reliable communication of the SUs with the CBS is maintained.

We formulate in Chapter 3 throughput and delay optimal scheduling problems for centralized CRNs under interference temperature constraints. We also propose heuristic algorithms called MFS and ProbFS for these problems. We evaluate the throughput and delay performance of the optimal as well as the suboptimal schedulers through simulations. We use optimization software CPLEX [5] to solve the optimization problems. While the performance results of the optimal schedulers serve as a baseline, suboptimal schedulers have significantly less computational complexity at the expense of reduced throughput and increased delay performance.

Although the computational simplicity of MFS and ProbFS makes them attractive, better performing suboptimal schedulers are needed. To this end, we propose GA based schedulers in Chapter 4. Our proposed GA based schedulers alleviate the computational complexity drawback of the throughput and delay optimal schedulers in Chapter 3. Specifically, we formulate GA-based algorithms and chromosome encoding methods as well as fitness function evaluation and comparison techniques for both throughput and delay optimal scheduling problems. We compare the performance of different selections, crossovers, encodings, and initial population creation techniques. Simulation results show that our GA based throughput maximizing and delay minimizing schedulers work best with uniformly random generation of each bit of the chromosomes during initial population creation, tournament selection, and uniform

crossover. Furthermore, our proposed GA-based suboptimal schedulers yield close to optimal performance with a reasonable number of iterations, while at the same time resulting in much better performance than MFS and ProbFS.

Since IT concept spurred a lot of debate since its inception due to its practical implementation complexity, FCC abandoned this concept. From Chapter 5 onwards, we distance ourselves from the IT debate and rely on conventional physical layer sensing mechanisms. We formulate in Chapter 5 throughput maximizing, max-min fair, weighted max-min fair, and proportionally fair scheduling problems, referred to as TMS, $MMFS$, weighted $MMFS$, and PFS problems. Our proposed scheduling scheme is a very general model jointly accomplishing numerous goals such as making the frequency, time slot, data rate, and power allocation to the SUs, which possibly have multiple antennas, in a heterogenous multi-channel and multi-user scenario. One of the distinctive features of our fair schedulers is that they take into account the throughput performance of the SUs in the recent past through a windowing mechanism and utilize this information to make the scheduling decision in the current scheduling period. This mechanism provides our schedulers with the ability to compensate for the possible temporary throughput losses of the SUs in the subsequent scheduling periods. Moreover, our fair schedulers also have the property of providing joint temporal and throughput fairness. Furthermore, we propose an efficient heuristic algorithm for (weighted) $MMFS$ and PFS problems. We assess the performance of the TMS scheduler with respect to various parameters such as the number of SUs, PUs, frequencies, and antennas. In addition, we make a comparative evaluation of all schedulers in terms of average total throughput and Jain's fairness index for varying window size and varying number of SUs. We observe that increasing the window size does not change the performance of our $MMFS$ and PFS schedulers; however, it increases the average total throughput in the weighted $MMFS$ at the expense of an increase in the deviation from target weights. We demonstrate through simulations that our proposed heuristic yields close performance to the solutions obtained from optimization softwares CPLEX and KNITRO.

In Chapter 6 we present a graph theoretic approach to TMS, $MMFS$, and PFS

problems formulated in Chapter 5. We propose a polynomial time algorithm for the TMS problem and discuss some of its special cases. We then prove that the $MMFS$ problem is NP-Hard in the strong sense and inapproximable within any constant factor better than 2 unless $P = NP$. We also present an approximation algorithm for this problem with approximation ratio depending on the maximum possible data rates of the secondary users. We evaluate the average case behavior of our approximation algorithm and demonstrate that it provides reasonable average case minimum throughput performance. Moreover, we discuss some of the special cases of the $MMFS$ problem and elaborate on their combinatorial properties. Then, we prove that the PFS problem is also NP-Hard in the strong sense. Furthermore, we propose more efficient integer programming formulations for all the three problems. Our graph theoretic study indicates that the $MMFS$ problem is computationally very hard. We demonstrate that even very special cases of this problem cannot be approximated within any constant factor better than 2 unless $P = NP$. Moreover, the theoretical computer science community has still been unable to find efficient approximation algorithms for these special cases. The computational complexity of this problem together with its practical importance in cognitive radio networks call for heuristic techniques that provide efficient suboptimal solutions. On the other hand, we prove that PFS problem is also NP-Hard in the strong sense.

In Chapter 7, we formulate a throughput satisfaction based scheduling problem, in which the objective is to maximize the number of SUs that are satisfied in terms of throughput. We prove that the problem is NP-Hard in the strong sense and cannot be approximated within any constant factor better than 2 unless $P = NP$. We also prove that this problem is at least as hard as the $MMFS$ problem. Heuristic algorithms for this problem as part of her MS thesis have been developed in [9].

We formulate in Chapter 8 a scheduling problem that considers different hardware delays that occur during switching to different frequency bands. We propose a polynomial time heuristic algorithm called S^{2}DASA to solve our formulated problem. The simulation results show that the throughput S^{2}DASA yields is very close to its upper bound. Moreover, S^{2}DASA is robust to changes in the hardware spectrum

switching delay. Furthermore, throughput savings it achieves increase as the number of frequencies in the CRN cell and the hardware switching delay for a unit frequency difference increases. Furthermore, throughput savings of our algorithm are significant even when there are a small number of SUs, and the savings remain significant as the number of SUs increases. Simulation results demonstrate that our idea of taking into account different hardware delays that occur during switching to different frequency bands is essential for CRNs since the assumption of constant switching delay can lead to low throughput performance.

9.2. Practical Implications of the Foundations of the Thesis

Proliferation and widespread use of CRNs in the future is inevitable due to the increased need for spectrum. When we attempt to enter CR/DSA business as a CBS operator, we need a set of scheduling algorithms that meet the unique needs and challenges of CRN environment. The set of scheduling algorithms we propose in this thesis can be readily used by the CBS operator in a dynamic and adaptive manner. Hence, our proposed algorithms can be regarded as part of a scheduling model for a CBS operator. Our findings indicate that each of our schedulers work better under different conditions. Scheduling model at the CBS can dynamically change the executed scheduler according to the network conditions. In particular, the scheduling model can implement the following:

- If the number of SUs is small, the model can execute TMS because the simulation results indicate that the fairness index of TMS is only slightly lower than the ones of fair schedulers. Therefore, it does not make sense to sacrifice from total throughput when there is a small number of SUs
- If the number of SUs is large, the CBS operator gives importance to fairness, there is no priority difference between SUs, and the spectrum and channel conditions between SUs is fairly uniform, then the model can execute MMFS. In order to deal with the computational difficulty associated with the MMFS scheduler, the scheduling model can follow the guidelines outlined in Section 6.2
- If the number of SUs is large, the CBS operator gives importance to fairness, there

is no priority difference between SUs, and the spectrum and channel conditions between SUs is highly heterogeneous, then the scheduling model can execute PFS. The reason for this decision is that if MMFS is executed, an SU with very bad channel conditions can drive the total throughput of all SUs to very low values. PFS scheduler provides a good tradeoff between maximizing total throughput and achieving fairness. In order to deal with the computational difficulty associated with the MMFS scheduler, the scheduling model can follow the guidelines outlined in Section 6.2

- If the number of SUs is large, the CBS operator gives importance to fairness, and there is priority difference between SUs, then the scheduling model can execute weighted MMFS scheduler in order to provide service differentiation capability to the SUs
- If the SUs execute different applications (real time/non-real time) with different minimum throughput requirements, then the scheduling model can execute MNSU scheduler in order to maximize the number of SUs that are satisfied in terms of throughput
- If the CBS cell uses a wide range of frequencies, then the scheduling model can execute the spectrum switching delay aware scheduler

9.3. Future Work

As a future work, an analytical study on the impact of mobility in centralized cognitive radio networks may be investigated. Whether mobility increases the capacity of cognitive radio networks and its relation to the spectrum occupancy behavior of PUs is an unexplored topic in the literature.

In a practical scenario, the number of SUs or frequencies in a CRN cell can change dynamically. SUs can roam between different CBS cells or a centralized entity like a spectrum broker can change the set of frequencies assigned to a CBS cell. Instead of calculating the new solution from scratch, reoptimization/rescheduling approaches may be utilized, where the previous solution is updated in response to the change in the number of SUs or frequencies. That is to say, reoptimization/rescheduling techniques

use prior knowledge about the previous solution and therefore they can lead to more efficient solutions.

We assumed in this thesis that physical layer information such as channel conditions and PU spectrum occupancy process is fed to the scheduling model. In practice, this information consists of estimated values. Sensing error occurs and the decisions regarding the PU activity may be erroneous. Likewise, information regarding SU and PU locations may also be erroneous. Instead of treating our proposed optimization schedulers as deterministic optimization problems, another approach might be to model them as stochastic optimization problems and hence provide a cross layer scheduling formulation.

REFERENCES

1. FCC, *03-222 Notice of proposed rule making and order*, Technical Report, 2003.

2. Mitola, J., *Cognitive Radio: An Integrated Agent Architecture for Software Defined Radio*, Ph.D. Thesis, Royal Institute of Technology (KTH), 2000.

3. Akyildiz, I., W. Lee, M. Vuran and S. Mohanty, "NeXt Generation/Dynamic Spectrum Access/Cognitive Radio Wireless Networks: A Survey", *Computer Networks*, Vol. 50, No. 13, pp. 2127–2159, 2006.

4. FCC, *03-289 Notice of Inquiry and Proposed Rulemaking*, Technical Report, 2003.

5. CPLEX, `http://www.ilog.com/products/cplex`, accessed at May 2012.

6. FCC, *07-78A1 Notice of Inquiry and Proposed Rulemaking*, Technical Report, 2007.

7. Quan, Z., S. Cui and A. Sayed, "Optimal Linear Cooperation for Spectrum Sensing in Cognitive Radio Networks", *IEEE Journal on Selected Topics in Signal Processing*, Vol. 2, No. 1, pp. 28–40, 2008.

8. Jiang, H., L. Lai, R. Fan and H. Poor, "Optimal Selection of Channel Sensing Order in Cognitive Radio", *IEEE Transactions on Wireless Communications*, Vol. 8, No. 1, pp. 297–307, 2009.

9. Eraslan, B., *Heuristic Algorithms for Scheduling in Centralized Cognitive Radio Networks*, M.S. Thesis, Computer Engineering, Bogazici University, 2011.

10. Knopp, R. and P. Humblet, "Information Capacity and Power Control in Single-Cell Multiuser Communications", *IEEE International Conference on Communications (ICC)*, 1995.

11. Viswanath, P., D. Tse and R. Laroia, "Opportunistic Beamforming Using Dumb Antennas", *IEEE Transactions on Information Theory*, Vol. 48, No. 6, pp. 1277–1294, 2002.

12. Andrews, M., K. Kumaran, K. Ramanan, A. Stolyar, P. Whiting and R. Vijayakumar, "Providing Quality of Service over a Shared Wireless Link", *IEEE Communications Magazine*, Vol. 39, No. 2, pp. 150–154, 2001.

13. Zhou, C. and G. Wunder, "A Novel Low Delay Scheduling Algorithm for OFDM Broadcast Channel", *IEEE Global Telecommunications Conference (GLOBECOM)*, pp. 3709–3713, 2007.

14. Tang, J., S. Misra and G. Xue, "Joint Spectrum Allocation and Scheduling for Fair Spectrum Sharing in Cognitive Radio Wireless Networks", *Computer Networks*, Vol. 52, No. 11, pp. 2148–2158, 2008.

15. Lee, W. and I. Akyildiz, "A Spectrum Decision Framework for Cognitive Radio Networks", *IEEE Transactions on Mobile Computing*, Vol. 10, No. 2, pp. 161–174, 2010.

16. Lee, W. and I. Akyildiz, "Joint Spectrum and Power Allocation for Inter-cell Spectrum Sharing in Cognitive Radio Networks", *IEEE International Symposium on Dynamic Spectrum Acceess Networks (DySPAN)*, 2008.

17. Thoppian, M., S. Venkatesan, R. Prakash and R. Chandrasekaran, "MAC-Layer Scheduling in Cognitive Radio Based Multi-Hop Wireless Networks", *IEEE International Symposium on on World of Wireless, Mobile and Multimedia Networks*, pp. 191–202, 2006.

18. Ma, M. and D. Tsang, "Impact of Channel Heterogeneity on Spectrum Sharing in Cognitive Radio Networks", *IEEE International Conference on Communications (ICC)*, pp. 2377–2382, 2008.

19. Li, J., B. Xu, Z. Xu, S. Li and Y. Liu, "Adaptive Packet Scheduling Algorithm for Cognitive Radio System", *International Conference on Communication Technology (ICCT)*, pp. 1–5, 2006.

20. Hamdi, K., W. Zhang and K. Ben Letaief, "Uplink Scheduling with QoS Provisioning for Cognitive Radio Systems", *IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 2592–2596, 2007.

21. Wang, W. and X. Liu, "List-Coloring Based Channel Allocation for Open-Spectrum Wireless Networks", *IEEE Vehicular Technology Conference (VTC)*, Vol. 1, 2005.

22. Urgaonkar, R. and M. Neely, "Opportunistic Scheduling with Reliability Guarantees in Cognitive Radio Networks", *IEEE International Conference on Computer Communications (INFOCOM)*, pp. 1301–1309, 2008.

23. Clancy, T., "Achievable Capacity under the Interference Temperature Model", *IEEE International Conference on Computer Communications (INFOCOM)*, 2007.

24. Xing, Y., C. Mathur, M. Haleem, R. Chandramouli and K. Subbalakshmi, "Dynamic Spectrum Access with QoS and Interference Temperature Constraints", *IEEE Transactions on Mobile Computing*, Vol. 6, No. 4, pp. 423–433, 2007.

25. Wang, W., T. Peng and W. Wang, "Optimal Power Control under Interference Temperature Constraints in Cognitive Radio Network", *IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 116–120, 2007.

26. Bater, J., H. Tan, K. Brown and L. Doyle, "Modelling Interference Temperature Constraints for Spectrum Access in Cognitive Radio Networks", *IEEE International Conference on Communications (ICC)*, pp. 6493–6498, 2007.

27. Habib, I., M. Sherif, M. Naghshineh and P. Kermani, "An Adaptive Quality of

Service Channel Borrowing Algorithm for Cellular Networks", *Wiley's International Journal of Communication Systems (IJCS)*, Vol. 16, No. 8, pp. 759–777, 2003.

28. Sandalidis, H., P. Stavroulakis and J. Rodriguez-Tellez, "Comparison of Two Novel Heuristic Dynamic Channel Allocation Techniques in Cellular Systems", *Wiley's International Journal of Communication Systems (IJCS)*, Vol. 11, No. 6, pp. 379–386, 1998.

29. Rondeau, T., B. Le, C. Rieser and C. Bostian, "Cognitive Radios with Genetic Algorithms: Intelligent Control of Software Defined Radios", *SDR Forum Technical Conference*, 2006.

30. Kim, J., S. Sohn, N. Han, G. Zheng, Y. Kim and J. Lee, "Cognitive Radio Software Testbed using Dual Optimization in Genetic Algorithm", *International Conference on Cognitive Radio Oriented Wireless Networks and Communications (CROWNCOM)*, pp. 1–6, 2008.

31. Friend, D., M. Elnainay, Y. Shi and A. Mackenzie, "Architecture and Performance of an Island Genetic Algorithm-based Cognitive Networks", *IEEE Consumer Communications and Networking Conference (CCNC)*, pp. 993–997, 2008.

32. Newman, T., R. Rajbanshi, A. Wyglinski, J. Evans and G. Minden, "Population Adaptation for Genetic Algorithm-based Cognitive Radios", *Mobile Networks and Applications*, Vol. 13, No. 5, pp. 442–451, 2008.

33. Liu, X., E. Chong and N. Shroff, "Opportunistic Transmission Scheduling with Resource-Sharing Constraints in Wireless Networks", *IEEE Journal on Selected Areas in Communications*, Vol. 19, No. 10, pp. 2053–2064, 2001.

34. Liu, X., E. Chong and N. Shroff, "A Framework for Opportunistic Scheduling in Wireless Networks", *Computer Networks*, Vol. 41, No. 4, pp. 451–474, 2003.

35. Huang, X. and B. Bensaou, "On Max-Min Fairness and Scheduling in Wireless Ad-hoc networks: Analytical Framework and Implementation", *ACM International Symposium on Mobile Ad Hoc Networking & Computing*, pp. 221–231, 2001.

36. Liu, Y. and E. Knightly, "Opportunistic Fair Scheduling over Multiple Wireless Channels", *IEEE International Conference on Computer Communications (INFOCOM)*, Vol. 2, pp. 1106–1115, 2003.

37. Nandagopal, T., T. Kim, X. Gao and V. Bharghavan, "Achieving MAC Layer Fairness in Wireless Packet Networks", *ACM International Conference on Mobile Computing and Networking (MOBICOM)*, pp. 87–98, 2000.

38. Nguyen, M. and H. Lee, "Effective Scheduling in Infrastructure-based Cognitive Radio Network", *IEEE Transactions on Mobile Computing*, Vol. 10, No. 6, pp. 853–867, 2011.

39. Shi, Y., Y. Hou, S. Kompella and H. Sherali, "Maximizing Capacity in Multihop Cognitive Radio Networks under the SINR Model", *IEEE Transactions on Mobile Computing*, Vol. 10, No. 7, pp. 954–967, 2010.

40. Peng, C., H. Zheng and B. Zhao, "Utilization and Fairness in Spectrum Assignment for Opportunistic Spectrum Access", *Mobile Networks and Applications*, Vol. 11, No. 4, pp. 555–576, 2006.

41. Urgaonkar, R. and M. Neely, "Opportunistic Scheduling with Reliability Guarantees in Cognitive Radio Networks", *IEEE Transactions on Mobile Computing*, Vol. 8, No. 6, pp. 766–777, 2009.

42. Loguinov, D., J. Casas and X. Wang, "Graph-Theoretic Analysis of Structured Peer-to-Peer Systems: Routing Distances and Fault Resilience", *IEEE/ACM Transactions on Networking*, Vol. 13, No. 5, pp. 1107–1120, 2005.

43. Shalom, M. and S. Zaks, "A 10/7+ ε Approximation for Minimizing the Number

of ADMs in SONET Rings", *IEEE/ACM Transactions on Networking*, Vol. 15, No. 6, pp. 1593–1602, 2007.

44. Blough, D., G. Resta and P. Santi, "Approximation Algorithms for Wireless Link Scheduling with SINR-based Interference", *IEEE/ACM Transactions on Networking*, Vol. 18, No. 6, pp. 1701–1712, 2010.

45. Djukic, P. and S. Valaee, "Delay Aware Link Scheduling for Multi-hop TDMA Wireless Networks", *IEEE/ACM Transactions on Networking*, Vol. 17, No. 3, pp. 870–883, 2009.

46. Wang, W. and X. Liu, "List-Coloring Based Channel Allocation for Open-Spectrum Wireless Networks", *IEEE Vehicular Technology Conference (VTC)*, Vol. 62, 2005.

47. Khozeimeh, F. and S. Haykin, "Dynamic Spectrum Management for Cognitive Radio: An Overview", *Wireless Communications and Mobile Computing*, Vol. 9, No. 11, pp. 1447–1459, 2009.

48. Yuan, Y., P. Bahl, R. Chandra, T. Moscibroda and Y. Wu, "Allocating Dynamic Time-Spectrum Blocks in Cognitive Radio Networks", *ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pp. 130–139, 2007.

49. Santos, R., F. Lima, W. Freitas and F. Cavalcanti, "QoS-based Radio Resource Allocation and Scheduling with Different User Data Rate Requirements for OFDMA Systems", *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2007.

50. Rodrigues, E. and F. Casadevall, "Adaptive Radio Resource Allocation Framework for Multi-User OFDM", *IEEE Vehicular Technology Conference (VTC)*, 2009.

51. Pal, S., S. Kundu, M. Chatterjee and S. Das, "Combinatorial Reverse Auction

Based Scheduling in Multi-Rate Wireless Systems", *IEEE Transactions on Computers*, Vol. 56, No. 10, pp. 1329–1341, 2007.

52. Aristomenopoulos, G., T. Kastrinogiannis, V. Kaldanis, G. Karantonis and S. Papavassiliou, "A Novel Framework for Dynamic Utility-based QoE Provisioning in Wireless Networks", *IEEE Global Communications Conference (GLOBECOM)*, 2010.

53. Kastrinogiannis, T. and S. Papavassiliou, "Utility Based Short-Term Throughput Driven Scheduling Approach for Efficient Resource Allocation in CDMA Wireless Networks", *Wireless Personal Communications*, Vol. 52, No. 3, pp. 517–535, 2010.

54. Raman, C., J. Singh, R. Yates and N. Mandayam, "Scheduling Variable Rate Links–Centralized and Decentralized Approaches", *Cognitive Wireless Networks: Concepts, Methodologies and Visions Inspiring the Age of Enlightenment of Wireless Communications*, p. 285, Springer, 2007.

55. Zhang, Z., Y. He and E. Chong, "Opportunistic Scheduling for OFDM Systems with Fairness Constraints", *EURASIP Journal on Wireless Communications and Networking*, Vol. 2008, pp. 1–12, 2008.

56. Khalil, K., M. Karaca, O. Erçetin and E. Ekici, "Optimal Scheduling in Cooperate-to-Join Cognitive Radio Networks", *IEEE International Conference on Computer Communications (INFOCOM)*, 2011.

57. Kim, H. and K. Shin, "Efficient Discovery of Spectrum Opportunities with MAC-Layer Sensing in Cognitive Radio Networks", *IEEE Transactions on Mobile Computing*, Vol. 7, No. 5, pp. 533–545, 2007.

58. Cheng, G., W. Liu, Y. Li and W. Cheng, "Spectrum Aware On-Demand Routing in Cognitive Radio Networks", *IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, 2007.

59. Cheng, G., W. Liu, Y. Li and W. Cheng, "Joint On-Demand Routing and Spectrum Assignment in Cognitive Radio Networks", *IEEE International Conference on Communications (ICC)*, 2007.

60. Filippini, I., E. Ekici and M. Cesana, "Minimum Maintenance Cost Routing in Cognitive Radio Networks", *IEEE International Conference on Mobile Ad hoc and Sensor Systems (MASS)*, 2009.

61. Chowdhury, K. and M. Felice, "SEARCH: A Routing Protocol for Mobile Cognitive Radio Ad-hoc Networks", *Computer Communications*, Vol. 32, No. 18, pp. 1983–1997, 2009.

62. Chen, J., H. Li, J. Wu and R. Zhang, "STARP: A Novel Routing Protocol for Multi-hop Dynamic Spectrum Access Networks", *ACM Workshop on Mobile Internet Through Cellular Networks*, 2009.

63. Krishnamurthy, S., M. Thoppian, S. Venkatesan and R. Prakash, "Control Channel Based MAC-Layer Configuration, Routing and Situation Awareness for Cognitive Radio Networks", *IEEE Military Communications Conference (MILCOM)*, 2005.

64. Mitran, P., "Interference Reduction in Cognitive Networks via Scheduling", *IEEE Transactions on Wireless Communications*, Vol. 8, No. 7, pp. 3430–3434, 2009.

65. Zhao, Z., Z. Peng, S. Zheng and J. Shang, "Cognitive Radio Spectrum Allocation using Evolutionary Algorithms", *IEEE Transactions on Wireless Communications*, Vol. 8, No. 9, pp. 4421–4425, 2009.

66. Gözüpek, D. and F. Alagöz, "Throughput and Delay Optimal Scheduling in Cognitive Radio Networks under Interference Temperature Constraints", *Journal of Communications and Networks (JCN)*, Vol. 11, No. 2, pp. 147–155, 2009.

67. Clancy, T., "Formalizing the Interference Temperature Model", *Wireless Com-*

munications and Mobile Computing, Vol. 7, No. 9, pp. 1077–1086, 2007.

68. IEEE, *802.22 Standard*, `http://www.ieee802.org/22/`, accessed at May 2012.

69. Cover, T., J. Thomas, J. Wiley and W. InterScience, *Elements of Information Theory*, Wiley-Interscience, New York, 2006.

70. Land, A. and A. Doig, "An Automatic Method for Solving Discrete Programming Problems", *Econometrica*, Vol. 28, No. 3, pp. 497–520, 1960.

71. Sherali, H. and D. Myers, "The Design of Branch and Bound Algorithms for a Class of Nonlinear Integer Programs", *Annals of Operations Research*, Vol. 5, No. 1-4, pp. 463–484, 1986.

72. OPNET, `http://www.opnet.com`, accessed at May 2012.

73. Clancy, T., "Interference Temperature Multiple Access", , 2006.

74. Gözüpek, D. and F. Alagöz, "Genetic Algorithm-based Scheduling in Cognitive Radio Networks under Interference Temperature Constraints", *Wiley's International Journal of Communication Systems (IJCS)*, Vol. 24, No. 2, 2011.

75. Holland, J., *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, Michigan, 1975.

76. Haupt, R. and S. Haupt, *Practical Genetic Algorithms*, Second Edition, John Wiley & Sons Inc., Hoboken, NJ, 2004.

77. Fonseca, C. and P. Fleming, "Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms. I. A Unified Formulation", *IEEE Transactions on Systems, Man and Cybernetics, Part A*, Vol. 28, No. 1, pp. 26–37, 1998.

78. Montgomery, D., *Design and Analysis of Experiments*, John Wiley & Sons, New

York, 2006.

79. Gözüpek, D. and F. Alagöz, "An Interference Aware Throughput Maximizing Scheduler for Centralized Cognitive Radio Networks", *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pp. 1–6, 2010.

80. FCC, *08-260 Notice of Inquiry and Proposed Rulemaking*, Technical Report, 2008.

81. Wang, L. and A. Chen, "Effects of Location Awareness on Concurrent Transmissions for Cognitive Ad hoc Networks Overlaying Infrastructure-based Systems", *IEEE Transactions on Mobile Computing*, Vol. 8, No. 5, pp. 577–589, 2009.

82. Wang, H., J. Ren and T. Li, "Resource Allocation with Load Balancing for Cogntive Radio Networks", *IEEE Global Communications Conference (GLOBECOM)*, 2010.

83. Digham, F., "Joint Power and Channel Allocation for Cognitive Radios", *IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 882–887, 2008.

84. Rajan, D., A. Sabharwal and B. Aazhang, "Delay-Bounded Packet Scheduling of Bursty Traffic over Wireless Channels", *IEEE Transactions on Information Theory*, Vol. 50, No. 1, pp. 125–144, 2004.

85. Li, L., M. Pal and Y. Yang, "Proportional Fairness in Multi-Rate Wireless LANs", *IEEE International Conference on Computer Communications (INFOCOM)*, pp. 1004–1012, 2008.

86. KNITRO, `http://www.ziena.com/knitro.htm`, accessed at May 2012.

87. Montgomery, D., *Design and Analysis of Experiments*, John Wiley & Sons, 2006.

88. Montgomery, D. and G. Runger, *Applied Statistics and Probability for Engineers*,

John Wiley & Sons, New York, 2007.

89. Jain, R., A. Durresi and G. Babic, "Throughput Fairness Index: An Explanation", *ATM Forum Contribution*, Vol. 45, 1999.

90. Lovász, L. and M. Plummer, *Matching Theory*, AMS Chelsea Publishing, Providence, R.I., 2009.

91. Pulleyblank, R., *Faces of Matching Polyhedra*, Ph.D. Thesis, Univ. of Waterloo, Dept. Combinatorics and Optimization, 1973.

92. Schrijver, A., *Combinatorial Optimization: Polyhedra and Efficiency*, Springer, New York, 2003.

93. Bansal, N. and M. Sviridenko, "The Santa Claus Problem", *ACM Symposium on Theory of Computing (STOC)*, p. 40, 2006.

94. König, D., "Graphok Es Alkalmazasuk A Determinansok Es A Halmazok Elmeletere [Hungarian]", *Mathematikai es Termeszettudomanyi Ertesito*, Vol. 34, pp. 104–119, 1916.

95. Hall, P., "On Representatives of Subsets", *Journal of the London Mathematical Society*, Vol. 10, pp. 26–30, 1934.

96. Tutte, W., "The Factorization of Linear Graphs", *Journal of the London Mathematical Society*, Vol. 22, pp. 107–111, 1947.

97. Chekuri, C. and S. Khanna, "A Polynomial Time Approximation Scheme for the Multiple Knapsack Problem", *SIAM Journal on Computing*, Vol. 35, No. 3, pp. 713–728, 2005.

98. Martello, S. and P. Toth, *Knapsack Problems Algorithms and Computer Implementations*, John Wiley& Sons, Chichester, New York, 1990.

99. Kershenbaum, A., *Telecommunications Network Design Algorithms*, McGraw-Hill, New York, 1993.

100. Lawler, E., *Combinatorial Optimization: Networks and Matroids*, Dover Publications, Mineola, N.Y., 2001.

101. Bezáková, I. and V. Dani, "Nobody Left Behind: Fair Allocation of Indivisible Goods", *ACM SIGecom Exchanges*, Vol. 5, 2005.

102. Lenstra, J., D. Shmoys and E. Tardos, "Approximation Algorithms for Scheduling Unrelated Parallel Machines", *Mathematical Programming*, Vol. 46, No. 1, pp. 259–271, 1990.

103. Bateni, M., M. Charikar and V. Guruswami, "Max-Min Allocation via Degree Lower-Bounded Arborescences", *ACM Symposium on Theory of Computing (STOC)*, pp. 543–552, 2009.

104. Woeginger, G., "A Polynomial-Time Approximation Scheme for Maximizing the Minimum Machine Completion Time", *Operations Research Letters*, Vol. 20, No. 4, pp. 149–154, 1997.

105. Asadpour, A. and A. Saberi, "An Approximation Algorithm for Max-Min Fair Allocation of Indivisible Goods", *ACM Symposium on Theory of Computing (STOC)*, pp. 114–121, 2007.

106. Asadpour, A., U. Feige and A. Saberi, "Santa Claus Meets Hypergraph Matchings", *Approximation, Randomization and Combinatorial Optimization Algorithms and Techniques (APPROX)*, pp. 10–20, 2008.

107. Asadpour, A., U. Feige and A. Saberi, "Santa Claus meets hypergraph matchings", *to appear in ACM Transactions on Algorithms*, `http://www.stanford.edu/~asadpour/papers/santa.pdf`, 2009.

108. Haeupler, B., B. Saha and A. Srinivasan, "New Constructive Aspects of the Lovasz Local Lemma", *IEEE Symposium on Foundations of Computer Science (FOCS), Arxiv preprint arXiv:1001.1231*, 2010.

109. Burkard, R., M. Dell'Amico and S. Martello, *Assignment Problems*, Society for Industrial Mathematics, Philadelphia, 2009.

110. Hou, J., J. Yang and S. Papavassiliou, "Integration of Pricing with Call Admission Control to Meet QoS Requirements in Cellular Networks", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 13, No. 9, pp. 898–910, 2002.

111. Assmann, S., D. Johnson, D. Kleitman and J. Leung, "On a Dual Version of the One-Dimensional Bin Packing Problem", *Journal of Algorithms*, Vol. 5, No. 4, pp. 502–525, 1984.

112. Jansen, K. and R. Solis-Oba, "An Asymptotic Fully Polynomial Time Approximation Scheme for Bin Covering", *Theoretical Computer Science (TCS)*, Vol. 306, No. 1-3, pp. 543–551, 2003.

113. Maroug, S., "Frequency-Switching Speed and Post-Tuning Drift Measurement of Fast-Switching Microwave-Frequency Synthesisers", *IET Science, Measurement & Technology*, Vol. 1, No. 2, pp. 82–86, 2007.

114. Ma, H., L. Zheng and X. Ma, "Spectrum Aware Routing for Multi-hop Cognitive Radio Networks with a Single Transceiver", *International Conference on Cognitive Radio Oriented Wireless Networks (CROWNCOM)*, 2008.

115. "TCI 7234 Wideband SHF Tuner Data Specification", `http://www.tcibr.com/ufiles/Library/7234_webp.pdf`, accessed at May 2012.

116. "TCI 715 Data Specification", `http://www.tcibr.com/ufiles/Library/715_webp.pdf`, accessed at May 2012.

117. Cesana, M., F. Cuomo and E. Ekici, "Routing in Cognitive Radio Networks: Challenges and Solutions", *Ad Hoc Networks*, Vol. 9, No. 3, pp. 228–248, 2011.

118. "TCI 735 Data Specification", `http://www.tcibr.com/ufiles/Library/735_webp.pdf`, accessed at May 2012.

119. "TCI 745 Data Specification", `http://www.tcibr.com/ufiles/Library/745_webp.pdf`, accessed at May 2012.

120. Wang, B. and K. Liu, "Advances in Cognitive Radio Networks: A Survey", *IEEE Journal of Selected Topics in Signal Processing*, Vol. 5, No. 1, pp. 5–23, 2011.

121. Ileri, O., D. Samardzija and N. Mandayam, "Demand Responsive Pricing and Competitive Spectrum Allocation via a Spectrum Server", *IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, pp. 194–202, 2005.

122. Kamal, H., M. Coupechoux and P. Godlewski, "Inter-Operator Spectrum Sharing for Cellular Networks using Game Theory", *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pp. 425–429, 2009.

123. Yamada, T., D. Burgkhardt, I. Cosovic and F. Jondral, "Resource Distribution Approaches in Spectrum Sharing Systems", *EURASIP Journal on Wireless Communications and Networking*, Vol. 2008, p. 8, 2008.

124. Koch, P. and R. Prasad, "The Universal Handset", *IEEE Spectrum*, Vol. 46, No. 4, pp. 36–41, 2009.

125. Yun, M., Y. Zhou, A. Arora and H. Choi, "Channel-Assignment and Scheduling in Wireless Mesh Networks Considering Switching Overhead", *IEEE International Conference on Communications (ICC)*, pp. 1–6, 2009.

126. Hemmecke, R., M. Köppe, J. Lee and R. Weismantel, "Nonlinear Integer Programming", *50 Years of Integer Programming 1958-2008*, pp. 561–618, 2010.

127. Schrijver, A., *Theory of Linear and Integer Programming*, Wiley-Interscience, New York, 1999.