On the Optimal Set of Channels to Sense in Cognitive Radio Networks

by

Afshin Arefi

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Communications

Department of Electrical and Computer Engineering

University of Alberta

Abstract

We study a channel selection problem in cognitive radio with imperfect spectrum sensing. In this problem, a secondary (unlicensed) user must select a subset of M channels out of N channels to sense. The user then accesses up to $K \leq M$ channels that were sensed free. The objective is to maximize the user's expected throughput. This work is motivated by the surprising fact that an intuitive solution proposed in the literature is only optimal for $K = M$. We perform a worst-case performance analysis of the intuitive solution and show that its performance can be a factor of $\Omega(N)$ (that is a constant factor of N) worse than that of the optimal solution. We propose polynomial-time optimal solutions for cases where $K = 1$, $M = \mathcal{O}(1)$, or $N - M = \mathcal{O}(1)$. For the general case, we propose a sub-optimal polynomial-time algorithm, as well as a polynomial-time algorithm to calculate an upper bound on the maximum throughput achievable.

"Those who intend on becoming great should love neither themselves nor their own things, but only what is just, whether it happens to be done by themselves or others."

- - Plato

Acknowledgements

I want to use this opportunity to thank everyone who has helped me in preparation of this work. First of all, I should thank my supervisor, Dr. Majid Khabbazian, who have been my greatest help during the past two years in every aspect. Second, I want to thank the committee members, Dr. Masoud Ardakani and Dr. Hai Jiang, for dedicating their time and energy, reading my thesis and providing their invaluable ideas. Then, I want to also thank every teacher, professor or friend who has helped me learn something new during all my years of education, fulfilling my thirst for knowledge. Last, but not least, I want to thank everyone in history who has helped build the foundations of academia and the scientific world whose discoveries has enabled me to become another brick in this great structure today. To quote Sir. Isaac Newton, one of my greatest inspirations in the academic world, I must say

"If I have seen further, it is by standing on the shoulders of giants."

Contents

List of Tables

List of Figures

List of Abbreviations

List of commonly used abbreviations

PU	Primary User
SU	Secondary User
CRN	Cognitive Radio Network
C.D.P.	Common Detection Probability

List of Symbols

List of Symbols

N	Number of channels
M	Number of sensed channels
K	Number of channels to transmit on
T	Time slot length
τ	Sensing portion of time slot
$\mathcal{C}$	Channel
$B(\mathcal{C})$	Bandwidth of channel $\mathcal{C}$
$\theta(\mathcal{C})$	Probability of channel $\mathcal{C}$ being free
$\alpha(\mathcal{C})$	Probability of false alarm for channel $\mathcal{C}$
$\mu(\mathcal{C})$	Probability of miss detection for channel $\mathcal{C}$
$\phi(\mathcal{C})$	Probability of channel $\mathcal{C}$ being sensed free
$\mathcal{R}^c(\mathcal{C})$	Conditional reward
$\mathcal{R}^b(\mathcal{C})$	Blind reward
X_i	Random variable for transmitted bits on a channel
$\mathcal{X}_{(i)}$	i_{th} *order statistic* of $X_1, ..., X_N$
$\Psi(S)$	Number of bits transmitted using set S
ζ	Optimal to intuitive solution ratio for $K = 1$
$\mathbf{G}_k^S$	Maximum gain achieved using S and accesing k channels
$\mathbf{G}_{n,m,k}$	Maximum gain over n, m, k
$\mathbf{H}_{n,m,k}$	Maximum gain over $n, m, (k, k-1)$ balanced using $\phi(\mathcal{C}_{n+1})$
$\mathcal{C}^*$	Channel with the largest blind reward
$\mathbf{G}_{n,m,\_}$	Maximum gain over $n, m, 1$
$\hat{\mathbf{G}}_{n,m,k}$	Maximum gain over $n, m, 1$
$\mathbf{G}_{\_,m,k}$	Maximum gain over m specific channels
Γ	Maximum ratio of bandwidth in the system
$\mathcal{O}, \Omega, \Theta$	Asymptotic notations

Chapter 1

Introduction

1.1 Cognitive Radio Networks

In 1999, Joseph Mitola III and Gerald Q. Maguire, Jr. proposed a new technology in wireless networks called Cognitive Radio [1]. Their original thought was using such technology for PDAs (personal digital assistants) and hand-held devices which was starting to emerge at the time. Cognitive Radio Networks are different from traditional radio networks as there are no fixed radio frequency or set of channels for transmission. Instead, a device taking advantage of Cognitive Radio Network technology will intelligently decide on what channel to use based on the available information. Cognitive Radio Networks are categorized into two main types:

- **Full Cognitive Radio (Mitola radio)**: The devices takes advantage of every observable parameter to choose the best radio frequency for transmission. A full cognitive radio system is mostly research based and probably will not be implemented in the near future.

- **Spectrum-Sensing Cognitive Radio**: The devices only use the availability of radio frequencies to make the best decision on using such channel for radio transmission. Spectrum-Sensing Cognitive Radio systems

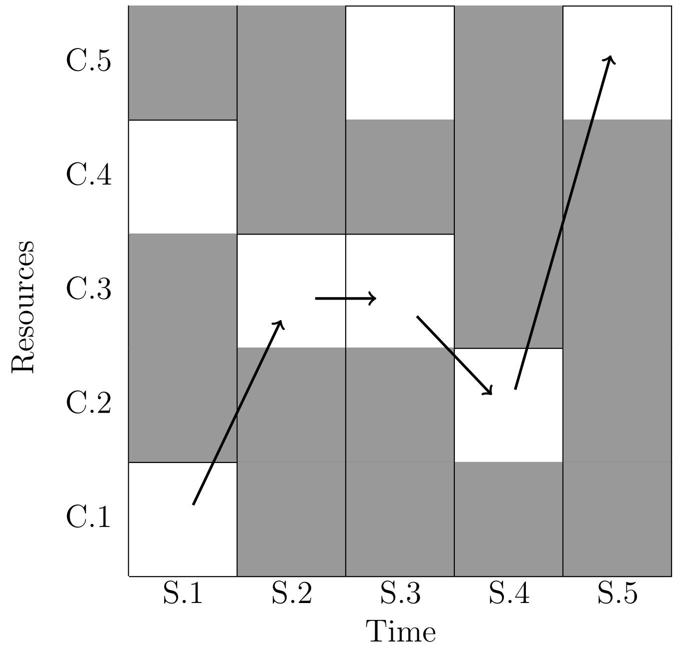

Figure 1.1: A Secondary User (Unlicensed User) will transmit on a free channel without disrupting its Primary User (Licensed User). As soon as the primary user starts using such channel, the secondary user will stop transmitting on that channel and move to a new free channel. In the above figure, grey cells show primary user activity on the channel and the arrows show secondary user's choice of channel for transmission.

are mostly used to improve the efficiency of usage in existing bands.

In cognitive radio networks, the available radio channels are either unlicensed (e.g. Wi-Fi Frequencies like 2.4 GHz and 5 GHz) or licensed under some other user. The license holders of these radio channels are called Primary Users in cognitive radio networks. On the other hand, the user taking advantage of the cognitive radio network technology is called a Secondary User. In different cognitive radio systems, there can either be a single secondary user or multiple secondary users, either transmitting independently or in the same network with each other.

Cognitive radio networks can utilize multiple radio channels by taking advantage of their free times. In a cognitive radio system, the secondary user will start transmitting on a free channel, but as soon as it senses activity on such channel, it can move to another free channel as shown in Figure 1.1.

There are multiple challenges when it comes to using cognitive radio networks. First of all, the radio channel allocations and the regulatory bodies are based on fixed channel transmission and do not let users use the reserved channels licenced for other purposes, even if the secondary user creates virtually no disruption for the primary user. However, in the recent decade, the regulatory bodies have started to discuss the possibility of regulating the cognitive radio networks and creating a possibility for secondary users to take advantage of the licensed bands and channels without disrupting their primary users' transmission.

Another challenge in using cognitive radio networks is the dependability of sensing and the importance of not disrupting the licenced user. As we will explain more in the net section, Sensing a channel for activity does not give us fully dependable results. A free sensed channel can be actually busy, or a busy sensed channel can be free. Based on the importance of the licenced user's activity and the probability of error, cognitive radios might not be feasible in some situations.

Cognitive radio networks have many applications. One of the main applications is utilization of many licenced spectrum which are not assigned efficiently and are wasted. Since these channels are licenced no one but their assigned user have been able to use them for transmission. But in most cases, the primary users are not taking full advantage of the band. Cognitive radio networks create the opportunity for unlicensed users to take advantage of such channels without disrupting the primary user.

Another important application is the ability to set-up a wireless network in any place without any knowledge of the spectrum usage, assignments or doing field tests. Such systems can easily adopt to any location by choosing the less frequently used channels automatically and creating an stable transmission medium for any purpose.

Cognitive radios have not yet been fully commercialized in large scales. But some devices are available which take advantage of the technology to deliver higher bandwidths to the users.

1.1.1 Channel Sensing

Wireless systems need to sense the channel they are going to use for transmission before using it to make sure there is no activity on that channel or frequency. Most of the time the sensing process is not perfect and the result might not be reliable. Two main errors might happen when sensing,

1. **False Alarm**: False alarm happens when a channel is free but due to, for example, noise the user decides that the frequency is not available.

2. **Miss Detection**: Miss detection happens when the radio frequency is being used but the user cannot sense the activity due to, for example, distance and decides that the frequency is available for transmission.

1.2 A Nontechnical Abstraction

The main problem discussed in this thesis can be abstracted using a simple game. This abstraction serves two main purposes:

1. It presents a nontechnical view of the problem, making it easy for anyone with minimum background to understand the concept.

2. It is a more abstracted and general form of the problem, giving it broader applications in computer science.

Imagine a game where you are presented with N closed boxes. Each box contains some money and is labelled with the amount of money inside. You are offered to take the money from K of those boxes. The problem is that, unfortunately, some of the boxes have been emptied before. You do not know the empty boxes, but you have been able to calculate a probability of each box being empty. Since this is not a favourable situation, we are allowed to open and take a look at inside of M ($K \leq M \leq N$) boxes. Other boxes (those that are not opened) are discarded. From the boxes that are opened, we can, then, choose K boxes and collect the money from them.

If you know which M boxes to look inside, the "money-collection" step will be quite easy. After looking inside the boxes, you will choose the K with the highest amount of money. The important decision you have to make is which M boxes to look inside. The case where $N = M$ is trivial. Also, the case where $M = K$ has been proved to be intuitive: in this case, one must choose the M boxes with the highest expected reward ($Prob_{full} \times \mathit{Reward}$). The important remaining question will be which boxes to look inside when $N < M < K$.

Let's see the difficulty of the choice using a simple example. You are presented with $N = 3$ boxes (B_1, B_2, and B_3). The boxes are labelled \$2, \$11, and \$100, respectively. Based on your experience and previous observations, you know that the first box is full ($P_1 = 100\%$), the second box has a 10% probability of being full ($P_2 = 10\%$), and the last box has a 1% probability of being full ($P_3 = 1\%$). You can see the inside of two boxes of your choice, and take money from one of those two boxes. There are three possible options/choices on what two boxes to open: $\{(B_1, B_2), (B_1, B_3), (B_2, B_3)\}$. For each choice, the expected money that will be collected will be:

$$\mathit{Reward}_{(B_1,B_2)} = (0.10 \times 11) + (1 - 0.10) \times 1.00 \times 2 = \$2.80$$

$$\mathit{Reward}_{(B_1,B_3)} = (0.01 \times 100) + (1 - 0.01) \times 1.00 \times 2 = \$2.98$$

$$\mathit{Reward}_{(B_2,B_3)} = (0.01 \times 100) + (1 - 0.01) \times 0.10 \times 11 = \$2.089$$

The intuitive solution (which is to choose the boxes with the highest expected reward) gives (B_1, B_2) which will result in $\mathit{Reward}_{(B_1,B_2)} = \$2.80$. However, choosing (B_1, B_3) will give the expected reward of $\mathit{Reward}_{(B_1,B_3)} = \$2.98$.

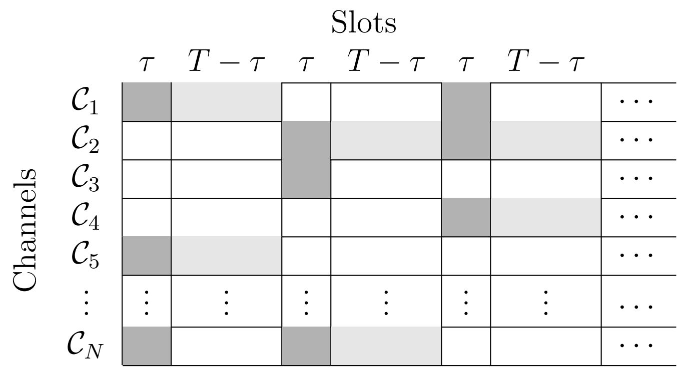

Figure 1.2: Dark grey areas show the $M = 3$ channels chosen to be sensed, and light grey areas show the $K = 2$ channels used for transmission. A transmission only happens on channels that were sensed free.

1.3 System Model and Definitions

We use a slightly-generalized model as the one used in [2]. Similar models have also been used in [3, 4, 5, 6].

In this model, there is a single secondary user (SU), and N independent primary channels, $\mathcal{C}_1, \ldots, \mathcal{C}_N$. The time is divided into slots of length T. As shown in Figure 1.2, each time slot is further divided into two phases, a *sensing phase* of length τ and a *transmission phase* of length $T - \tau$. In the sensing phase of every time slot, the secondary user senses M, $M \leq N$, channels. Then, in the transmission phase, it uses K, $K \leq M$, channels from the ones sensed free, to transmit data. If the number of channels sensed free is less than K, then secondary user uses all the sensed free channels to transmit data. We assume that the secondary user can transmit $B\left(\mathcal{C}_i\right)$ bits on a sensed free channel $\mathcal{C}_i$ in one slot. This generalizes the model used in [2], as in that model $B\left(\mathcal{C}_i\right) = B\left(\mathcal{C}_j\right)$ for every $1 \leq i, j \leq N$, while in our model $B\left(\mathcal{C}_i\right), 1 \leq i \leq N$ can be different numbers.

We use $\theta\left(\mathcal{C}_i\right)$, $1 \leq i \leq N$, to denote the probability that channel $\mathcal{C}_i$ is free in a slot. Miss-detection probability of $\mathcal{C}_i$, defined as the probability of $\mathcal{C}_i$ sensed free when it is busy, is denoted by $\mu\left(\mathcal{C}_i\right)$. Also, false alarm probability of $\mathcal{C}_i$, defined as the probability of $\mathcal{C}_i$ sensed busy when it is free, is denoted by $\alpha\left(\mathcal{C}_i\right)$.

The probability of channel $\mathcal{C}_i$ being sensed free, $\phi(\mathcal{C}_i)$, can then be calculated as

$$\phi(\mathcal{C}_i) = \theta(\mathcal{C}_i)(1 - \alpha(\mathcal{C}_i)) + (1 - \theta(\mathcal{C}_i))\mu(\mathcal{C}_i). \tag{1.1}$$

We define the *conditional reward* of a channel $\mathcal{C}_i$ (denoted by $\mathcal{R}^c(\mathcal{C}_i)$) as the expected number of bits that can be successfully transmitted on $\mathcal{C}_i$ given that $\mathcal{C}_i$ is sensed free. Formally,

$$\mathcal{R}^c(\mathcal{C}_i) = \frac{\theta(\mathcal{C}_i)(1 - \alpha(\mathcal{C}_i))}{\phi(\mathcal{C}_i)} B(\mathcal{C}_i). \tag{1.2}$$

Furthermore, we define the *blind reward* of a channel $\mathcal{C}_i$ (denoted by $\mathcal{R}^b(\mathcal{C}_i)$) as

$$\mathcal{R}^b(\mathcal{C}_i) = \theta(\mathcal{C}_i)(1 - \alpha(\mathcal{C}_i))B(\mathcal{C}_i) = \phi(\mathcal{C}_i)\mathcal{R}^c(\mathcal{C}_i), \tag{1.3}$$

that is the expected number of bits that can be successfully transmitted on $\mathcal{C}_i$ before knowing the result of sensing. Note that the secondary user does not transmit on a sensed-busy channel. Also, because of miss-detection, a transmission by the secondary user on a sensed-free channel may not be successful. Without loss of generality, throughout the thesis, we assume that

$$\mathcal{R}^c(\mathcal{C}_1) \leq \mathcal{R}^c(\mathcal{C}_2) \leq \ldots \leq \mathcal{R}^c(\mathcal{C}_N),$$

that is channels $\mathcal{C}_1, \ldots, \mathcal{C}_N$, are sorted by their conditional rewards.

1.4 Problem Definition & Objective

The objective is to maximize SU's *gain*, defined as the expected number of bits successfully transmitted by secondary user in a slot. To achieve this, secondary user first needs to decide on what channels to sense. Then, if the number of channels sensed free is more than K, secondary user has to decide on what K channels to access. The second decision is easy to make; As stated in [2], the optimum solution is to access the K channels with the K largest conditional rewards. The first decision, on the other hand, is difficult in general, and is the target of this work. More formally, the problem is defined as follows. Given N channels $\mathcal{C}_1, \ldots, \mathcal{C}_N$ with characteristics $\theta(\mathcal{C}_i), \alpha(\mathcal{C}_i), \mu(\mathcal{C}_i)$, $1 \leq i \leq N$, let random variables $X_1, \ldots, X_N$ be

$$X_i = \begin{cases} \mathcal{R}^c(\mathcal{C}_i) & \text{with probability } \phi(\mathcal{C}_i); \\ 0 & \text{with probability } 1 - \phi(\mathcal{C}_i). \end{cases}$$

Also for a subset $S \subseteq \{\mathcal{C}_1, \ldots, \mathcal{C}_N\}$ let

$$\Psi(S) = \sum_{i=0}^{K-1} \mathcal{X}_{(N-i)}$$

where $\mathcal{X}_{(i)}$ is the i_{th} *order statistic* of $X_1, \ldots, X_N$.

The problem is to find

$$\underset{\substack{S \subseteq \{\mathcal{C}_1, \ldots, \mathcal{C}_N\} \\ |S| = M}}{\operatorname{argmax}} \; \Psi(S).$$

1.5 Thesis Motivations and Contributions

Strict wireless network regulations has resulted in inefficient radio spectrum usage. A considerable part of costly radio spectra is under-utilized, while some part of it is heavily used [7, 8]. One of the most promising approaches in exploiting the allocated bandwidth more efficiently is cognitive radio in which an unlicensed (secondary) user can take advantage of the under-utilized radio channels. To avoid disturbing licensed (primary) users, a channel must always be sensed free by a secondary user before each transmission. Also, a secondary user must immediately back-off as soon as a primary user starts using the spectrum. Implementing these restrictions has been one of the main challenges in the design of cognitive radio strategies and algorithms.

In practice, there are usually far more channels available than what a single secondary user needs. With the increase in the number of available channels the search space grows exponentially. This leads to the problem of fast optimal channel set selection. Suitability of a channel selection strategy depends on the design objective, which can be optimizing power consumption, finding the required free channels in a limited time, or minimizing the chance of disturbing the primary user. Several different models and objectives to this problem exist in the literature. In some models, the sensing is assumed to be perfect without *a priori* knowledge on channel availability and activity, while some other models, including the one adopted in this work, assume imperfect channel sensing with *a priori* knowledge of channel activity. In the model with imperfect channel sensing, at the beginning of a time slot, the secondary user senses a set of M channels. When it comes to accessing channels which are sensed free, two scenarios can be considered. In the first scenario, the secondary user access all the sensed-free channels. In this scenario, the optimal set of channels to sense is easily found [2]. We call this approach the *intuitive solution*, since, it intuitively selects channels with highest expected “rewards”. In the other scenario, the secondary user only needs to access a limited number K, $K < M$, of the sensed-free channels. Interestingly, as shown in [2] through an example, in this scenario, the *intuitive solution* is not necessarily optimal. Motivated by this fact, in this thesis we study the gap between the performances of the intuitive solution and the optimal solution. We also study the optimal solution when $K < M$. Some of the contributions of this work are:

- We prove that the performance ratio between *optimal solution* and the *intuitive solution* can grow linearly with the number of channels. We explain a scenario where such large gaps appear, and use simulations to verify that.

- We show that there is a single channel that is part of an optimal solution for any K, M, and N.

- We propose an optimal polynomial-time solution for $K = 1$. This is an interesting special case, as simulation results suggest that the performance gap is maximized when $K = 1$.

- We propose a polynomial-time algorithm to calculate the maximum throughput for a given set of channels when $N = M$. Using this algorithm, we propose a sub-optimal algorithm for the general case, derive an upper bound on the performance of the optimal solution, and show that polynomial-time optimal solution exist when M is small or when it is large (that is close to N).

1.6 Thesis Outline

This thesis is organized as follows. In Chapter 1, we describe an abstracted game presentation of the problem. We also explain the model in details, and formally define the problem. Chapter 2 covers some related works, and explains some necessary background to the thesis material. In Chapter 3, we present the main contribution. This chapter is further divided into the following sections. In Section 3.1, we prove that there is a linear gap between the optimal and intuitive solution. Section 3.2 provides a recursive equation to calculate the optimal solution, and show that there is a channel which is part of any optimal solution. In Section 3.3, using dynamic programming, we propose an optimal polynomial-time solution for the case $K = 1$. Using a similar dynamic programming, we derive an upper bound on SU's gain for the general case of $K \leq M$ in Section 3.4. We also provide a sub-optimal solution in Section 3.5. In Chapter 4 we use simulation to verify some of the analytical results. Finally, Chapter 5 discusses some future directions and concludes the thesis.

Chapter 2

Background & Related Work

2.1 Asymptotic Notations

Asymptotic Notations are used to compare growth rates of mathematical functions. There are two main notations used more often than the others: i) $\mathcal{O}(.)$ notation, and ii) $\Omega(.)$ notation. These two notations show the upper and lower bound on the asymptotic growth rate of a function, and are related with each other according to the following formula

$$f(x) = \Omega(g(x)) \iff g(x) = \mathcal{O}(f(x)) \tag{2.1}$$

The $\mathcal{O}(.)$ and $\Omega(.)$ notations are also defined as sets of all the functions with the given characteristics, so we can also write the equation as

$$f(x) \in \Omega(g(x)) \iff g(x) \in \mathcal{O}(f(x))$$

1. **Big-O Notation**:

   The $\mathcal{O}(.)$ Notation shows an upper bound on the growth rate of a function.

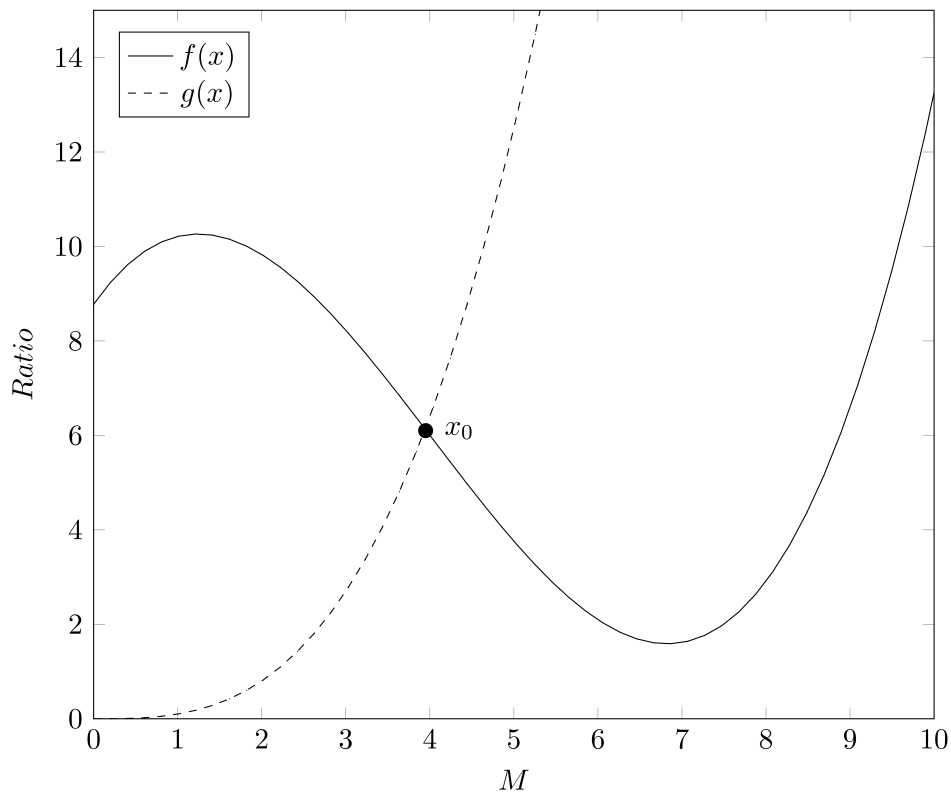

Figure 2.1: $f(x) = \mathcal{O}(g(x))$ as there exist a real number c (for example, $c = 1$), such that for all numbers larger than $x_0 = 4$ we have $f(x) \leq cg(x)$. Using Eq. [2.1], we also have $g(x) = \Omega(f(x))$.

Formally speaking

$$f(x) = \mathcal{O}(g(x)) \iff \forall_{x \geq x_0} |f(x)| \leq c|g(x)|,$$

for some positive real number c, and some real number x_0. As shown in Figure 2.1, $f(x) = \mathcal{O}(g(x))$, which roughly means $g(x)$ asymptotically faster than $f(x)$.

2. **Big-Omega Notation**:

   The $\Omega(.)$ notation shows a lower bound on the growth rate of a function. As shown in Figure 2.1, $g(x) = \Omega(f(x))$, which again roughly means $f(x)$ grows asymptotically slower than $g(x)$. A very widespread definition of $\Omega(.)$ is based on the definition of $\mathcal{O}(.)$ which was presented in Eq. [2.1].

Example:
Let

$$f(x) = x^5 - x^3 + 2x^2 + 3.$$

Then $\mathcal{O}(.)$ and $\Omega(.)$ will contain

$$\mathcal{O}(f(x)) = \{x^5, x^5 - x^4 - 7, 100x^2, \log x, \cdots\},$$
$$\Omega(f(x)) = \{x^5, x^5 - x^4 - 7, x^7, 2^x, \cdots\}.$$

If a function is in both $\mathcal{O}(.)$ and $\Omega(.)$ sets, it is called to be a member of $\Theta(.)$. So, based on the example we have

$$\Theta(f(x)) = \{2x^5, x^5 - x^4 - 7, \cdots\}.$$

2.1.1 Time/Space Complexity of Algorithms

The run time of an algorithm is a function of the length of the string representing the input. The time complexity of an algorithm quantifies this, and is

commonly expressed using asymptotic notations such as those discussed earlier. For example, let n denote the input size. We say that time complexity of an algorithm is $\mathcal{O}(n^4)$, if the function representing the run time of the algorithm is $\mathcal{O}(n^4)$. Similarly, the space complexity of an algorithm is defined. Roughly speaking, the time and space complexity of an algorithm give us a feeling about the run time and the space requirement of the algorithm for large inputs.
Five common complexity classes are: 1)*constant*, 2)*Logarithmic*, 3)*Linear*, 4)*Polynomial*, and 5)*Exponential*.

1. **Constant**: An algorithm has constant time complexity if its run time is always less than t units of time, for some real number t. In other words, the function representing the run time is $\Theta(1)$.

   An example of an algorithm with constant run time is finding the minimum of a sorted list. The algorithm just needs to check the first member of the list, independent of how long the list is.

2. **Logarithmic**: An algorithm has logarithmic time complexity if its run time grows logarithmically with the input size. We say such algorithms have $\Theta(\log(n))$ time complexity.

   A well-known example of an algorithm with logarithmic run time is *Binary Search*. Binary search is used to find or validate the existence of a member in a sorted list. The algorithm works by turning the list into a virtual tree. In each step, the algorithm checks the middle member of the list, which will be the root of the tree. If that member is smaller than the desired value, then the desired value must be on the right side, which the node's right sub tree. If the it is greater than the desired value, then the search must continue on the left sub tree. This process will be repeated on the targeted sub tree until either there are no more sub trees or the desired member is found. Since the size of the list is divided by two in each step, there can be at most $\log(N)$ operations on a list of size N, hence, a logarithmic run time.

3. **Linear**: An algorithm has linear time complexity if its run time function is $\Theta(n)$.

   A simple example of an algorithm with linear run time is finding the maximum member of an unsorted list. The algorithm just goes through the list from the beginning of the list to the end and compares each member with the maximum member up to there. Since the algorithm just makes a comparison on each member it will run in linear time.

4. **Polynomial**: An algorithm has polynomial time complexity if its run time function is $\Theta(n^m)$ for some integer m.

   An example of an algorithm with polynomial run time is *Bubble Sort*. Bubble sort goes through a list and compares each neighbouring pair, swapping them if they are not in the correct order. As a result each time the algorithm goes through the list the last member will definitely be in the correct place. As a result, each time the algorithm can ignore the new last member and repeats the process on the reminder of the list. After N rounds on a list of length N the list will be sorted. Each of these rounds goes through the list and performs $\mathcal{O}(N)$ calculations. As a result, the algorithm performs $\mathcal{O}(N^2)$ calculations, hence, a polynomial time algorithm.

5. **Exponential**: An algorithm has exponential time complexity if its run time function is $\Theta(m^n)$ for some real number $m > 1$. Algorithms with exponential time complexities are considered impractical.

   Algorithm 1 a simple algorithm that calculates the N_{th} member of the Fibonacci sequence, and has exponential run time time. To calculate the N_{th} member of the Fibonacci sequence, Algorithm 1 calculates the $(N-1)_{th}$ and $(N-2)_{th}$ members, which also are recursively dependent

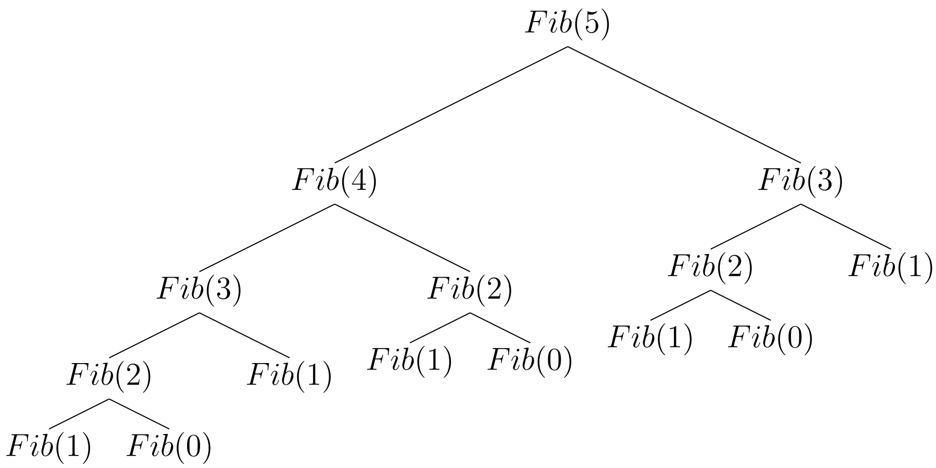

Figure 2.2: The Fibonacci Tree

on other values. This will create a binary tree shown in Figure 2.2 with maximum height N which will have $\mathcal{O}(2^N)$ nodes, hence, an exponential run time.

Algorithm 1 The recursive algorithm written in C programming language. This code is simple but computationally inefficient, the run time of the code is $\mathcal{O}(2^n)$ and the space requirement is $\mathcal{O}(n)$ (The recursive function call needs to keep the information in stack).

```c
int Fibonacci(int n)
{
  if(n==0 || n==1)
    return 1;
  return Fibonacci(n-1)+Fibonacci(n-2);
}
```

2.2 Dynamic Programming

Many computationally intensive problems are made up of smaller sub problems. *Dynamic Programming* takes advantage of these smaller sub problems and tries to build the main solution from bottom up. Since many of these smaller sub problems repeat themselves in the process, solving each of these sub problems once and using the solution in future occurrences of the problem can boost the

Index	$Fib(0)$	$Fib(1)$	$Fib(2)$	$Fib(3)$	$Fib(4)$	$Fib(5)$	$\cdots$
Values	1	1	2	3	5	8	$\cdots$

Figure 2.3: The Fibonacci Table

performance of overall solution significantly.

Example:

One of the most famous examples of a problem with repetitive sub problems is finding the N_{th} member of the Fibonacci sequence. Each Fibonacci number is calculated based on the value of the two previous numbers. As it is shown in Figure 2.2 the number of nodes in the tree grows exponentially as the Fibonacci number grows (Finding $Fibonacci(40)$ using Algorithm 1 will be impossible in plausible time). However, most of the values in the tree are redundant and are not necessary to recalculate. As a result, by breaking the problem into sub problems and changing our approach to Figure 2.3 and Algorithm 2, we can find the N_{th} Fibonacci number in $\mathcal{O}(n)$ (which allows us to calculate $Fibonacci(2^{20})$ almost instantly).

Algorithm 2 The use of dynamic programming on the Fibonacci algorithm reduces the run time to $\mathcal{O}(n)$. The space requirement is still $\mathcal{O}(n)$ for keeping the table shown in Figure 2.3.

```
int Fibonacci(int n)
{
  int DP[N];
  DP[0]=DP[1]=1;
  for(int i=2;i<=n;i++)
    DP[i]=DP[i-1]+DP[i-2];
  return DP[n];
}
```

Algorithm 2 space consumption is $\mathcal{O}(n)$ which can be reduced by not keeping the result of unnecessary sub problems when there is no more use for them. Algorithm 3 shows this enhancement by replacing the table shown in Figure 2.3 with a small, constant size table of size three.

Algorithm 3 Sometimes, in dynamic programming algorithms, keeping the result of old calculations is not necessary. This allows us to reduce the space complexity of the algorithm significantly. For example, the code below reduces the space complexity of Algorithm 2 to $\mathcal{O}(1)$ without affecting the time complexity.

```
int Fibonacci(int n)
{
  int DP[3];
  DP[0]=DP[1]=1;
  for(int i=2;i<=n;i++)
    DP[i%3]=DP[(i%3)-1]+DP[(i%3)-2];
  return DP[n%3];
}
```

2.2.1 Two-Dimensional Dynamic Programming

A two-dimensional dynamic programming is very similar to a one-dimensional dynamic programming with the difference that it deals with two input variables instead of one. For these kind of problems we can use a matrix (2D array) to store the calculated outputs.

Example:

A famous problem taking advantage of two-dimensional dynamic programming is finding the number of paths between two points (a source and a destination) on a grid with the minimum Manhattan distance (the distance when only allowed to move horizontally or vertically). As seen in Figure 2.4, there are some blocked cells between the source and destination. In order to find all different paths, we can start from the destination and count the number of paths based on neighbouring cells. If we assume that there is one path from the destination to itself, then the number of paths from each cell to the destination will be the sum of its upper and right neighbour paths to the destination, with the exception that the blocked cells will have no path to the destination.

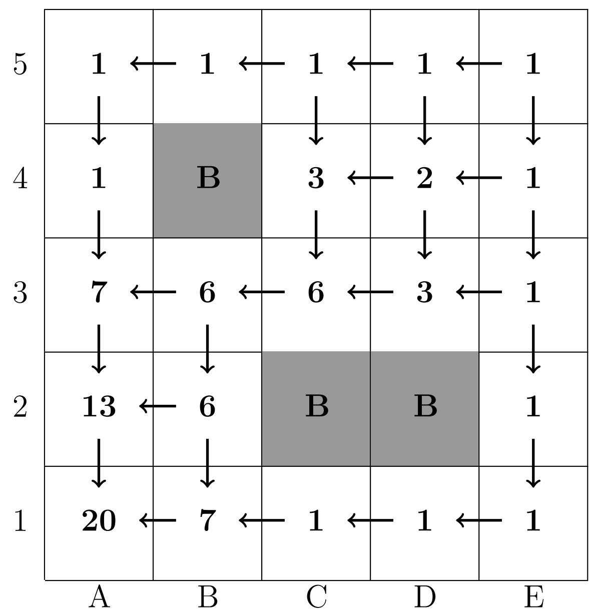

Figure 2.4: The figure shows the process of finding the number of different paths from the cell *A1* to cell *E5* with the minimum Manhattan distance (the distance when only allowed to move horizontally or vertically). As it can be seen, the number of paths for each cell is the sum of its top and right neighbour, with the exception of blocked cells (shown by the character **B**) which cannot have a path go through them. Using this method and starting to fill the table from the destination and continuing by filling each cell's neighbours, the result can be seen in cell *A1*.

2.2.2 Higher Dimensional Dynamic Programming and Special cases

For higher dimensions of dynamic programming, the logic stays the same and the only factor changing is the number of input variables which our answer depend on. For each variable, we need to add another dimension to our array.

In some special cases, we can use other data structures other than an array to save space. For example, in some cases, the dependency is not on all values and our multidimensional matrix will be sparse. In these cases, it might be useful to take advantage of a dependency graph, a tree, or many other methods used for saving space in sparse matrices.

2.3 Related Work

Our work is mainly based on the paper published by Zhou Zhang et al. about choice of channels in cognitive radios with imperfect sensing [2]. In their work, they analyze some special cases and present the optimal solution for them:

- **With homogeneous sensing**: In this category, they consider the case where all channels have the same miss detection probability, that is

  $$\forall_{\mathcal{C}\in S} \quad \mu\left(\mathcal{C}\right) = \mu,$$

  where μ is a constant, and they all have the same false alarm probability, i.e.

  $$\forall_{\mathcal{C}\in S} \quad \alpha\left(\mathcal{C}\right) = \alpha,$$

  where α is a constant. In this case, they show that intuitively sorting the channels by their probability of being free $\theta\left(\mathcal{C}\right)$ and choosing the M largest will give the optimal solution.

- **With common detection probability**: This category, also assumes all the channels to have the same false alarm probability

$$\forall_{\mathcal{C}\in S} \quad \alpha\left(\mathcal{C}\right) = \alpha,$$

but the channels can have different miss detections. In this case, they show that if sorting the channels by their probability of being free $\theta\left(\mathcal{C}\right)$ gives the same order as sorting them by their blind reward $\mathcal{R}^{b}\left(\mathcal{C}\right)$, then the solution again will be choosing the M channels with the largest probability of being free $\left(\theta\left(\mathcal{C}\right)\right)$.

- **General case**: In the general case Zhou Zhang et al. show that if sorting the channels by their probability of being sensed free $\phi\left(\mathcal{C}\right)$ will also sort the channels by their *Conditional Reward* $\mathcal{R}^{c}\left(\mathcal{C}\right)$, then the optimal set of channels to sense to create the optimal reward will be the M channels with the largest *Conditional Reward* $\mathcal{R}^{c}\left(\mathcal{C}\right)$ which is also the same as the M channels with the largest probability of being sensed free $\phi\left(\mathcal{C}\right)$.

One of the main generalization we made to this work is using a general bandwidth $B\left(\mathcal{C}\right)$, in contrast to Zhou Zhang et al. who set the bandwidth value $B\left(\mathcal{C}\right) = 1$ (the same as $\Gamma = 1$ in our notations) to simplify the formulas.

The other existing related works can be categorized by either the adopted model or their objective. Some models assume no prior knowledge of the system. In particular, these models assume no prior information about channel availability and their distribution. A model based on the above assumption is the one used by Xi Fang et al. [9]. In their model, they assume perfect sensing, and that any sensed-free channel can be accessed.

On the other hand, some models assume *a priori* knowledge of the system, either the probability of the channels availability or a distribution of the channels usage. For example, in the model used by Zhou Zhang et al. in [2], it is assumed that the probability of channels availability is known. We use the same assumption in our model.

Another criteria that can distinguish different models is whether the channel sensing is perfect or faulty. Models like the one used by Zhou Zhang et al. assume imperfect sensing [10]. In these models, a free channel can be sensed busy, and a busy channel can be sensed free. Our model uses this assumption as this is the case in practice.

The existing work can also be categorized by their objectives. Some existing work try to optimize the total amount of data transmitted [9], while some other try to optimize the energy consumption. For example, [11] focuses on energy consumption and optimizing the length and power of the sensing and transmission periods.

Hiteshi Sharma et al. models a system with imperfect sensing where the secondary user can only sense one channel at a time. They use Optimal Stopping Problem to decide whether a sensed free channel should be used for transmission or the secondary user should ignore it due to its quality and continue sensing other channels [12].

Hossein Shokri-Ghadikolaei et al. focuses on ensuring a minimum Quality of Service for the primary user while the secondary user uses a randomly selected order of channels to roam between them in order to avoid overlapping with the primary user. They use Markov chain analysis to find the average throughput for both the primary and secondary user [13].

In this thesis, we use a simple but practical definition of the problem used in [2]. Our model assumes imperfect sensing. It also assumes that, in each time slot, the secondary user can sense a limited number of channels, which can be more than the number channels needed for transmission. These characteristic are simplified into some probabilistic values explained in Section 1.3. As in [2] our objective is to maximize the average number of bits transmitted.

Chapter 3

Channel Sensing in Cognitive Radio Networks

3.1 Optimal Solution vs. Intuitive Solution

As mentioned earlier, the intuitive solution is optimal when $K = M$. However, as shown in [2] through an example, the gain offered by the optimal solution can be higher than that achieved by the intuitive solution when $K < M$. One might wonder if the gap between the gains achieved by the two solutions can grow arbitrarily large as the number of channels increases. In this section, we show that this is indeed the case, in the worst case. That is, we can select channels' parameters in a way that the gain achieved by the optimal solution is arbitrary higher than the gain achieved by the intuitive solution.

Proposition 1. *Assume $B(\mathcal{C}_i) = B(\mathcal{C}_j)$ for $1 \leq i, j \leq N$, $K = 1$ and $N \geq 2M$. Let S_1, S_2, and S_3 be three disjoint subsets of channels with $|S_1| = M$, $|S_2| = M$, and $|S_3| = N - 2M$. Suppose*

$$\forall \mathcal{C} \in S_1 \quad \theta(\mathcal{C}) = \theta_1 = \frac{1}{M} + \epsilon,$$

for some arbitrary small positive real number ϵ, and

$$\forall \mathcal{C} \in S_2 \quad \theta(\mathcal{C}) = \theta_2 = \frac{1}{M},$$

$$\forall \mathcal{C} \in S_3 \quad \theta(\mathcal{C}) = 0,$$

$$\forall \mathcal{C} \in S_i, i = \{1,2,3\} \quad \alpha(\mathcal{C}) = \alpha_i = 0,$$

$$\forall \mathcal{C} \in S_i, i = \{2,3\} \quad \mu(\mathcal{C}) = \mu_i = 0,$$

$$\forall \mathcal{C} \in S_1 \quad \mu(\mathcal{C}) = \mu_1,$$

for some constant number μ_1.

Then, the gain achieved by the optimal solution is a factor of $\Omega(M)$ larger than the gain achieved by the intuitive solution.

Proof. We have

$$\forall \mathcal{C} \in S_i, i = \{1,2,3\} \quad \theta(\mathcal{C}) = \theta_i, \alpha(\mathcal{C}) = \alpha_i \\ \textit{and, } \mu(\mathcal{C}) = \mu_i,$$

therefore

$$\forall \mathcal{C} \in S_i, i = \{1,2,3\} \quad \phi(\mathcal{C}) = \phi_i \textit{ and } \mathcal{R}^b(\mathcal{C}) = \mathcal{R}^b_i,$$

for some real numbers ϕ_i and $\mathcal{R}^b_i$. Thus, by the definition of $\mathcal{R}^b$, we get

$$\mathcal{R}^b_2 = \mathcal{R}^b_1 - \epsilon' > \mathcal{R}^b_3,$$

where $\epsilon' \to 0$ as $\epsilon \to 0$.

The intuitive solution chooses the M channels with the largest $\mathcal{R}^b(\mathcal{C})$, which in this case will be channels in S_1. Now, consider another solution that chooses the set of channels S_2. Let's call this the "Non-intuitive solution". The ratio of the gains of the solutions will be

$$\frac{\mathcal{R}_2^b + (1-\phi_2)\,\mathcal{R}_2^b + \cdots + (1-\phi_2)^{M-1}\,\mathcal{R}_2^b}{\mathcal{R}_1^b + (1-\phi_1)\,\mathcal{R}_1^b + \cdots + (1-\phi_1)^{M-1}\,\mathcal{R}_1^b}.$$

As ϵ approaches zero, the above ratio approaches

$$\frac{1 + (1-\phi_2) + (1-\phi_2)^2 + \cdots + (1-\phi_2)^{M-1}}{1 + (1-\phi_1) + (1-\phi_1)^2 + \cdots + (1-\phi_1)^{M-1}}$$

which can be simplified to

$$\begin{aligned}\zeta =& \frac{\dfrac{1-(1-\phi_2)^M}{1-(1-\phi_2)}}{\dfrac{1-(1-\phi_1)^M}{1-(1-\phi_1)}} = \frac{\dfrac{1-(1-\phi_2)^M}{\phi_2}}{\dfrac{1-(1-\phi_1)^M}{\phi_1}} \\ =& \frac{\phi_1}{\phi_2} \cdot \frac{1-(1-\phi_2)^M}{1-(1-\phi_1)^M}.\end{aligned} \tag{3.1}$$

Since $\alpha_1 = \alpha_2 = 0$ and $\mu_2 = 0$

$$\phi_1 = \frac{1}{M}(1-\alpha_1) + \left(1 - \frac{1}{M}\right)\mu_1 = \frac{1}{M} + \left(1 - \frac{1}{M}\right)\mu_1 \tag{3.2}$$

and

$$\phi_2 = \frac{1}{M}(1-\alpha_2) + \left(1 - \frac{1}{M}\right)\mu_2 = \frac{1}{M}.$$

Replacing ϕ_2 with $\frac{1}{M}$ in (3.1), we get

$$\begin{aligned}\zeta =& \frac{\phi_1}{\phi_2} \cdot \frac{1-(1-\phi_2)^M}{1-(1-\phi_1)^M} = \frac{\phi_1}{\frac{1}{M}} \cdot \frac{1-\left(1-\dfrac{1}{M}\right)^M}{1-(1-\phi_1)^M} \\ \geq& \frac{\phi_1}{\frac{1}{M}} \cdot \frac{1-\left(e^{-1}\right)}{1-(1-\phi_1)^M} \geq \frac{\phi_1}{\frac{1}{M}} \cdot \frac{0.63}{1-(1-\phi_1)^M} \\ =& 0.63M\left(\frac{\phi_1}{1-(1-\phi_1)^M}\right) \geq 0.63M\left(\frac{\mu_1}{1-(1-\mu_1)^M}\right),\end{aligned}$$

where the last inequality holds because $\frac{\phi_1}{1-(1-\phi_1)^M}$ is an increasing function of ϕ_1, and $\phi_1 > \mu_1$ by (3.2). Finally, note that $\frac{\mu_1}{1-(1-\mu_1)^M} > \mu_1$, therefore when μ_1 is a constant, we get $\zeta = \Omega(M)$. For example, setting $\mu_1 = 10\%$ (a typical maximum value for μ_1), we get $\zeta \geq 0.063M$. □

The result of Proposition 1 is verified by simulation in Chapter 4 (see Figure 4.1)

3.2 A Recursive Equation

For a set of channels $S \subseteq \{\mathcal{C}_1, \ldots, \mathcal{C}_n\}$, and a non-negative integer k, let $\mathbf{G}_k^S$ denote the maximum gain that can be achieved if all channels in S are sensed first, and then up to k sensed-free channels are accessed. We define $\mathbf{G}_{n,m,k}$ as

$$\mathbf{G}_{n,m,k} = \begin{cases} \max\limits_{\substack{S \subseteq \{\mathcal{C}_1,\ldots,\mathcal{C}_n\} \\ |S|=m}} \mathbf{G}_k^S & n \geq m; \\ 0 & otherwise. \end{cases} \tag{3.3}$$

In other words, $\mathbf{G}_{n,m,k}$ is the maximum achievable gain over the set of channels $\{\mathcal{C}_1, \ldots, \mathcal{C}_n\}$, if only m channels can be sensed, and up to k sensed-free channels can be accessed.

Note that the conditional reward of channel $\mathcal{C}_i$ is greater or equal to that of $\mathcal{C}_j$, for $i > j$. Therefore, when it comes to choosing between $\mathcal{C}_i$ and $\mathcal{C}_j$ for access, channel $\mathcal{C}_i$ should be chosen if both channels $\mathcal{C}_i$ and $\mathcal{C}_j$ are sensed free. As a consequence, when channel sensing is done (i.e., at the beginning of the transmission phase), a way to maximize gain is to iterate through channels from $\mathcal{C}_n$ to $\mathcal{C}_1$, and select a channel $\mathcal{C}$ for access if i) $\mathcal{C}$ was sensed free; ii) the number of channels that were selected so far is at most $k-1$ (that is, we stop selecting channels for access if we reach k, the maximum number of channels that needs to be accessed). We call this selection approach the *order-based selection process*. Note that, when there are channels with identical conditional rewards, the above process may not be the only optimal way to select channels for access. However, without loss of generality, we assume that, for selecting channels to access, the above process is used, since it always achieves the maximum gain. The following proposition provides a recursive equation to calculate the maximum gain. This proposition is the base of our proposed dynamic programming algorithms, as well as the upper bound derived in Section 3.4.

Proposition 2. *We have*

$$\boldsymbol{G}_{n,m,k} = \max\{ \boldsymbol{G}_{n-1,m,k}, \mathcal{R}^b\left(\mathcal{C}_n\right) + \boldsymbol{H}_{n-1,m-1,k}\}, \tag{3.4}$$

where

$$\boldsymbol{H}_{n-1,m-1,k} = \max_{\substack{S\subseteq\{\mathcal{C}_1,\ldots,\mathcal{C}_{n-1}\}\\ |S|=m-1}} \{\phi\left(\mathcal{C}_n\right) \boldsymbol{G}^S_{k-1} + \left(1-\phi\left(\mathcal{C}_n\right)\right) \boldsymbol{G}^S_k\}. \tag{3.5}$$

Proof. To calculate $\mathbf{G}_{n,m,k}$, we consider two cases. In the first case, $\mathcal{C}_n$ is not selected to be sensed. In this case, $\mathbf{G}_{n,m,k}$ is clearly equal to $\mathbf{G}_{n-1,m,k}$. In the second case, $\mathcal{C}_n$ is selected to be sensed. In this case, if $\mathcal{C}_n$ is sensed free, it will be selected for access based on the *order-based selection process*. Therefore, in this case, the maximum gain will be

$$\max_{\substack{S\subseteq\{\mathcal{C}_1,\ldots,\mathcal{C}_{n-1}\}\\ |S|=m-1}} \left(\phi\left(\mathcal{C}_n\right) \mathbf{G}^S_{k-1} + \left(1-\phi\left(\mathcal{C}_n\right)\right)\mathbf{G}^S_k\right)$$
$$+\phi\left(\mathcal{C}_n\right)\mathcal{R}^c\left(\mathcal{C}_n\right) = \mathcal{R}^b\left(\mathcal{C}_n\right) + \mathbf{H}_{n-1,m-1,k},$$

which completes the proof. □

An interesting result following Proposition 3 is as follows. Let $\mathcal{C}^*$ be a channel with the largest blind reward[1]. Then C^* is always part of an optimal

[1] There may be more than one channel with the largest blind reward.

Channels	$\theta(\mathcal{C})$	$\alpha(\mathcal{C})$	$\mu(\mathcal{C})$	$\mathcal{R}^c(\mathcal{C})$
$\mathcal{C}_1$	0.1	0.1	0.0	1.0
$\mathcal{C}_2$	0.5	0.0	0.1	0.909
$\mathcal{C}_3$	0.9	0.1	0.1	0.987

Table 3.1: In the case $(N = 3, M = 2, K = 1)$, $\mathcal{C}_1$ has the greatest *conditional reward*, but the best gain is achieved using $S = \{\mathcal{C}_2, \mathcal{C}_3\}$ where $\mathbf{G}_1^S = 0.899$.

solution for any n, m, and $k > 0$. A channel(s) with the largest *conditional reward* does not have this property. A simple example to show this is presented in Table 3.1 where $\mathcal{C}_1$ has the greatest *conditional reward* but the best choice is sensing $\mathcal{C}_2, \mathcal{C}_3$.

Proposition 3. *The channel $\mathcal{C}^*$ is always part of an optimal solution*[2].

Proof. Suppose

$$S = \{\mathcal{C}'_1, \mathcal{C}'_2, \ldots, \mathcal{C}'_{m-1}, \mathcal{C}'_m\} \subseteq \{\mathcal{C}_1, \ldots, \mathcal{C}_n\}$$

is an optimal set of channels to be sensed. Without loss of generality, assume that the conditional reward of $\mathcal{C}'_i$ is at least equal to that of $\mathcal{C}'_j$ if $i \geq j$. Suppose that $\mathcal{C}^* \notin S$. We show that the maximum achievable gain does not reduce if we replace $\mathcal{C}'_1$ with $\mathcal{C}^*$.

By the order-based selection process, if the number of channels in $S \setminus \{\mathcal{C}'_1\}$ that are sensed free is at least k, then $\mathcal{C}'_1$ is not accessed (even if it is sensed free). Clearly, in this case, replacing $\mathcal{C}'_1$ with $\mathcal{C}^*$ does not affect the gain. However, in the second case, where the number of channels in $S \setminus \{\mathcal{C}'_1\}$ that are sensed free is less than k, $\mathcal{C}'_1$ will be accessed if it is sensed free. In this case, replacing $\mathcal{C}'_1$ with $\mathcal{C}^*$ does not reduce the gain as the blind reward of $\mathcal{C}^*$ is not less that that

[2]If there is only one channel with the largest blind reward, then that channel will be part of any optimal solution, for any N, M, and $K > 0$.

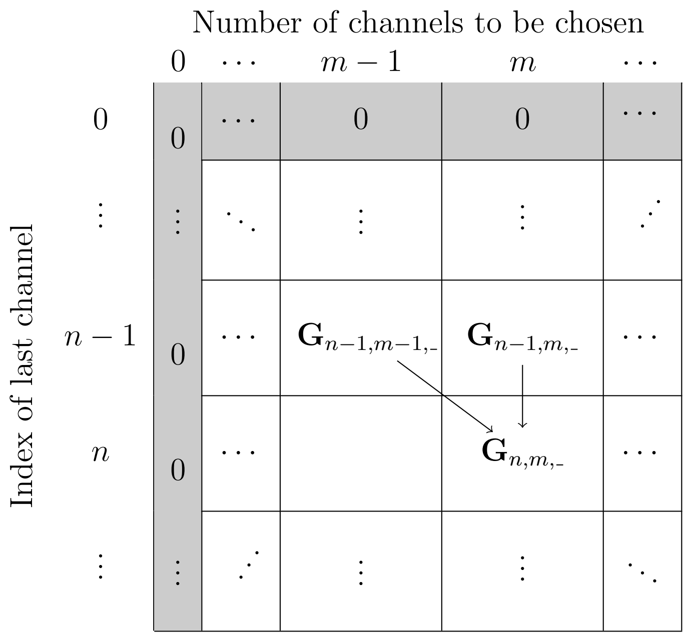

Figure 3.1: Each cell can be calculated using upper and upper left cells, as a result the total complexity would be $\mathcal{O}(NM)$. A binary matrix is used to record if the value of a cell comes from the upper cell or the upper left cell, this matrix can be traversed at the end to create the set of M channels.

of $\mathcal{C}_1'$. $\square$

3.3 Optimal Solution: $K = 1$

In this special case, secondary user senses m channels, and needs to access only one sensed-free channel for data transmission. We use $\mathbf{G}_{n,m,\_}$ to represent $\mathbf{G}_{n,m,1}$ to differentiate the general case formula from the special case formula. The following corollary directly follows from Proposition 2.

Corollary 1. *We have*

$$\boldsymbol{G}_{n,m,\_} = \max\{\boldsymbol{G}_{n-1,m,\_}, \mathcal{R}^b(\mathcal{C}_n) + (1 - \phi(\mathcal{C}_n))\boldsymbol{G}_{n-1,m-1,\_}\}. \tag{3.6}$$

Proof. We have

$$\begin{aligned}
\mathbf{H}_{n-1,m-1,1} &= \max_{\substack{S\subseteq\{\mathcal{C}_1,\ldots,\mathcal{C}_{n-1}\}\\|S|=m-1}} \left(\phi\left(\mathcal{C}_n\right)\mathbf{G}_0^S + (1-\phi\left(\mathcal{C}_n\right))\mathbf{G}_1^S\right)\\
&= \max_{\substack{S\subseteq\{\mathcal{C}_1,\ldots,\mathcal{C}_{n-1}\}\\|S|=m-1}} (1-\phi\left(\mathcal{C}_n\right))\mathbf{G}_1^S\\
&= (1-\phi\left(\mathcal{C}_n\right)) \max_{\substack{S\subseteq\{\mathcal{C}_1,\ldots,\mathcal{C}_{n-1}\}\\|S|=m-1}} \mathbf{G}_1^S\\
&= (1-\phi\left(\mathcal{C}_n\right))\mathbf{G}_{n-1,m-1,1}.
\end{aligned}$$

By replacing the above formula in Eq. [3.4], we get

$$\mathbf{G}_{n,m,1} = \max\{\mathbf{G}_{n-1,m,1}, \mathcal{R}^b\left(\mathcal{C}_n\right) + (1-\phi\left(\mathcal{C}_n\right))\mathbf{G}_{n-1,m-1,1}\}.$$

□

Figure 3.1 illustrates how Corollary 1 can be used to compute $\mathbf{G}_{N,M,\_}$, which is the maximum gain achievable by SU given the set of all N channels. As depicted in Figure 3.1, value of every cell can be computed using values of its top and top-left neighbouring cells. Also, $\mathbf{G}_{n,0,\_}$ and $\mathbf{G}_{0,m,\_}$ are zeros for every $0 \leq n \leq N$, and $0 \leq m \leq M$. Therefore, values of all cells can be computed by scanning the table row by row starting from top left.

Using the above dynamic programming, we can not only compute the maximum gain, but also find a set of channels that achieve that maximum gain. To do so, a separate binary matrix can be used to keep track of which neighbour has been used in calculating each cell; if the value comes from $\mathbf{G}_{n-1,m,\_}$ then it means that channel $\mathcal{C}_n$ is not part of the solution, and if it comes from $\mathbf{G}_{n-1,m-1,\_}$ it implies that channel $\mathcal{C}_n$ is a part of the solution. At the end, by traversing back the choices cell by cell from $\mathbf{G}_{N,M,\_}$, the list of M channels can be created.

Proposition 4. *The running time and space complexity of the proposed dy-*

namic programming is $\mathcal{O}(NM)$.

Proof. As the recursive Eq. [3.6] has two arguments $1 \leq n \leq N$ and $1 \leq m \leq M$, then there would be at most $N \times M$ possible values for $\mathbf{G}_{n,m,\_}$. Since every value can be calculated in $\mathcal{O}(1)$ using the dynamic programming approach explained above, then we need $\mathcal{O}(NM)$ operations to calculate the final result. The space complexity is also $\mathcal{O}(NM)$ as we are using only two tables of size $N \times M$ (one to keep track of chosen channels and the other to calculate the gains). □

Proposition 5. *Using a tree instead of the matrix, the space complexity can be reduced to $\mathcal{O}(M^2)$ without affecting the time complexity.*

Proof. Since each value of $\mathbf{G}_{n,m,\_}$ is only dependent on the previous row in the dynamic programming approach explain above, then we need to keep only the previous row. As a result the space needed to calculate the optimal gain will be of $\mathcal{O}(M)$. Furthermore, to keep track of the actual set of answers, the tree explained above can be pruned by removing the choices of using a channel in the solution. As a result, the necessary data structure will become a tree with height of at most M, and with at most M nodes in each level of the tree. Such a tree has at most $M \times M$ nodes and can be stored in a matrix of the same size. As a result the total space complexity of the problem will become $\mathcal{O}(M^2)$. □

3.4 An upper bound for the general case

Inspired by Eq. [3.4], we define $\hat{\mathbf{G}}_{n,m,k}$ recursively as:

$$\begin{aligned}\hat{\mathbf{G}}_{n,m,k} = \max\{&\hat{\mathbf{G}}_{n-1,m,k}, \mathcal{R}^b\left(\mathcal{C}_n\right) + (1-\phi\left(\mathcal{C}_n\right))\hat{\mathbf{G}}_{n-1,m-1,k} \\ &+ \phi\left(\mathcal{C}_n\right)\hat{\mathbf{G}}_{n-1,m-1,k-1}\}\end{aligned} \tag{3.7}$$

for $n, m, k \geq 1$, with $\hat{\mathbf{G}}_{n,m,k} = 0$ if $n = 0$, $m = 0$, or $k = 0$. The following proposition shows that $\hat{\mathbf{G}}_{n,m,k}$ is an upper bound on the maximum gain that

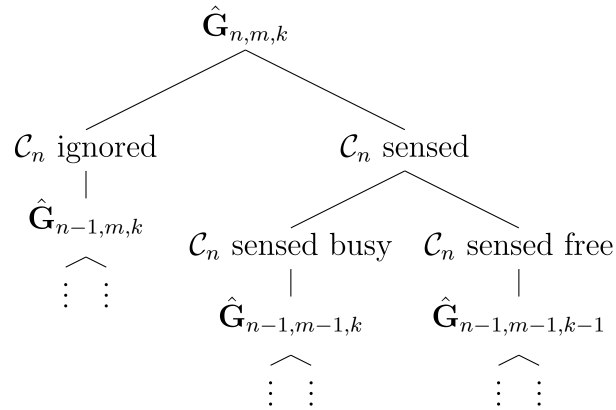

Figure 3.2: In the optimal solution, the two right branches cannot be optimized independently, as a result by relaxing the problem and optimizing them independently, an upper bound is derived.

can be achieved (i.e. $\mathbf{G}_{n,m,k}$).

Proposition 6. *For every positive integers $k \leq m \leq n$, we have*

$$\boldsymbol{G}_{n,m,k} \leq \hat{\boldsymbol{G}}_{n,m,k}.$$

Proof. The proof is by induction on n. For induction basis, we have

$$\forall_{m \geq k \geq 0} \quad \mathbf{G}_{0,m,k} = \hat{\mathbf{G}}_{0,m,k} = 0,$$

thus

$$\forall_{m \geq k \geq 0} \quad \mathbf{G}_{0,m,k} \leq \hat{\mathbf{G}}_{0,m,k}.$$

Now suppose that $\mathbf{G}_{n,m,k} \leq \hat{\mathbf{G}}_{n,m,k}$ for every $m \geq k \geq 0$, and $n = N - 1$, for some positive integer N. We show that $\mathbf{G}_{n,m,k} \leq \hat{\mathbf{G}}_{n,m,k}$ for every $m \geq k \geq 0$, and $n = N$.

By Eq. [3.5] we have

$$\begin{aligned}
\mathbf{H}_{N-1,m-1,k} = & \max_{\substack{S\subseteq\{\mathcal{C}_1,\ldots,\mathcal{C}_{N-1}\}\\|S|=m-1}} \\
& \left((1-\phi\left(\mathcal{C}_N\right))\mathbf{G}_k^S + \phi\left(\mathcal{C}_N\right)\mathbf{G}_{k-1}^S\right) \\
\leq & \max_{\substack{S\subseteq\{\mathcal{C}_1,\ldots,\mathcal{C}_{N-1}\}\\|S|=m-1}} \left((1-\phi\left(\mathcal{C}_N\right))\mathbf{G}_k^S\right) + \\
& \max_{\substack{S\subseteq\{\mathcal{C}_1,\ldots,\mathcal{C}_{N-1}\}\\|S|=m-1}} \left(\phi\left(\mathcal{C}_N\right)\mathbf{G}_{k-1}^S\right) \\
= & (1-\phi\left(\mathcal{C}_N\right)) \max_{\substack{S\subseteq\{\mathcal{C}_1,\ldots,\mathcal{C}_{N-1}\}\\|S|=m-1}} \mathbf{G}_k^S + \\
& \phi\left(\mathcal{C}_N\right) \max_{\substack{S\subseteq\{\mathcal{C}_1,\ldots,\mathcal{C}_{N-1}\}\\|S|=m-1}} \mathbf{G}_{k-1}^S \\
= & (1-\phi\left(\mathcal{C}_N\right))\mathbf{G}_{N-1,m-1,k} + \\
& \phi\left(\mathcal{C}_N\right)\mathbf{G}_{N-1,m-1,k-1}.
\end{aligned}$$

Adding the term $\mathcal{R}^b\left(\mathcal{C}_N\right)$ to both sides of the above inequality, we get

$$\begin{aligned}
\mathcal{R}^b\left(\mathcal{C}_N\right) + \mathbf{H}_{N-1,m-1,k} \leq & \mathcal{R}^b\left(\mathcal{C}_N\right) + \\
& (1-\phi\left(\mathcal{C}_N\right))\mathbf{G}_{N-1,m-1,k} + \\
& \phi\left(\mathcal{C}_N\right)\mathbf{G}_{N-1,m-1,k-1}.
\end{aligned}$$

Therefore, we must have

$$\begin{aligned}
& \max\{\mathbf{G}_{N-1,m,k}, \mathcal{R}^b\left(\mathcal{C}_N\right) + \mathbf{H}_{N-1,m-1,k}\} \\
\leq & \max\{\mathbf{G}_{N-1,m,k}, \mathcal{R}^b\left(\mathcal{C}_N\right) + (1-\phi\left(\mathcal{C}_N\right))\mathbf{G}_{N-1,m-1,k} + \\
& \quad \phi\left(\mathcal{C}_N\right)\mathbf{G}_{N-1,m-1,k-1}\}.
\end{aligned}$$

By induction hypothesis

$$\mathbf{G}_{N-1,m,k} \leq \hat{\mathbf{G}}_{N-1,m,k}$$
$$\mathbf{G}_{N-1,m-1,k} \leq \hat{\mathbf{G}}_{N-1,m-1,k}$$
$$\mathbf{G}_{N-1,m-1,k-1} \leq \hat{\mathbf{G}}_{N-1,m-1,k-1}.$$

Thus

$$\begin{aligned}
&\max\{\mathbf{G}_{N-1,m,k}, \mathcal{R}^b\left(\mathcal{C}_N\right) + \mathbf{H}_{N-1,m-1,k}\} \\
\leq &\max\{\hat{\mathbf{G}}_{N-1,m,k}, \mathcal{R}^b\left(\mathcal{C}_N\right) + \\
&\quad (1-\phi\left(\mathcal{C}_N\right))\hat{\mathbf{G}}_{N-1,m-1,k} + \\
&\quad \phi\left(\mathcal{C}_N\right)\hat{\mathbf{G}}_{N-1,m-1,k-1}\},
\end{aligned}$$

which, by Definitions 3.4 and 3.7, implies

$$\mathbf{G}_{N,m,k} \leq \hat{\mathbf{G}}_{N,m,k},$$

which completes the proof. □

As shown in Figure 3.2, calculating $\hat{\mathbf{G}}_{n,m,k}$ needs the calculation of 3 other values. Since $\hat{\mathbf{G}}_{0,m,k}$, $\hat{\mathbf{G}}_{n,0,k}$ and $\hat{\mathbf{G}}_{n,m,0}$ are all zeroes, a three dimensional dynamic programming in which each cell is calculated using three of its neighbours can be used to calculate the desired value.

Proposition 7. *The running time complexity of the proposed dynamic programming is $\mathcal{O}(NMK)$ and the space complexity is $\mathcal{O}(MK)$.*

Proof. As the recursive Eq. [3.7] has three arguments $1 \leq n \leq N$, $1 \leq m \leq M$ and $1 \leq k \leq K$, there would be at most $N \times M \times K$ possible values for $\hat{\mathbf{G}}_{n,m,k}$. Since every value can be calculated in $\mathcal{O}(1)$ using a dynamic programming approach explained above, then we need $\mathcal{O}(NMK)$ time to calculate the final result. The space complexity is $\mathcal{O}(MK)$ as we are using only a table of size

$M \times K \times 2$ (As we only need the previous level of the $N-1$ to calculate the level of the N and keeping the previous ones is not necessary). $\square$

An interesting, perhaps non-trivial, observation from Figure 3.2 is that the upper bound $\hat{\mathbf{G}}_{n,m,k}$ is achievable in a "sequential sensing" scenario, where m channels are to be sensed sequentially. It is because in "sequential sensing", the result of sensing channel $\mathcal{C}_n$ is known prior to deciding on what channel to sense next. Note that, in this setting, to achieve $\hat{\mathbf{G}}_{n,m,k}$, instead of a set of fixed m channels, a dynamic programming using recursive Eq. [3.7] would suggest a decision tree, with m channels to be sensed on each path from the root to a leaf.

3.5 A Heuristic Algorithm

We call a set of M channels locally-optimal, if the gain achieved by those channels cannot be improved by replacing a channel from the set by one outside of the set. As will be explained shortly, the gain of a fixed set of M channels can be calculated in polynomial-time. Therefore, whether a set of channels is locally-optimal can be tested in polynomial-time. Also, a locally-optimal set can be calculated by starting with an arbitrary set (e.g., the set of channels with M largest blind rewards), and improving it through iterative replacements. We use this procedure to find a locally-optimal set, hence a lower bound on the maximum gain achievable. It is worth mentioning that, in general, the gain of a locally-optimal set can be strictly less than the maximum gain.

3.5.1 Gain Calculation for a Set of M Channels

Suppose that the channels to be sensed are $\mathcal{C}'_1, \ldots, \mathcal{C}'_m$. Let $\mathbf{G}_{\_,m,k}$ denote $\mathbf{G}_{m,m,k}$ to differentiate the general case formula from the special case formula, that is the gain achieved by the set of channels $\{\mathcal{C}'_1, \ldots, \mathcal{C}'_m\}$ when $n = m$.

Corollary 2. *We have*

$$\boldsymbol{G}_{\text{-},m,k} = \mathcal{R}^b\left(\mathcal{C}_m\right) + \left(1-\phi\left(\mathcal{C}_m\right)\right)\boldsymbol{G}_{\text{-},m-1,k} + \phi\left(\mathcal{C}_m\right)\boldsymbol{G}_{\text{-},m-1,k-1}, \tag{3.8}$$

for $m,k \geq 1$*, where* $\boldsymbol{G}_{\text{-},m,k} = 0$ *when* $m = 0$ *or* $k = 0$*.*

Proof. Setting $n = m = M$ in Eq. [3.5], we get

$$\mathbf{H}_{M-1,M-1,k} = \max_{\substack{S\subseteq\{\mathcal{C}_1,\ldots,\mathcal{C}_{M-1}\}\\|S|=M-1}} \{\phi\left(\mathcal{C}_M\right)\mathbf{G}^S_{k-1} + (1-\phi\left(\mathcal{C}_M\right))\mathbf{G}^S_k\}.$$

Since $\{\mathcal{C}_1,\ldots,\mathcal{C}_{M-1}\}$ has only one subset of size $M-1$, we have

$$\mathbf{H}_{M-1,M-1,k} = \phi\left(\mathcal{C}_M\right)\mathbf{G}^S_{k-1} + (1-\phi\left(\mathcal{C}_M\right))\mathbf{G}^S_k,$$

$$S = \{\mathcal{C}_1,\ldots,\mathcal{C}_{M-1}\}.$$

Also, setting $n = m = M-1$ in Eq. [3.3] , we get

$$\mathbf{G}_{M-1,M-1,k} = \max_{\substack{S\subseteq\{\mathcal{C}_1,\ldots,\mathcal{C}_{M-1}\}\\|S|=M-1}} \mathbf{G}^S_k,$$

which can be simplified to

$$\mathbf{G}_{M-1,M-1,k} = \mathbf{G}^S_k,$$

$$S = \{\mathcal{C}_1,\ldots,\mathcal{C}_{M-1}\}.$$

This will give us

$$\mathbf{H}_{M-1,M-1,k} = \phi\left(\mathcal{C}_M\right)\mathbf{G}_{M-1,M-1,k-1} + (1-\phi\left(\mathcal{C}_M\right))\mathbf{G}_{M-1,M-1,k}. \tag{3.9}$$

By Eq. [3.4], we have

$$\mathbf{G}_{M,M,k} = \max\{\mathbf{G}_{M-1,M,k}, \mathcal{R}^b\left(\mathcal{C}_M\right) + \mathbf{H}_{M-1,M-1,k}\}.$$

Therefore, using 3.9, we get

$$\begin{aligned}\mathbf{G}_{M,M,k} = \max\{\mathbf{G}_{M-1,M,k}, &\mathcal{R}^b\left(\mathcal{C}_M\right) + \\ &(1-\phi\left(\mathcal{C}_M\right))\mathbf{G}_{M-1,M-1,k} + \\ &\phi\left(\mathcal{C}_M\right)\mathbf{G}_{M-1,M-1,k-1}\}.\end{aligned}$$

Thus

$$\begin{aligned}\mathbf{G}_{M,M,k} =&\mathcal{R}^b\left(\mathcal{C}_M\right) + (1-\phi\left(\mathcal{C}_M\right))\mathbf{G}_{M-1,M-1,k} + \\ &\phi\left(\mathcal{C}_M\right)\mathbf{G}_{M-1,M-1,k-1},\end{aligned}$$

because $\mathbf{G}_{M-1,M,k} = 0$ by Eq. [3.3], hence

$$\mathbf{G}_{\_,m,k} = \mathcal{R}^b\left(\mathcal{C}_m\right) + (1-\phi\left(\mathcal{C}_m\right))\mathbf{G}_{\_,m-1,k} + \phi\left(\mathcal{C}_m\right)\mathbf{G}_{\_,m-1,k-1}.$$

□

Using the recursive Eq. [3.8], and a dynamic approach similar to the one explained in Section 3.3, the gain $\mathbf{G}_{\_,m,k}$ can be calculated in polynomial time. The space complexity of this dynamic programming will be lower than that of Section 3.3, since here we only need to calculate the gain.

Proposition 8. *The running time complexity of the proposed dynamic programming is $\mathcal{O}(MK)$ and the space complexity is $\mathcal{O}(K)$.*

Proof. As the recursive Eq. [3.8] has two arguments $1 \leq m \leq M$ and $1 \leq k \leq K$, then there would be at most $M \times K$ possible combinations for $\mathbf{G}_{\_,m,k}$. Since every combination can be calculated in $\mathcal{O}(1)$ using a dynamic programming approach explained above, we need $\mathcal{O}(MK)$ time to calculate the final result.

The space complexity is $\mathcal{O}(K)$ as we are only using the values at level $m-1$ to calculate values at level m and it is unnecessary to keep the other levels. □

A case where we can find the optimal solution using the Gain Calculation is when m is very small or close to n, the total number of channel choices becomes small. Since the reward of a set of m channels can be calculated in polynomial time in these cases, the optimal solution becomes computationally plausible.

3.5.2 Local optimal

Algorithm 4 shows a pseudo-code of finding a lower bound on the maximum gain. Initially, the set of M channels with largest blind rewards are selected as the starting set. In each iteration, the algorithm checks all possible replacement, and selects the one making the highest gain improvement. If there is no replacement that improves the gain, then the set is returned as a local optimal.

Algorithm 4 The algorithm is a heuristic solution to the general case of the problem, as well as a lower bound on the optimal solution.

$S \leftarrow$ *set of M channels with largest* $\mathcal{R}_b$
$T \leftarrow$ *the rest of channels*
while not at local optimal **do**
  $MaximumGain \leftarrow Gain(S)$
  for all $s \in S$ **do**
    for all $t \in T$ **do**
      if $Gain((S \setminus s) \cup \{t\}) > MaximumGain$ **then**
        $in \leftarrow t$
        $out \leftarrow s$
      end if
    end for
  end for
  $S \leftarrow (S \setminus out) \cup \{in\}$
end while

The replacement selection in each iteration can be done in polynomial-time using the gain calculation method explained earlier. Simulations show that the number of iterations needed to get to the local optimal is typically very small (Less than 8 for $N = 32$).

Chapter 4

Simulations

All simulations use uniformly random numbers in the ranges specified, created using C++ random library's *default_random_engine*. Each data point was obtained by averaging the results of over one million simulation runs.

4.1 Effects of K

The first simulation, whose results are shown in Figure 4.1, tests and confirms Proposition 1. The simulation's parameters are given in Table 4.1. In Table 4.1, the parameter Γ is

$$\Gamma = \max_{1 \leq i,j \leq N} \frac{B\left(\mathcal{C}_i\right)}{B\left(\mathcal{C}_j\right)}.$$

Therefore, $\Gamma = 1$ implies that $B\left(\mathcal{C}_i\right) = B\left(\mathcal{C}_j\right)$, for every $1 \leq i, j \leq N$. As shown in Figure 4.1, the ratio of the gain of the non-intuitive algorithm over the gain of the intuitive algorithm increases at least linearly as M increases. This ratio for a given M is maximized for the case $K = 1$. Fortunately, for this case we proposed an optimal polynomial time solution. For larger values of K, we can use the proposed heuristic algorithm. The simulation results show that, our heuristic algorithm performance matches that of the non-intuitive solution for the scenario parametrized by Table 4.1.

	Figure 4.1		Figure 4.2	
	Set S_1	Set S_2	Set S_1	Set S_2
θ	$\frac{1}{M}$		$\frac{1}{M}$	$\frac{1}{M.B}$
μ	0.1	0	0.1	0
Γ	1		1	$\{1, 2, 3, \cdots, 6\}$
K	$\{1, 2, 3, \cdots, 6\}$		1	
M	$\{1, 2, 3, \cdots, 99\}$			
α	0			

Table 4.1: In the simulations, we suppose $|S_1| = |S_2| = M$. The Intuitive algorithm will always choose the channels in set S_1, while the non-intuitive algorithm will always choose the channels in set S_2.

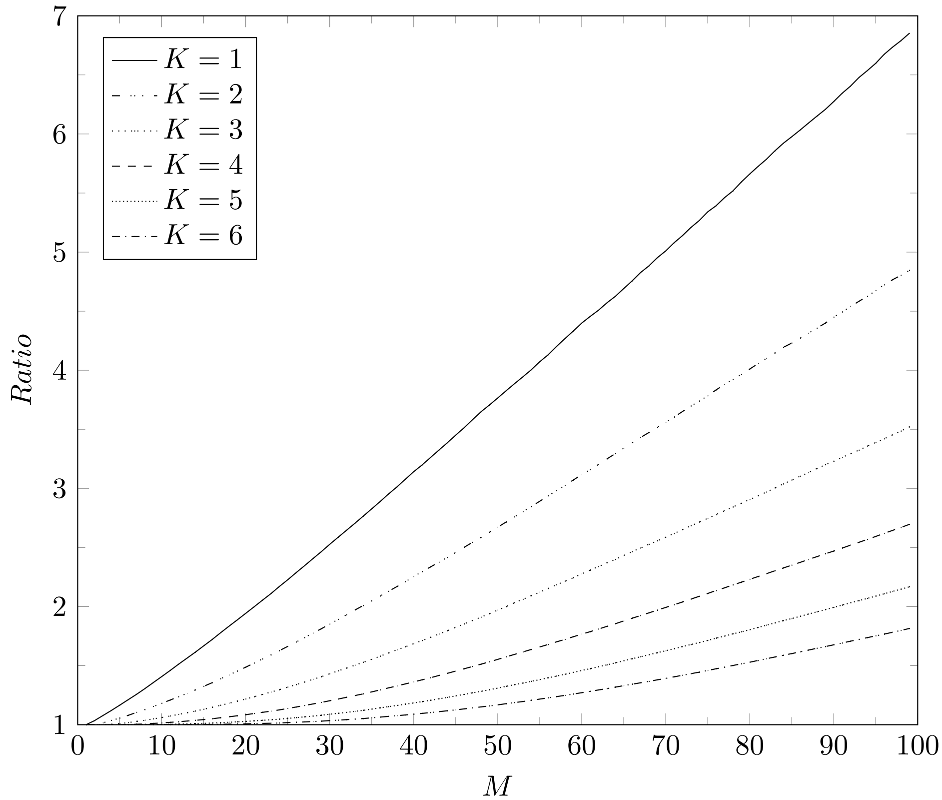

Figure 4.1: The maximum improvement in the optimal solution over the intuitive solution for different values of K over $1 \leq M \leq 99$ and channels of equal bandwidth ($\Gamma = 1$). For large values of M, the ratio is almost linear.

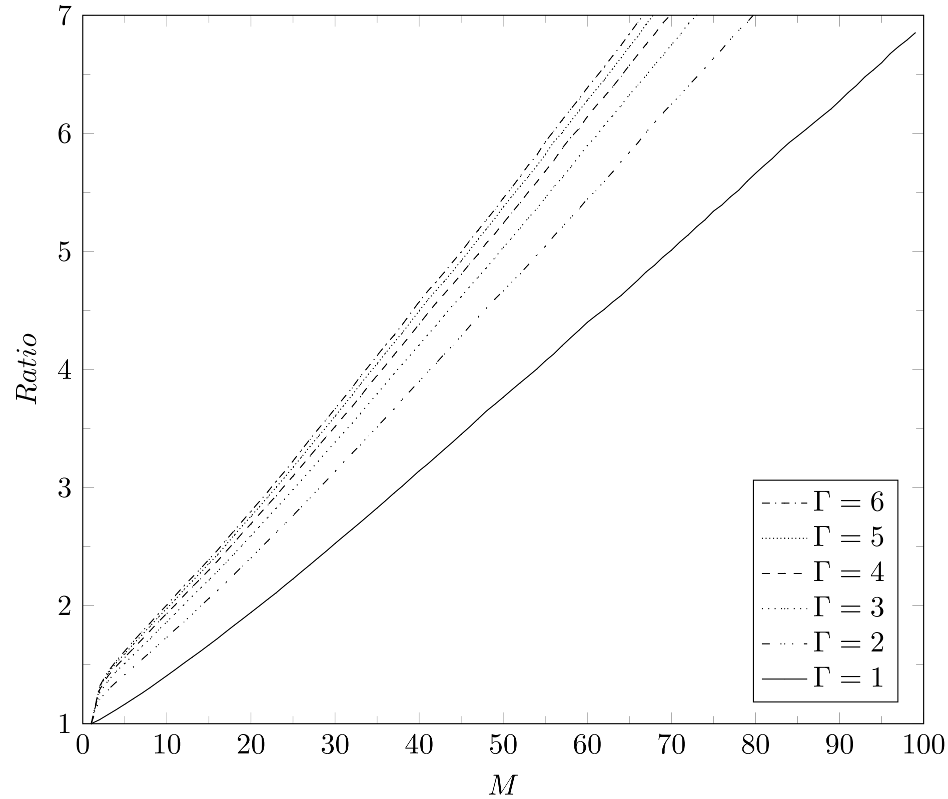

Figure 4.2: The maximum improvement in the optimal solution over the intuitive solution for different values of Γ over $1 \leq M \leq 99$ and $K = 1$. For large values of M, the ratio is almost linear.

4.2 Effects of Γ

In the second simulation, we test the effect of having channels with different values of $B(\mathcal{C})$ (i.e., larger values of Γ) on the gap between the performance of the intuitive and the non-intuitive algorithms. The parameters used in this simulation are given in Table 4.1. As shown in Figure 4.2, for the given $K = 1$, M, and N, the performance gap ratio increases with Γ.

4.3 The Heuristic Algorithm

Finally, in our third simulation, in an average case scenario, we compare the performance of our proposed heuristic algorithm (Algorithm 4) to the upper

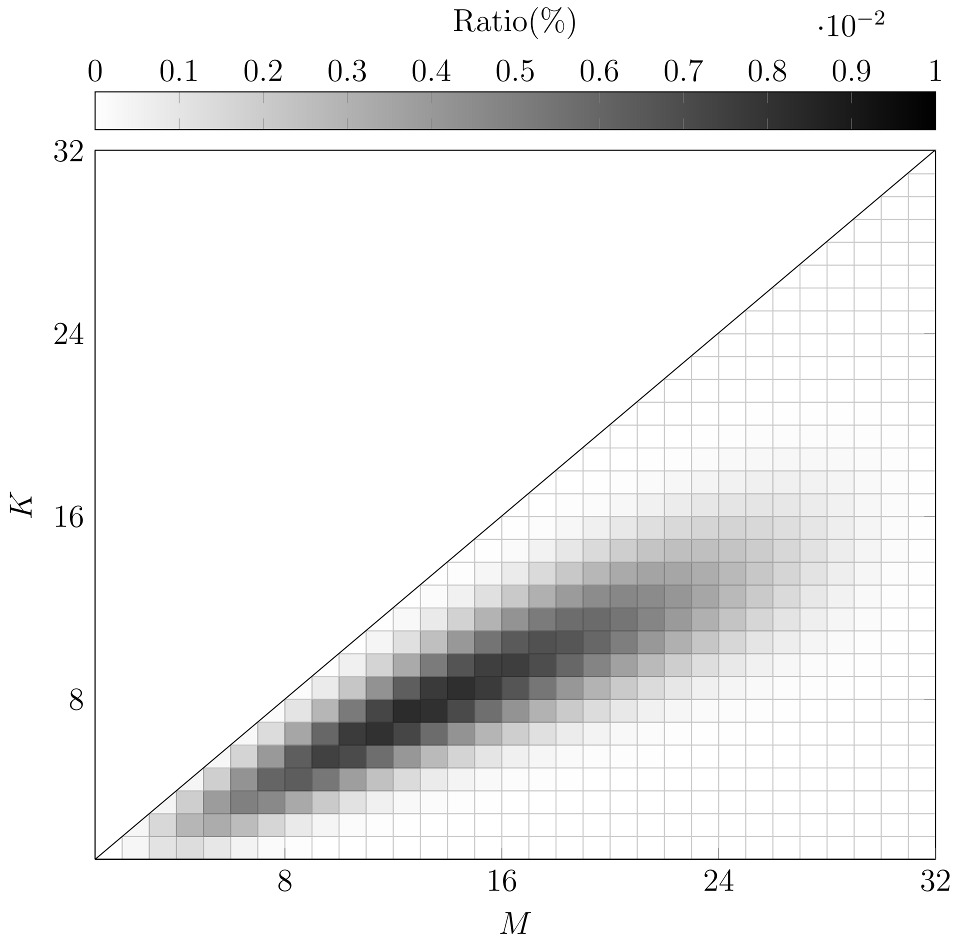

Figure 4.3: The average improvement in the ratio of upper bound $\hat{\mathbf{G}}_{N,M,K}$ over the heuristic algorithm for $N = 32$ and $1 \leq M, K \leq 32$.

bound given in Section 3.4. In this simulation, for every pair $1 \leq M, K \leq 32$, we set $\Gamma = 1$, and $N = 32$, and choose $\alpha(\mathcal{C})$ and $\mu(\mathcal{C})$ uniformly at random from $[0, 0.1]$, and $\theta(\mathcal{C})$ uniformly at random from $[0, 1]$. As shown in the heat map presented in Figure 4.3, the gain of the heuristic algorithm is at most 0.01% lower than the upper bound. Hence, for the average case scenario considered in the third simulation, the heuristic algorithm virtually is as good as the optimal solution.

4.4 Previous Works

In our last simulation, we use the test case presented by Zhou Zhang et al. in their work [2] to validate our solution for the special case $K = 1$. Table 4.2 has the values they used for their example. Figure 4.4 shows the gain achieved for all possible choices in the example. The intuitive solution presented by them produces the optimal solution only for the *Homogeneous* input. However as it is shown in the figure, our solution creates an optimal answer for all three examples.

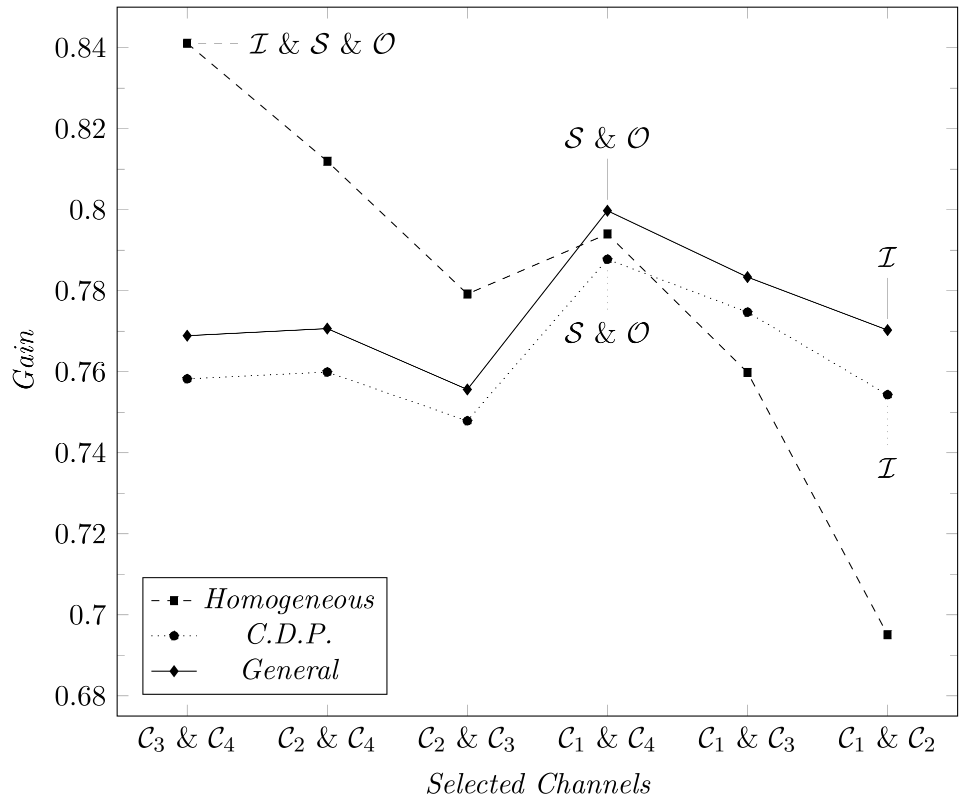

Figure 4.4: This graph is taken from the work of Zhou Zhang et al. [2] analyzing the three cases *Homogeneous*, *Common Detection Probability (C.D.P.)*, and *General*. The graph shows the achieved gain for selecting each pair of channels from the data presented in Table 4.2 and ($N = 4$, $M = 2$, $K = 1$). We use this graph to validate the results of our solution for the case of $K = 1$. The Intuitive Result, Our Result and the Optimal Result have been labelled with $\mathcal{I}$, $\mathcal{S}$, and $\mathcal{O}$ respectively. As it can be seen in the graph, in all three cases our solution produces the optimal solution, however, the intuitive solution only produces an optimal solution for the "Homogeneous" case.

<table>
<tr><th colspan="2">Channels</th><th>Homogeneous</th><th>C.D.P.</th><th>General</th></tr>
<tr><td rowspan="3">$\mathcal{C}_1$</td><td>$\theta\left(\mathcal{C}_1\right)$</td><td colspan="3">0.650</td></tr>
<tr><td>$\alpha\left(\mathcal{C}_1\right)$</td><td>0.3</td><td colspan="2">0.1</td></tr>
<tr><td>$\mu\left(\mathcal{C}_1\right)$</td><td colspan="2">0.3</td><td>0.2</td></tr>
<tr><td rowspan="3">$\mathcal{C}_2$</td><td>$\theta\left(\mathcal{C}_2\right)$</td><td colspan="3">0.727</td></tr>
<tr><td>$\alpha\left(\mathcal{C}_2\right)$</td><td>0.3</td><td colspan="2">0.28</td></tr>
<tr><td>$\mu\left(\mathcal{C}_2\right)$</td><td colspan="2">0.3</td><td>0.2</td></tr>
<tr><td rowspan="3">$\mathcal{C}_3$</td><td>$\theta\left(\mathcal{C}_3\right)$</td><td colspan="3">0.852</td></tr>
<tr><td>$\alpha\left(\mathcal{C}_3\right)$</td><td>0.3</td><td colspan="2">0.39</td></tr>
<tr><td>$\mu\left(\mathcal{C}_3\right)$</td><td colspan="2">0.3</td><td>0.2</td></tr>
<tr><td rowspan="3">$\mathcal{C}_4$</td><td>$\theta\left(\mathcal{C}_4\right)$</td><td colspan="3">0.918</td></tr>
<tr><td>$\alpha\left(\mathcal{C}_4\right)$</td><td>0.3</td><td colspan="2">0.43</td></tr>
<tr><td>$\mu\left(\mathcal{C}_4\right)$</td><td colspan="2">0.3</td><td>0.05</td></tr>
</table>

Table 4.2: The values of the example used by Zhou Zhang et al. [2] for the three cases *Homogeneous*, *Common Detection Probability (C.D.P.)*, and *General*.

Chapter 5

Conclusion & Future Works

We showed that the intuitive solution can perform much worse than the optimal solution, in the worst case scenario. The performance gap between the two algorithms is maximized when $K = 1$, that is when only one channel is needed for access. For this case, we proposed an optimal polynomial-time algorithm. In addition, for larger values of K, we proposed a heuristic polynomial time algorithm. The simulation results show that the performance of our proposed heuristic algorithm matches that of the non-intuitive algorithm in the worst case scenario considered. Also, its performance is virtually the same as the that of the optimal solution in the average case scenario considered.

An interesting extension to this work is to show whether or not there is a polynomial-time optimal algorithm for $K > 1$. Another extension is to consider a multiple secondary user scenario, where users compete over accessing channels.

Bibliography

[1] Joseph Mitola III and Gerald Q Maguire Jr. Cognitive radio: making software radios more personal. *Personal Communications, IEEE*, 6(4): 13–18, 1999.

[2] Zhou Zhang and Hai Jiang. Cognitive radio with imperfect spectrum sensing: The optimal set of channels to sense. *Wireless Communications Letters, IEEE*, 1(2):133–136, 2012.

[3] Ying-Chang Liang, Yonghong Zeng, Edward CY Peh, and Anh Tuan Hoang. Sensing-throughput tradeoff for cognitive radio networks. *Wireless Communications, IEEE Transactions on*, 7(4):1326–1337, 2008.

[4] Hai Jiang, Lifeng Lai, Rongfei Fan, and H Vincent Poor. Optimal selection of channel sensing order in cognitive radio. *Wireless Communications, IEEE Transactions on*, 8(1):297–307, 2009.

[5] Lifeng Lai, Hesham El Gamal, Hai Jiang, and H Vincent Poor. Cognitive medium access: Exploration, exploitation, and competition. *Mobile Computing, IEEE Transactions on*, 10(2):239–253, 2011.

[6] Ho Ting Cheng and Weihua Zhuang. Simple channel sensing order in cognitive radio networks. *Selected Areas in Communications, IEEE Journal on*, 29(4):676–688, 2011.

[7] Ian F. Akyildiz, Won-Yeol Lee, Mehmet C. Vuran, and Shantidev Mohanty. Next generation/dynamic spectrum access/cognitive radio wire-

less networks: A survey. *Comput. Netw.*, 50(13):2127–2159, September 2006. ISSN 1389-1286. doi: 10.1016/j.comnet.2006.05.001. URL `http://dx.doi.org/10.1016/j.comnet.2006.05.001`.

[8] Gregory Staple and Kevin Werbach. The end of spectrum scarcity [spectrum allocation and utilization]. *Spectrum, IEEE*, 41(3):48–52, 2004.

[9] Xi Fang, Dejun Yang, and Guoliang Xue. Taming wheel of fortune in the air: An algorithmic framework for channel selection strategy in cognitive radio networks. *Vehicular Technology, IEEE Transactions on*, 62(2):783–796, 2013.

[10] Zhou Zhang, Hai Jiang, Peng Tan, and Jim Slevinsky. Channel exploration and exploitation with imperfect spectrum sensing in cognitive radio networks. *Selected Areas in Communications, IEEE Journal on*, 31(3):429–441, 2013.

[11] Lei Li, Wenzhong Zhang, Sihai Zhang, Ming Zhao, and Wuyang Zhou. Energy-efficient channel aggregation in cognitive radio networks with imperfect sensing. In *Wireless Communications and Networking Conference (WCNC), 2013 IEEE*, pages 2817–2822. IEEE, 2013.

[12] Hiteshi Sharma, Aaqib Patel, SN Merchant, and UB Desai. Optimal spectrum sensing for cognitive radio with imperfect detector.

[13] Hossein Shokri-Ghadikolaei and Carlo Fischione. Analysis and optimization of distributed random sensing order in cognitive radio networks. *arXiv preprint arXiv:1401.1294*, 2014.