LOW-COMPLEXITY LIST DETECTION ALGORITHMS FOR THE MULTIPLE-INPUT MULTIPLE-OUTPUT CHANNEL

A Dissertation
Presented to
The Academic Faculty

by

David L. Milliner

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
December 2009

LOW-COMPLEXITY LIST DETECTION ALGORITHMS FOR THE MULTIPLE-INPUT MULTIPLE-OUTPUT CHANNEL

Approved by:

Professor John R. Barry, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Xiaoli Ma
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Gordon Stüber
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Alfred Andres
School of Mathematics
Georgia Institute of Technology

Professor Ye (Geoffrey) Li
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: 14th of October 2009

ACKNOWLEDGEMENTS

This thesis is the result of my work in the Communication Theory Research Group at the Georgia Institute of Technology in Atlanta Georgia, the Communications Systems Laboratory at Texas Instruments in Dallas Texas and the Vodafone Chair Mobile Communications Systems group in Dresden Germany. First and foremost I would like to express my profound gratitude towards my advisor John Barry. I have been fortunate to have such a dedicated and insightful advisor who has always been able to provide valuable feedback concerning my endeavors. I will always be thankful for his guidance and advice.

In the Spring of 2008 I was a Guest Scientist in the Vodafone Mobile Communications Systems Chair at the Technische Universität Dresden. This opportunity was available to me thanks largely to Ernesto Zimmermann. Ernesto is a brilliant mind and has contributed significantly to the content of this thesis in many important areas. In particular Ernesto has contributed heavily to the work in this dissertation on smart candidate adding and LLR clipping. I learned much from collaborating with Ernesto and appreciate his generous allocation of time to think about MIMO detection and other mutually interesting topics, technical or otherwise. My gratitude extends to all the members of the Vodfaone Chair who made my stay in Dresden so fulfilling. In particular I would like to mention Steffen Bittner, Marco Krondorf, and Gerhard Fettweis.

Many current and former members of the Communication Theory Research Group have contributed to my graduate study efforts. In particular I would like to thank Anuj Batra. I first met Anuj while interning at Texas Instruments and since that time he has become an incredible mentor and close friend. His mentorship has contributed significantly to the work in this dissertation, in particular the work in chapter 6 on nonuniform computational complexity allocation. I would also like to recognize Deric Waters for his contributions

pertaining to the work in this dissertation on B-Chase ordering and nonuniform computational allocation. Deric has always been available to discuss research topics with me and his ideas have always spurred interesting explorations. Thanks are also in order to Mohanned Sinnokrot for his contributions pertaining to the work in this dissertation on soft-output detection of the golden code.

Deep thanks are in order to Texas Instruments, and in particular the members of the Communication Systems Laboratory. I am grateful to TI for many reasons including providing funding for significant portions of my research, opportunities to intern and mentorship in many important areas of development over the years. In particular I would like to recognize those who have been particularly involved with my Ph.D. research: Don Shaver, Srinath Hosur, Bob Hewes, Deric Waters and Anuj Batra. In addition to the contributions of Deric and Anuj mentioned previously, I would like to express my appreciation for the sincere interest Don, Sri and Bob who have always shown a sincere interest in my endeavors.

I would also like to thank the members of my dissertation defense committee: Gordon Stüber, Ye (Geoffrey) Li, Xiaoli Ma, and Alfred Andrew. There are also several people who have aided me tremendously in areas concerning the logistics of my graduate studies. In particular I would like to thank Cordai Farrar for her help during the countless times I have sought her assistance. I would also like to thank Josyane Roschitz and Florence Stoia for handling the myriad of travel issues relating to my studies at the Georgia Tech Lorraine campus in Metz, France. I would also like to thank Marilou Mycko for the coordination of my ECE graudate studies and Pat Dixon for all of her prompt attention to many other administrative issues.

Last but not least I would like to thank my family and friends for their support and shared experiences. In particular I would like to thank my roommate Erich Stuntebeck for all the fun times outside the office. I would also like to thank my new friends made at

Georgia Tech Atlanta, Georgia Tech Lorraine and TU Dresden during my time as a graduate student. The memories and opportunities each of you has provided to me will last a lifetime and in many cases have formed lifelong friendships. Some of these friends I shared office space with me for extended periods of time and I would like to thank Matthieu Bloch, Arumugam Kannan, Demijan Klinc, Arunkumar Subramanian, Jiaxi Xiao, and Willie Harrison for the fond memories and pleasant working environment. Finally, as only a small recognition for their unwavering love and support I would like to dedicate this thesis to my Mom and Dad.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Motivation and Research Focus

The thirst of today's society for communication systems supporting ever-increasing data rates remains unquenched. This thirst originates from user growth and increasingly data-intensive applications such as next-generation voice and video communications, high resolution imaging, mobile data distribution, and network backhauling. Quenching the global thirst for high-data-rate communication systems is not easy, particularly in the case of wireless communications, where regulation and fundamental physical constraints manifest the scarcity of radio frequency spectrum. Because frequency spectrum is a limited natural resource, utilizing this resource efficiently is paramount to meeting the demands of current and future communication systems.

A way to intelligently utilize radio frequency spectrum is through the use of antenna arrays. Antenna arrays can be used to increase the radiated power in a desired direction (i.e. beamsteering), combat fading environments, and utilize radio frequency spectrum efficiently. The differing antennas in an array may be physically co-located or distributed. In the case of a distributed antenna array, cooperation is typically required amongst the separate locations. In this dissertation we focus on receiver design for high data rate communication systems utilizing physically co-located arrays, although the principles presented may be applied to distributed networks.

When antenna arrays are found at both the transmitter and the receiver, i.e. more than one antenna at both locations, a multiple-input multiple-output (MIMO) channel is formed. This MIMO channel allows for communication systems to exploit spatial diversity for data

rate and/or reliability gains. These gains are most significant in rich-scattering environments where the MIMO channel is uncorrelated (i.e. independent fading at both the transmitter and receiver) and of high rank.

The potential data rates of MIMO communication systems are higher than single-input single-output (SISO) systems. Specifically, in a rich-scattering propagation environment, the upper bound on the amount of information that can be reliably transmitted over a communications channel, i.e. the capacity, increases linearly with the minimum of the number of transmit and receive antennas. Schemes that exploit the spatial dimension to transmit multiple data streams simultaneously over each of the transmit antennas are known as spatial multiplexing schemes.

In addition to increased capacity, MIMO channels enable greater reliability due to the fact that an increase in the number of propagation paths between the transmitter and receiver decreases the probability that all paths will be subject to a deep signal fade simultaneously. Schemes that exploit the spatial dimension to provide the receiver with multiple copies of the same message, thereby increasing the communication system's robustness, are known as spatial diversity schemes.

Examples of spatial multiplexing schemes include bit-interleaved coded modulation (BICM) [21] and the Bell Labs Layered Space-Time (BLAST) transmission scheme [33]. Examples of spatial diversity schemes include space-time block codes [2, 89] and antenna beamforming [36]. It is also possible to exploit both the spatial diversity and the spatial multiplexing capabilities of the MIMO channel. Such schemes are sometimes referred to as diversity-multiplexing schemes [108]. Algebraic space-time codes can be considered diversity-multiplexing schemes. A particularly important algebraic space-time code considered in this work is the golden code [13, 27].

In this dissertation we constrain ourselves to communication systems obtaining high data rates over a given frequency band, i.e. high spectral efficiency, where the transmitter complexity is low and the receiver design is a challenge. Specifically, in this dissertation

we elect to focus on multiantenna communication systems transmitting at least 8 bits per second per Hertz (and often much higher spectral efficiencies). Due to system constraints we will impose such as no channel state information at the transmitter and moderate to large transmission alphabets, the multiantenna receiver design is a challenging problem. This dissertation presents detection algorithms for receivers in a multiantenna communication system that simultaneously possess low computational complexity and achieve good error rate performance. In particular we will focus on receivers for BICM spatial multiplexed systems and diversity-multiplexing systems employing the golden code.

Error-control coding is an essential technique for both approaching the capacity in a communication system and ensuring reliable communication for the same. Like any communication system, a MIMO system relies on error-control coding. And while error-control coding is relatively easy to implement at the transmitter, even for a MIMO system, the problem of error-control decoding at the receiver is greatly complicated by the presence of a MIMO channel. In fact, an optimal receiver that *jointly* accounts for the mutual interference of a MIMO channel and error-control coding is completely intractable for any realistic code and channel. For this reason, a practical MIMO receiver will account for the constraints introduced by the channel code and the mutual interference introduced by the MIMO channel separately using first a MIMO detector, which effectively ignores the presence of the code by assuming that the code bits are independent and uniformly distributed, and then an error-control decoder.

Ignoring the presence of coding is tantamount to assuming that the coded bits are independent and equally likely to be 0 or 1. In this setting, the best the MIMO detector can do is to compute the *a posteriori probability (APP)* for each of the coded bits, which is the conditional probability that each coded bit is 1 (or 0) given the observation of the channel output.

A detector that produces binary decisions about the coded bits by comparing each APP

to a threshold of one-half is called a *hard-output* detector; anything else is called a *soft-output* detector. A detector that quantizes each APP to two or more bits of precision is an example of soft-output detector, while the exact APP detector is the ultimate soft-output detector.

The literature on hard-output MIMO detection is vast. The optimal hard-output detector, known as the joint maximum likelihood (JML) detector, finds the best symbol vector from all possible transmission vectors and has a worst-case computational complexity that grows exponentially in the number of transmitters. Suboptimal hard-output detection algorithms exist, but they require increased power at the transmitter to achieve the same error-rate performance as the JML detector. Consequently, there is a trade-off between the error-rate performance and the required computational complexity in MIMO detection. This same trade-off exists for the more challenging problem of soft-output detection, where, in the presence of error-control coding and for the same signal-to-noise ratio, lower error rates than those obtained via hard-output detection are achieved.

The problem of soft-output detection, which aims to compute or approximate the exact APPs, is important for two fundamental reasons: First, the performance of the error-control decoder depends critically on how well its inputs approximate the true APPs, and second, the high complexity of soft-output detection can easily dominate the other receiver tasks such as error-control decoding. This is because the complexity of exact APP computation grows exponentially with both the number of transmit antennas and the number of bits per transmitted symbol, and is prohibitively complex even for MIMO systems with moderately small antenna arrays and transmission alphabet sizes. Consequently, the soft-output detector is often the critical determining factor in both the performance and the complexity of the overall system. The significant impact of soft detection on the performance and complexity of MIMO systems makes an efficient and accurate soft-output detector essential for modern communication systems.

While many efficient soft-output detection algorithms with low-error-rate performance

exist [47, 42, 87], gaps in the research on this important topic remain. In particular, there is a lack of soft-output detection algorithms that approach the error-rate performance of the exact APP detector with *low and fixed* computational complexity. This dissertation constructs soft-output detection algorithms to fill this gap. Other gaps in the literature that we fill in this dissertation pertain to the analysis of computational complexity in relation to list-based soft-output detection, near exact APP detection of the golden code with low and fixed computational complexity, and the use of channel state information to constrain approximated APPs. Additionally, we apply list detection to the design of hard-output receivers for multiantenna orthogonal frequency-division multiplexing (MIMO-OFDM) channels.

We use the remainder of this chapter to present the structure of and notation for the rest of this dissertation.

1.2 Dissertation Outline

- Chapter 2 serves as a summary of prior art on list detection algorithms for the MIMO channel. The chapter begins by introducing the MIMO channel model, followed by the coded system model used throughout this dissertation. We then provide an introduction to the soft-output MIMO detection problem, with a focus on tree-based detection algorithms. The chapter then provides a baseline understanding of classical and state of the art approaches for tackling the soft-output MIMO detection problem. In section 2.5 we introduce the notion of computational complexity, provide metrics for computational complexity, and establish bounds and non-reference computational complexities for tree-based MIMO detection. Next, we describe the significant impact on system error-rate performance and/or computational complexity resulting from mapping layers in the detection tree to transmitted symbols. We then conclude the chapter with a general discussion of soft-output MIMO detection algorithms and provide suggestions for further reading.

  Portions of section 2.5 represent original and collaborative contributions (see [69]),

while the rest of chapter 2 describes prior art. The remainder of the dissertation represents original contributions, with the aid of collaborators as noted in the acknowledgements.

- Chapter 3 proposes a soft-output MIMO detection algorithm called the smart-ordered candidate-adding (SOCA) algorithm. The algorithm is motivated and connected to prior breadth-first detection algorithms. Then, after presenting the proposed algorithm, a framework for fixed computational complexity breadth-first MIMO detection algorithms is provided. The chapter concludes with a collection or performance versus computational complexity results.

- Chapter 4 presents a low- and fixed- computational complexity soft-output detector for an important algebraic space-time code known as the golden code.

- The value at which the log-likelihood ratio (LLR) of conditional probabilities for a transmitted bit being either a 1 or a 0 is *clipped* has an impact on the overall system performance. Chapter 4 proposes a new approach for determining an LLR *clipping* level to improve the overall system performance. In contrast to prior LLR clipping approach which employ a predetermined fixed LLR clipping level, the contribution in this chapter is an LLR clipping algorithm that exploits channel state information to improve the system performance of suboptimal list MIMO detection algorithms.

- Orthogonal frequency-division multiplexing is an effective technique for combatting frequency-selective wideband communication channels. It is common practice for MIMO-OFDM detectors to implement the same detector at each subcarrier, in which case the overall performance is dominated by the weakest subcarrier. In chapter 6 we propose a hard-output list detection receiver strategy for MIMO-OFDM channels called *nonuniform computational complexity allocation (NCCA)*, whereby the receiver adapts the computational resources of the MIMO detector at each subcarrier

to match a metric of the corresponding channel quality. The proposed nonuniform algorithm is shown to improve performance over uniform allocation.

- A summary of this dissertation, suggestions for future work, and concluding remarks are provided in chapter 7.

1.3 Notation

We now briefly provide the necessary notation. Matrices are set in boldface capital letters and vectors in boldface lowercase letters. We denote the entry in the i^{th} row and v^{th} column of the matrix $\mathbf{R}$ as R_{iv}, the v^{th} column of $\mathbf{R}$ as $\mathbf{R}_v$ and the i^{th} entry of the vector $\mathbf{b} = [b_1\ b_2\ \ldots\ b_N]^T$ as b_i.

$\mathbf{h}^{\mathrm{T}}$, $\mathbf{H}^{\mathrm{T}}$	The transpose of a vector $\mathbf{h}$, or a matrix, $\mathbf{H}$.
$\mathbf{h}^*$, $\mathbf{H}^*$	The conjugate transpose of a vector $\mathbf{h}$, or a matrix, $\mathbf{H}$.
$\mathbf{H}^\dagger$	The Moore-Penrose pseudoinverse of $\mathbf{H}$.
$\|\|\mathbf{h}\|\|$	The Euclidian norm of a vector $\mathbf{h}$, i.e. $\|\|\mathbf{h}\|\| = \sqrt{\mathbf{h}^*\mathbf{h}}$.
$\|\mathcal{A}\|$	The cardinality or size of a finite set $\mathcal{A}$, i.e. the number of elements in $\mathcal{A}$.
$\mathbb{R}$, $\mathbb{C}$, $\mathbb{Z}$	The sets of real, complex and integer numbers.
$\Pr(x)$	Probability of x.
$\Pr(x\|y)$	Probability of x given y.
$\hat{a}$, $\hat{\mathbf{a}}$	Estimate of the scalar, a, or vector $\mathbf{a}$.
$[\cdot]$	Component-wise rounding to $\mathbb{Z}$.
$[\cdot]_{\mathcal{A}}$	Component-wise rounding to the nearest element of $\mathcal{A}$.
$\mathcal{N}(\mu,N_0)$	Gaussian distribution with mean μ and variance N_0.
$\ln$	Natural logarithm.
$\log_b$	Base b logarithm.

$\mathbf{I}_N$ and $\mathbf{0}_N$	$N \times N$ identity and zero matrices, respectively.
$\mathbb{E}[\cdot]$	The expectation operator.

CHAPTER 2

CLASSICAL AND STATE-OF-THE ART LIST MIMO DETECTION

2.1 Introduction

Multiantenna communication systems are capable of increased data rates and improved reliability relative to single antenna systems. Like any communication receiver, the goal of a MIMO receiver is error free recovery of the transmit sequence. An essential component of a receiver is the detector, whose job is the extraction of information concerning the input data sequence in the absence of cooperation with the sender. Multiantenna detection is the central theme of this dissertation.

A detector for which the extracted information is strictly a decision of 0 or 1 for each of the transmitted bits is known as a hard-output detector. Any detector that goes beyond a simple decision of 0 or 1 for a transmitted bit to express the detector's confidence in a given decision is known as a soft-output detector. In the case where the detector assumes a bit sequence is independent and equally likely to be 0 or 1, the best the MIMO detector can do is to compute the *a posteriori probability* for each of the coded bits, which is the conditional probability that each coded bit is 1 (or 0) given the observation of the channel output.

This dissertation's primary objective is the design of multiantenna detectors whose performance is as close as possible to the exact a posteriori probability detector with *low and fixed* computational complexity. In order to achieve this objective there are many important concepts and prior art which must be presented in order to establish the foundation for the original contributions subsequently presented. Laying this foundation is the function of the current chapter. Specifically, the purpose of this chapter is to summarize classical and state-of-the-art solutions to the MIMO detection problem, with a focus on soft-output

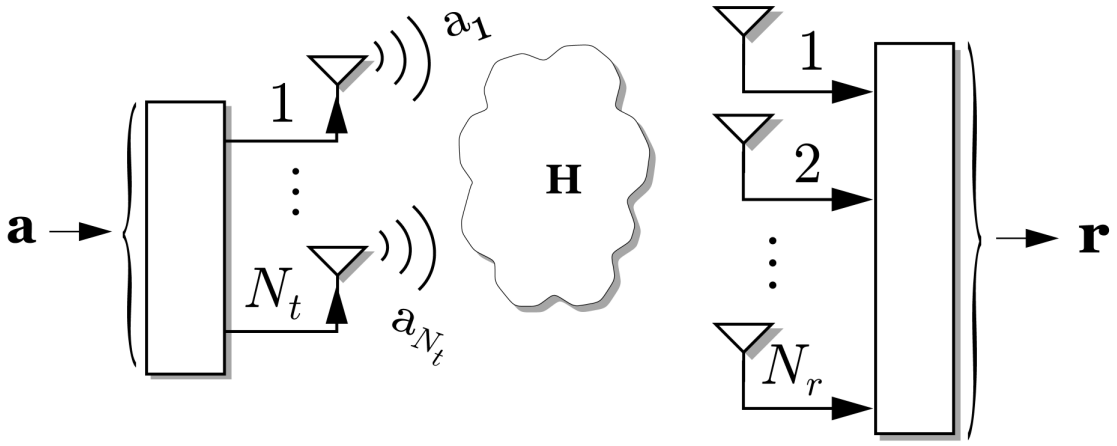

Figure 2.1: A MIMO channel.

detection.

The first step in laying the foundation upon which the contributions in this dissertation are based is to establish the multiantenna channel model and overall coded system model. With sufficient understanding of the system setup we are ready to tackle the problem of MIMO detection. In this dissertation we will almost exclusively solve the MIMO detection problem using a construct known as a detection tree. The detection tree is formulated using the MIMO channel and, with the addition of the received signal, can then be searched thereby facilitating detection. Prior art pertaining to tree-based detection comprises much of the content in this chapter. We also use this chapter to establish benchmarks on computational complexity so that in future chapters we are able to assess the contribution of our proposed detection algorithms relative to prior art. Other topics in this chapter include receiver preprocessing algorithms and tree-based search enumeration algorithms. We conclude the chapter with a discussion of MIMO detectors that fall outside the scope of this dissertation and provide suggestions for further reading.

2.2 MIMO Channel Model

The N_t-input N_r-output memoryless multiantenna channel model used in this dissertation is depicted in Fig. 2.1. The linear and complex-valued baseband model may be expressed

as:

$$\mathbf{r} = \mathbf{H}\mathbf{a} + \mathbf{w}, \tag{2.1}$$

where $\mathbf{a} \in \mathbb{C}^{[N_t \times 1]}$ is the vector of transmitted symbols, one for each transmit antenna, $\mathbf{r} \in \mathbb{C}^{[N_r \times 1]}$ is the vector of received samples, one for each receive antenna, $\mathbf{w} \in \mathbb{C}^{[N_r \times 1]}$ is additive noise, and $\mathbf{H} \in \mathbb{C}^{[N_r \times N_t]}$ is the channel matrix.

We assume a single-carrier narrowband flat-fading channel. Typical broadband communication channels are frequency-selective. A common solution for overcoming frequency selective channels is the use of orthogonal frequency division multiplexing (OFDM) [5]. The use of OFDM transforms a frequency-selective MIMO channel into a parallel bank of flat-fading MIMO channels, one for each subcarrier. This transformation enables the use of a memoryless single-carrier MIMO detection algorithm (i.e., one designed for a flat-fading channel) by simply applying the same detection algorithm at each subcarrier. Consequently, MIMO detection algorithms designed for transmission over narrowband flat-fading single-carrier MIMO channels may be applied to broadband frequency-selective channels and our assumption is justified.

Throughout this dissertation we assume additive white Gaussian noise (AWGN), so that the components of $\mathbf{w}$ are zero-mean, circularly symmetric, independent and identically-distributed (i.i.d.) complex Gaussian random variables with variance N_0, so that $\mathcal{E}[\mathbf{w}\mathbf{w}^*] = N_0\mathbf{I}$, where $\mathbf{w}^*$ denotes the complex conjugate of $\mathbf{w}$. The entry in the i^{th} row and v^{th} column of $\mathbf{H}$ represents the complex channel gain between transmit antenna v and receive antenna i and is normalized such that $\mathbb{E}[|H_{i,v}|] = 1$. The SNR at any receive antenna is SNR $= E/N_0$. Throughout this dissertation we consider Rayleigh fading, typical of non-line-of-sight communication systems, so that the entries of $\mathbf{H}$ are i.i.d. complex Gaussian random variables. The adoption of a non-line-of-sight propagation environment is necessary to support spatial multiplexing at the transmitter, because sufficient scattering is required for this transmission scheme at practical signal-to-noise ratios (SNRs), e.g. less than 30 dB.

Throughout this dissertation the entries of $\mathbf{a}$ are chosen from the same complex q-ary quadrature amplitude modulation (QAM) alphabet $\mathcal{A}$ with energy E/N_t, where $q = |\mathcal{A}|$. Additionally, we will exclusively focus on the scenario where $N_r \geq N_t$. This is justified by the fact they were are interested in exploiting the spatial multiplexing gain made possible by the MIMO channel, where the multiplexing gain is defined as $\min\{N_t, N_r\}$. It is therefore imprudent to increase the number of transmit streams beyond N_r. When $N_t > N_r$ a more sensible approach would be to encode N_r transmit streams across the N_t transmit antennas using a space-time code. Additionally, the assumption that $N_r \geq N_t$ avoids the complexity challenge inherent in underdetermined systems [25]. We assume the receiver knows the channel perfectly. Finally, we assume the transmitter has no knowledge of channel state information (CSI). We justify this by noting that providing accurate CSI to the transmitter requires overcoming mismatched CSI between the transmitter and receiver, caused by the analogue front-end and background noise. Moreover, transmitter CSI is only possible when the channel changes slowly with time; a situation typically not found in high frequency communications. While we have elected to not consider channel estimation and transmitter CSI, we note that these are important topics that prevent performance degradation and improve reliability/performance, respectively.

We now incorporate our MIMO channel model into a communication system for use throughout this dissertation. Our simple transmitter model is shown in Fig. 2.2-a [47]. The input is a vector $\mathbf{u}$ of i.i.d. uniform information bits that is encoded and interleaved, perhaps using a turbo or low-density parity-check (LDPC) code. We then partition the coded bit stream into blocks $\mathbf{c}$ of ωN_t bits and map each block onto a vector $\mathbf{a}$ whose N_t component symbols are taken from the complex alphabet $\mathcal{A}$ of size $q = |\mathcal{A}| = 2^\omega$ and energy $\mathbb{E}\left(|a_i|^2\right) = E/N_t$, where ω is the number of bits per symbol. The vector of transmitted symbols $\mathbf{a}$ is sent through the $N_r \times N_t$ MIMO channel model of 2.1 to produce the vector of received samples $\mathbf{r}$ at the receiver.

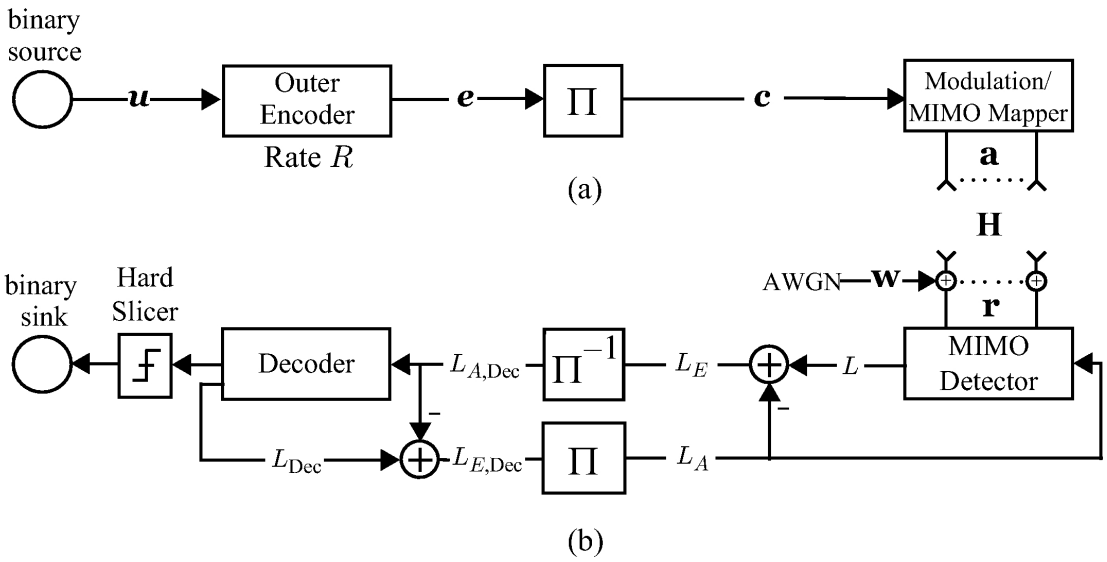

Figure 2.2: Coded system model depicting (a) MIMO transmitter and (b) MIMO receiver.

In Fig. 2.2-b we show a MIMO receiver consisting of a MIMO detector (this dissertation's focus), a deinterleaver, and an error-control decoder. The detector's job is to produce hard or soft estimates L for the bit stream $\mathbf{c}$. After the detector the receiver subtracts off any *a priori* information from the decoder, i.e. L_A, and then deinterleaves this signal producing $L_{A,\text{Dec}}$, the *a priori* information for the decoder. The decoder improves the estimate for $\mathbf{u}$ by exploiting the presence of the error correcting code. An iterative detection-decoding system based on the turbo principle [44] can improve performance.

In this dissertation we will not concern ourselves with a receiver employing iterative detection-decoding for two reasons. First, by focusing on a non-iterative receiver we are able to capture the essence of the soft-output MIMO detection problem with distracting ourselves with the added distraction of a detection-decoder symbiotic relationship based on the turbo-principle. Second, when the computational complexity resources available to the receiver are low, performing multiple executions of the detector and decoder is computationally burdensome and goes against our dissertation objective of low computational complexity. For these two reasons we limit our discussion to a non-iterative system where there is no feedback from the decoder to the detector.

2.3 Problem Statement

The aim of a soft-output detector is to calculate or approximate the *a posteriori* probability (APP) for each of the coded bits c_j in a given signaling interval, where $j \in \{1, \ldots \omega N_t\}$ is the bit index. This probability is conveniently represented by the so-called a posteriori log-likelihood ratio (LLR):

$$L(c_j|\mathbf{r}) := \ln \frac{\Pr[c_j = +1|\mathbf{r}]}{\Pr[c_j = -1|\mathbf{r}]}. \tag{2.2}$$

The sign of $L(c_j|\mathbf{r})$ is the maximum a posteriori (MAP) estimate for c_j, and the magnitude represents the reliability of the estimate. Larger magnitudes correspond to higher reliability, and smaller magnitudes indicate low reliability. In particular, the extreme case of $L = 0$ indicates that c_j is equally likely to be $+1$ and -1.

Applying Bayes' rule to (2.2) yields:

$$\begin{aligned} L(c_j|\mathbf{r}) &= \ln \frac{f(\mathbf{r}|c_j = +1)\left(\Pr[c_j = +1]/f(\mathbf{r})\right)}{f(\mathbf{r}|c_j = -1)\left(\Pr[c_j = -1]/f(\mathbf{r})\right)} \\ &= \ln \frac{\Pr[c_j = +1]}{\Pr[c_j = -1]} + \ln \frac{f(\mathbf{r}|c_j = +1)}{f(\mathbf{r}|c_j = -1)} \\ &= L_A(c_j) + L_E(c_j|\mathbf{r}), \end{aligned} \tag{2.3}$$

where $\Pr[c_j = +1]$ and $\Pr[c_j = -1]$ are the *a priori* probabilities that bit c_j is 1 or -1, respectively, and where

$$L_A(c_j) = \ln \frac{\Pr[c_j = +1]}{\Pr[c_j = -1]} \tag{2.4}$$

is the *a priori* LLR for the j-th bit. The second term L_E in (2.3) represents the *extrinsic* contribution to the a posteriori LLR [47]. Using the law of total probability, it can be written as:

$$L_E(c_j|\mathbf{r}) = \ln \frac{f(\mathbf{r}|c_j = +1)}{f(\mathbf{r}|c_j = -1)} = \ln \frac{\sum_{\hat{\mathbf{c}} \in \mathcal{X}_j^{+1}} f(\mathbf{r}|\hat{\mathbf{c}})}{\sum_{\hat{\mathbf{c}} \in \mathcal{X}_j^{-1}} f(\mathbf{r}|\hat{\mathbf{c}})}, \tag{2.5}$$

where we rely on a partitioning of the vector alphabet into two, depending on whether the bit of interest is 1 or -1. Specifically, the set of possible $\mathbf{c}$ vectors $\mathcal{X} = \{\pm 1\}^{\omega N_t}$ is partitioned into $\mathcal{X}_j^{+1}$ and $\mathcal{X}_j^{-1}$, where $\mathcal{X}_j^{+1}$ denotes the set of $2^{\omega N_t - 1}$ vectors $\mathbf{c} \in \mathcal{X}$ for which $c_j = +1$, and

$\mathcal{X}_j^{+1}$ denotes the set of $2^{\omega N_t - 1}$ vectors $\mathbf{c} \in \mathcal{X}$ for which $c_j = -1$. For use later, let us similarly define $\mathcal{Z} = \mathcal{A}^{N_t}$ as the set of all possible symbol vectors $\mathbf{a}$, one for each binary vector $\mathbf{c} \in \mathcal{X}$, as determined by the mapping from coded bits to transmitted symbols. Similar to $\mathcal{X}_j^{\pm 1}$, let $\mathcal{Z}_j^{\pm 1}$ denote the partitioning of $\mathcal{Z}$ depending on whether the j^{th} bit label is 1 or -1, namely:

$$\begin{aligned} \mathcal{Z}_j^{+1} &= \left\{ \mathbf{a}(\mathbf{c}) : c_j = +1 \right\}, \\ \mathcal{Z}_j^{-1} &= \left\{ \mathbf{a}(\mathbf{c}) : c_j = -1 \right\}. \end{aligned} \tag{2.6}$$

Fig. 2.3 shows an example of partitioning for a vector alphabet after linear transformation by $\mathbf{H}$. The vector alphabet size is 32, which might arise from a binary scalar alphabet and $N_t = 5$. The partitions in (a) and (b) correspond to bits one and two from the binary transmission vector.

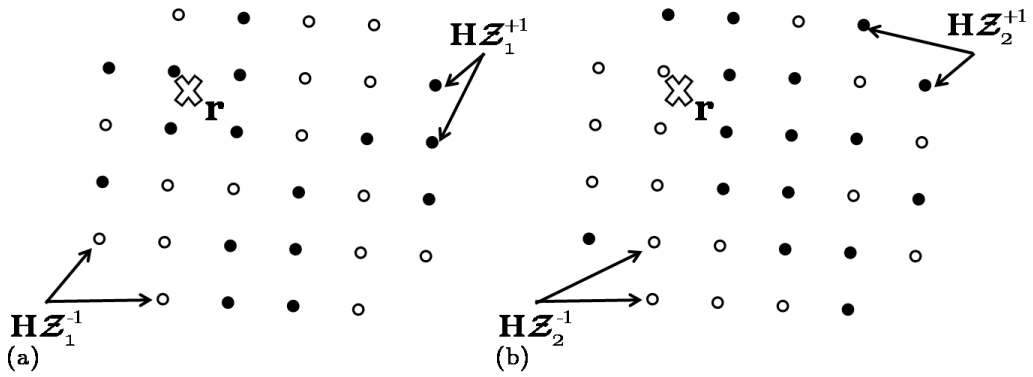

Figure 2.3: Partitioning of the set of transmission vectors $\mathcal{Z}$ after linear transformation by a channel $\mathbf{H}$ for the bit mapping corresponding to the (a) first bit transmitted from the first antenna c_1 and the (b) first bit transmitted from the second antenna c_2. Open and closed circles correspond to valid transmission vectors where the bit of interest is -1 and $+1$, respectively.

Since the noise is AWGN, the conditional probability density function $f(\mathbf{r}|\hat{\mathbf{c}})$ reduces to [12]:

$$f(\mathbf{r}|\hat{\mathbf{c}}) = \frac{1}{(\pi N_0)^{N_r}} \exp\left(\frac{-\|\mathbf{r} - \mathbf{H}\mathbf{a}(\hat{\mathbf{c}})\|^2}{N_0} \right), \tag{2.7}$$

where $\mathbf{a}(\hat{\mathbf{c}}) \in \mathcal{Z}$ is the unique vector of transmitted symbols associated with the bit vector $\hat{\mathbf{c}}$.

Substituting (2.7) and (2.5) into (2.3) yields:

$$L(c_j|\mathbf{r}) = \underbrace{\ln \frac{\Pr[c_j = +1]}{\Pr[c_j = -1]}}_{L_A(c_j)} + \underbrace{\ln \frac{\sum_{\hat{\mathbf{c}} \in \mathcal{X}_j^{+1}} \exp\left\{-\|\mathbf{r} - \mathbf{H}\mathbf{a}(\hat{\mathbf{c}})\|^2 / N_0\right\}}{\sum_{\hat{\mathbf{c}} \in \mathcal{X}_j^{-1}} \exp\left\{-\|\mathbf{r} - \mathbf{H}\mathbf{a}(\hat{\mathbf{c}})\|^2 / N_0\right\}}}_{L_E(c_j|\mathbf{r})}, \tag{2.8}$$

where L is broken into an *a priori* component L_A and an extrinsic component L_E.

In the context of the iterative receiver shown in Fig. 2.2, the *a priori* information L_A is provided to the detector via feedback from the error-control decoder, and the extrinsic information L_E is the extra contribution to the a posteriori LLR that was gleaned from the detector, above and beyond what was provided by the decoder. Computing (2.8) for a given bit c_j requires knowledge of the received vector $\mathbf{r}$, the channel $\mathbf{H}$, the mapping $\mathbf{a}(\cdot)$ from bits to symbols, and any *a priori* information, if available.

Exact evaluation of (2.8) requires that a computation of the form $\|\mathbf{r} - \mathbf{H}\mathbf{a}\|^2$ be computed q^{N_t} times. As an example, for a 4-input MIMO system with each input coming from a 64-QAM alphabet, this amounts to over 16 million times! Clearly a lower complexity solution is needed. Additionally, the exponential operation must be applied to each of these 16 million squared norms, resulting in an extremely high computational complexity.

The max-log approximation

$$\begin{aligned} \ln(e^a + e^b) &= \max\{a, b\} + \ln(1 + e^{-|a-b|}) \\ &\approx \{a, b\} \qquad |a - b| >> 1 \end{aligned} \tag{2.9}$$

significantly reduce the complexity of computing (2.8) with only a slight performance degradation. A common approximation for (2.8) is to use the *max-log* approximation:

$$L(c_j|\mathbf{r}) \approx L_A(c_j) + \max_{\hat{\mathbf{c}} \in \mathcal{X}_j^{+1}} \left\{ \frac{-\|\mathbf{r} - \mathbf{H}\mathbf{a}(\hat{\mathbf{c}})\|^2}{N_0} \right\} - \max_{\hat{\mathbf{c}} \in \mathcal{X}_j^{-1}} \left\{ \frac{-\|\mathbf{r} - \mathbf{H}\mathbf{a}(\hat{\mathbf{c}})\|^2}{N_0} \right\}. \tag{2.10}$$

The max-log approximation is based on the assumption that the exponential term with maximum argument in the sum of exponentials will dominate the summation. By avoiding the sum of exponentials only one exponential term remains in the numerator and one in the denominator of (2.8). After some simplifications, the result is (2.10). This approximation

is widely accepted because of its relatively small performance loss [43, 74]. The max-log approximation does not, however, reduce the problem size of soft-output MIMO detection. Specifically, a brute-force search for the $\mathbf{a}(\hat{\mathbf{c}}) \in \mathcal{Z}$ minimizing $\|\mathbf{r} - \mathbf{H}\mathbf{a}(\hat{\mathbf{c}})\|^2$ in (2.10) would still need to consider q^{N_t} possibilities.

Although at a glance it might appear from (2.10) that the receiver will need to perform a pair of optimizations for each of the ωN_t bits of interest, for a total of $2\omega N_t$ optimizations per signaling interval, in fact the MAP solution will always be one of each pair. So to compute (2.10) for each of the ωN_t bits, it would be sufficient to find the MAP solution once, and then, for each of the ωN_t bits, to find a candidate for which the bit of interest is negated. This candidate, for which the bit of interest is the negation of the MAP (or approximate MAP) solution is known as a *counterhypothesis*.

A detector that solves (2.10) exactly is *max-log optimal*. Specifically, a max-log optimal detector finds the MAP solution and the minimum cost counterhypothesis transmission vector for each transmitted bit. A max-log optimal detector that searches over the entire set of possible transmission vectors $\mathbf{a} \in \mathcal{Z}$ to solve (2.10) exactly is intractable for even a small number of transmit antennas and moderate alphabet sizes.

Near max-log optimal performance can be obtained by constructing a *list* of transmission vectors $\mathcal{L} \subseteq \mathcal{Z}$. *List detection* is the process of finding this list of candidates. The list detection version of (2.10) is given by:

$$L(c_j|\mathbf{r}) \approx L_A(c_j) + \max_{\hat{\mathbf{a}} \in \mathcal{L} \cap \mathcal{Z}_j^{+1}} \left\{ \frac{-\|\mathbf{r} - \mathbf{H}\hat{\mathbf{a}}\|^2}{N_0} \right\} - \max_{\hat{\mathbf{a}} \in \mathcal{L} \cap \mathcal{Z}_j^{-1}} \left\{ \frac{-\|\mathbf{r} - \mathbf{H}\hat{\mathbf{a}}\|^2}{N_0} \right\}. \tag{2.11}$$

Despite the notational changes to operate on the vector alphabet $\mathcal{Z}$ instead of the binary vector $\mathcal{X}$, the key difference between (2.10) and (2.11) is the insertion of the list $\mathcal{L}$, where the list length $\ell = |\mathcal{L}|$ plays a critical roll in the overall complexity and performance.

Two example lists, for the received vector $\mathbf{r}$ from Fig. 2.3, are depicted in Fig. 2.4. The elements within the boundaries of the circular region depicted in Fig. 2.4-a comprise the minimum-distance list. The elements within the boundaries of the region in Fig. 2.4-b comprise a suboptimal list. The performance of a system using the list in Fig. 2.4-b would

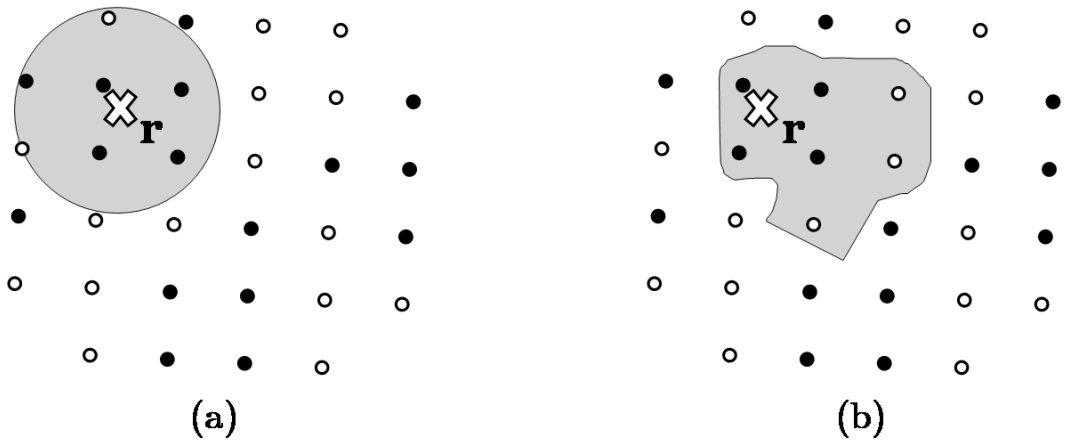

Figure 2.4: A minimum distance list (a) contains the ℓ candidates closest to $\mathbf{r}$, while a suboptimal list (b) might contain different candidates.

suffer relative to the one using the list in Fig. 2.4-a because the list in Fig. 2.4-b does not include the elements to optimize (2.10) for each of the 5 transmission bits.

In the remainder of this dissertation we consider the scenario where there is no *a priori* information, i.e. no feedback from the decoder to the detector, as justified in section **??**. In this setting (2.11) reduces to:

$$L(c_j|\mathbf{r}) \approx \frac{1}{N_0}\left(\min_{\hat{\mathbf{a}}\in\mathcal{L}\cap\mathcal{Z}_j^{+1}}\|\mathbf{r}-\mathbf{H}\hat{\mathbf{a}}\|^2 - \min_{\hat{\mathbf{a}}\in\mathcal{L}\cap\mathcal{Z}_j^{-1}}\|\mathbf{r}-\mathbf{H}\hat{\mathbf{a}}\|^2\right). \qquad (2.12)$$

In terms of the partition into white and black points, as shown in Fig. 2.4, one of the minimizations of (2.12) will produce the squared distance to the nearest black point, while the other minimization will produce the squared distance to the nearest white point. Specifically we see that, because the suboptimal list of Fig. 2.4-b excludes the closest white point, it will overestimate the reliability of the bit in question.

2.3.1 List Tree Search

A useful construct for efficiently finding a list of candidates is that of a tree [3, 55, 108]. The detection tree can be derived and interpreted in two ways: either geometrically or algebraically.

The geometric view is based on the fact that the candidate vectors after the channel fall on a lattice whenever the alphabets are QAM. And any lattice can be decomposed into the union of multiple sublattices that are translated relative to each other. Therefore, the squared distance from a received vector to any point on the lattice is easily expressed in terms of the projection of the received vector onto the hyperplane spanned by the lattice point's sublattice; namely, the squared distance decomposes into the sum of the squared distance from the received vector to the projection vector, plus the squared distance from the projection vector to the sublattice point. And this latter term can be computed recursively based on the same principle.

As an example of this geometric construction [88], consider the example shown in Fig. 2.5 for a 2-transmitter system with an antipodal alphabet $\mathcal{A} = \{+1, -1\}$. Fig. 2.5-a depicts a received vector $\mathbf{r}$ and the valid transmission vectors after being transformed by the channel. These detection vectors are denoted with a gray circle, a black circle, a gray-black and a black-gray circle corresponding to the valid vectors $\{-1, -1\}$, $\{+1, +1\}$, $\{-1, +1\}$ and $\{+1, -1\}$, respectively.

Ultimately we are interested in computing the squared Euclidean distance from the received vector to a valid transmission vector in the detection space, i.e. $\|\mathbf{r} - \mathbf{H}\hat{\mathbf{a}}\|^2$. For the purposes of the example in Fig. 2.5-a we can employ the Pythagorean theorem to compute these squared Euclidean norms. Specifically, if we are interested in computing the squared norm from $\mathbf{r}$ to the detection vector $\mathbf{H}\hat{\mathbf{a}}$, where $\hat{\mathbf{a}} = \{-1, -1\}$, then one leg of a triangle corresponds to the squared distance from $\mathbf{r}$ to Φ and the other leg corresponds to the squared distance from Φ to $\mathbf{H}\hat{\mathbf{a}}$. Summing these squared distances via Pythagoras, i.e. $c^2 = a^2 + b^2$, yields the squared Euclidean distance $\|\mathbf{r} - \mathbf{H}\hat{\mathbf{a}}\|^2$ when $\hat{\mathbf{a}} = \{-1, -1\}$. Similarly the squared

Euclidean distances to all other elements of $\mathbf{H}\mathcal{Z}$ can be computed.

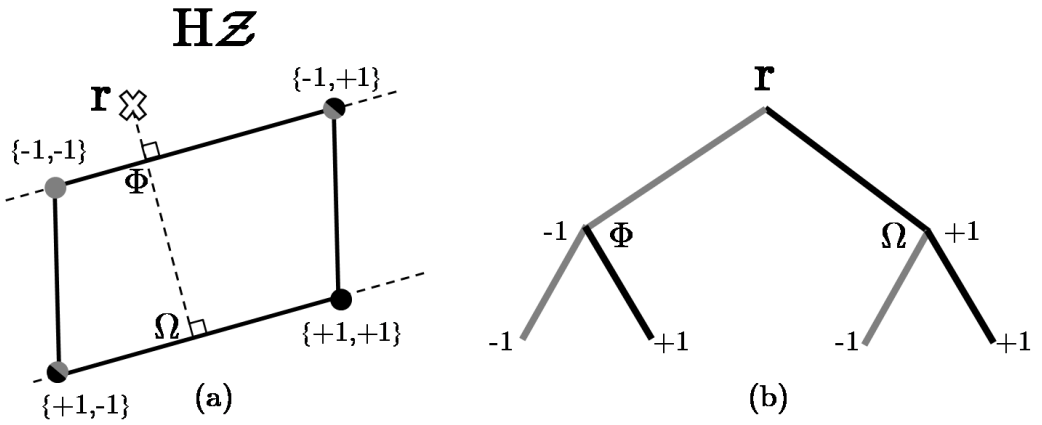

Figure 2.5: A q-ary tree for $N_t = 2$ and $\mathcal{Z} = \{-1,-1\},\{-1,+1\},\{+1,+1\}$ and $\{+1,-1\}$ ($q = 2$).

Fig. 2.5-b depicts these same squared Euclidean distance calculations using a tree. The distance from $\mathbf{r}$ to Φ is the cost when the first detected symbol is -1 and is represented by the gray branch emanating from the root node of the detection tree. Similarly, the distance from $\mathbf{r}$ to Ω corresponds to the cost when the first detected symbol is $+1$ and is represented by the black branch emanating from the root node. Computing the squared Euclidean distances from Φ to $\mathbf{H}\hat{\mathbf{a}}$, where $\hat{\mathbf{a}} = \{-1,-1\}$ and $\hat{\mathbf{a}} = \{-1,+1\}$ completes the left half of the tree and computing the squared Euclidean distances from Ω to $\mathbf{H}\hat{\mathbf{a}}$, where $\hat{\mathbf{a}} = \{+1,-1\}$ and $\hat{\mathbf{a}} = \{+1,+1\}$ completes the right half of the tree. Consequently, each of the nodes at the bottom of the detection tree, referred to as *leaf nodes*, corresponds to a unique decision from the set of all possible transmission vectors $\mathcal{Z}$.

In contrast to the geometric view, the algebraic view is based on a QR decomposition. Specifically, the squared distance for a candidate $\hat{\mathbf{a}}$ is:

$$J(\hat{\mathbf{a}}) = \|\mathbf{r} - \mathbf{H}\hat{\mathbf{a}}\|^2 \tag{2.13}$$

$$= \|\mathbf{y} - \mathbf{R}\hat{\mathbf{a}}\|^2 \tag{2.14}$$

$$= \sum_{1 \le i \le N_t} |y_i - \sum_{1 \le v \le i} R_{iv}\hat{a}_v|^2, \tag{2.15}$$

where $\mathbf{H} = \mathbf{QR}$ is the QR decomposition of the channel matrix $\mathbf{H}$, where $\mathbf{R}$ is an $N_t \times N_t$ lower triangular matrix, where $\mathbf{Q}$ is an orthogonal matrix and where $\mathbf{y} = \mathbf{Q}^*\mathbf{r}$. The QR decomposition is an orthogonal and triangular decomposition of the channel matrix $\mathbf{H}$ allowing for the computation of (2.15).

The cost function (2.15) can be interpreted as the sum of N_t *branch metrics*, one for each branch in a path from the root of the detection tree to a leaf node, where the metric for a branch in the i-th stage of the detection tree with path history $\{a_1, a_2, \ldots a_i\}$ is [12]:

$$\left| y_i - \sum_{1 \le v \le i} R_{iv} a_v \right|^2 . \tag{2.16}$$

We refer to the sum of the i branch metrics in the path from the root node to the node of interest in the detection tree as a *path metric*.

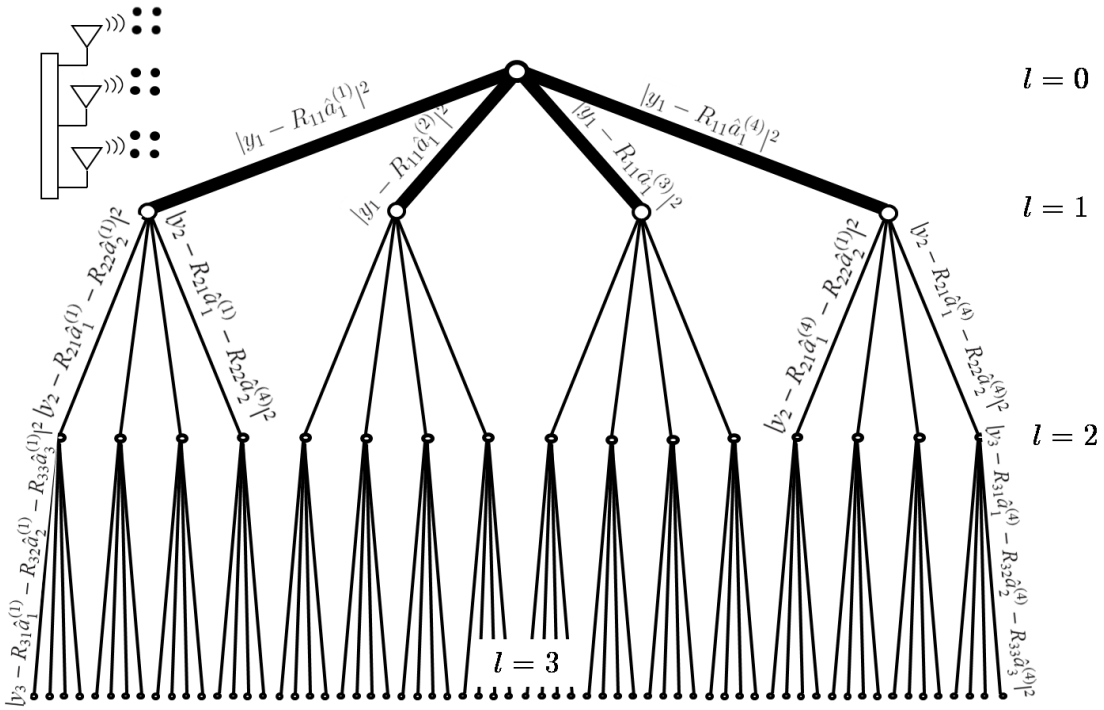

Figure 2.6: A 4-ary tree for $N_t = 3$.

Fig. 2.6 depicts a detection tree for a 3-input MIMO system employing a 4-ary alphabet. There are three *layers* of the tree, one for each input, and there are $q^{N_t} = 4^3 = 64$ leaf nodes. There are $4 + 16 + 64 = 84$ total branches. Some of the branch metrics are shown, where

the superscript for a decision $\hat{\mathbf{a}}_i^{(\cdot)}$ denotes the index from the q-ary alphabet.

Keeping in mind the problem of list detection, the objective of tree-based detection is to find the $\ell = |\mathcal{L}|$ leaf nodes in the tree, corresponding to valid elements from the set $\mathcal{Z}$, that yield an accurate solution to (2.12). Using the minimum distance list as our guide, we seek to find the ℓ leaf nodes in the tree with minimum cost. Specifically, the soft-output MIMO detection problem boils down to the following:

Goal: Find the ℓ leaf nodes in the tree with minimum cost.

2.4 Search Algorithms

There are many ways to search the detection tree in order to find a set of leaf nodes with low costs. We define an exact tree-based list search algorithm as one that finds the ℓ minimum cost leaf nodes in the tree. An approximate tree-based list search algorithm finds ℓ leaf nodes with *low* (but not necessarily lowest) cost. In this section we consider classical and state-of-the art tree search algorithms used to solve the soft-output list MIMO detection problem either exactly or approximately.

2.4.1 Exact Minimum Cost Tree Search

We now present two classical algorithms for exact tree-based list detection. The first is perhaps the most famous soft-output detection algorithm, known as the list sphere detector (LSD) [47]. The second is the list sequential (LISS) detector [11, 42], which in [111] was shown to be the tree search scheme with lowest complexity, under certain assumptions, for solving the soft output detection task *optimally* for a given list size, where optimality is defined as maximizing the a posteriori probability.

2.4.1.1 The List Sphere Detector

The LSD [47] is a famous variable complexity algorithm for finding the ℓ minimum cost leaf nodes in the tree exactly. In this section we begin with an intuitive description of the LSD. Following this, we provide a flowchart for implementing the LSD.

The LSD begins at the root node and advances through the tree in a greedy fashion. At each node the LSD selects the child node with minimum weight. This process continues until we reach a leaf node or the cost of the node we are visiting exceeds a threshold. For the purposes of this discussion we initialize the threshold to ∞. Consequently, a leaf node will always be found at the start of our search. This leaf node becomes the first element of a list, although it might later be replaced.

The LSD then backtracks one layer and considers the inclusion of "sibling" nodes of the leaf node it has just found. A Fincke-Pohst [32] enumeration would explore these siblings in a natural order, say from left to right, with no regard to their weights. Fewer nodes will be visited if a Schnorr-Euchner [76] enumeration is adopted, which explores the siblings in an order determined by their weights, with the best first. We assume Schnorr-Euchner enumeration in our discussion.

The process of adding sibling nodes to the first leaf node found continues until either there are no more sibling nodes to enumerate, or until the list consists of ℓ leaf nodes. In either case the algorithm backtracks an additional layer in the tree so that it is two layers removed from the leaf nodes. If the algorithm backtracks because the list is comprised of ℓ leaf nodes, then the threshold value must be updated to the weight of the highest cost leaf node in the list.

Now two layers removed from the leaf nodes in the tree, the LSD enumerates the lowest weight child node it has not yet explored and continues in a greedy fashion to either add leaf nodes to the list (if its cost is less than the threshold) or backtrack (if the cost exceeds the threshold). Anytime a leaf node is found with weight less than the threshold and the list size is already ℓ, the new leaf node replaces the leaf node in the list with highest cost.

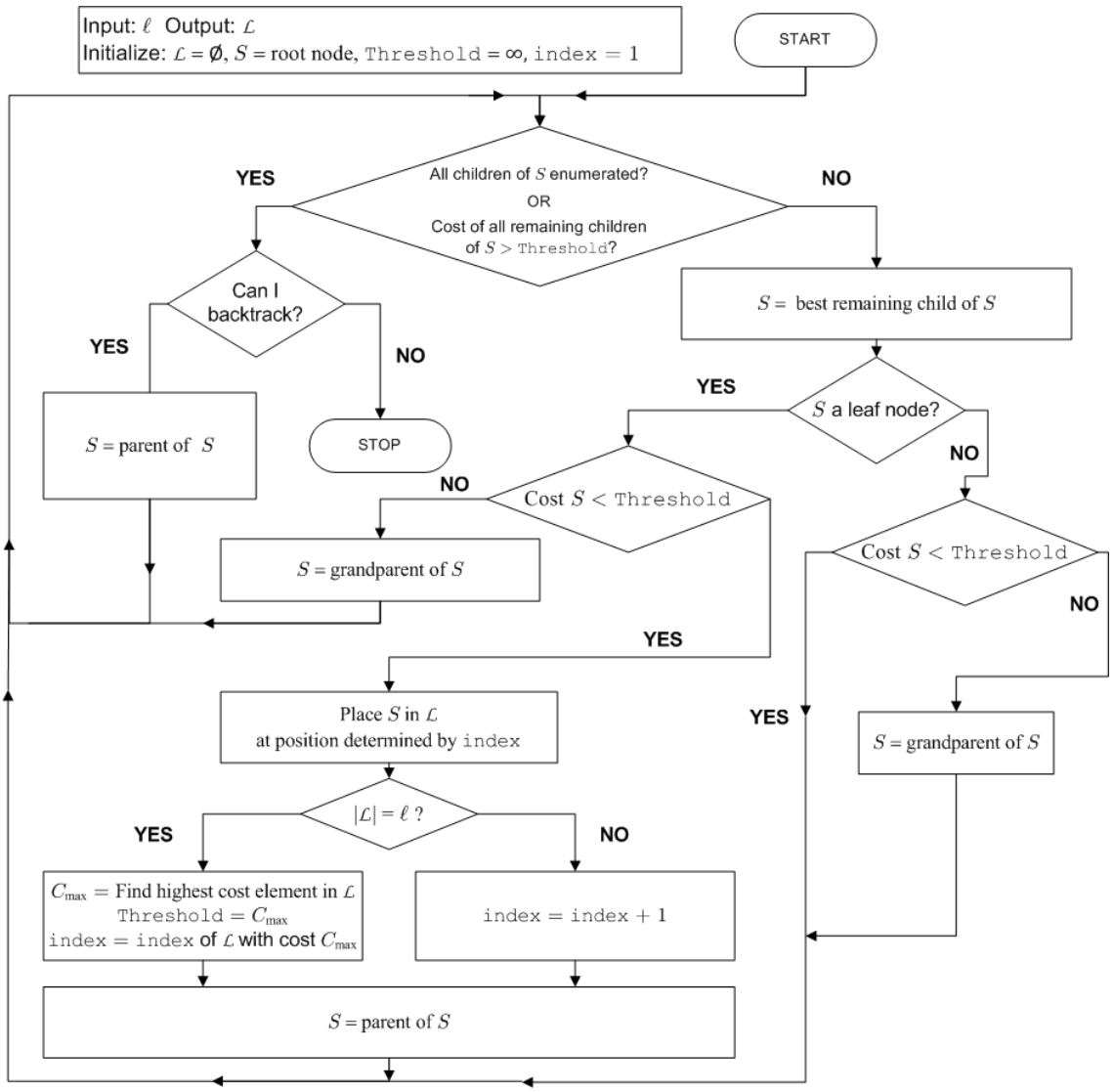

Figure 2.7: Flow chart for Schnorr-Euchner realization of list sphere detector [47].

Upon a replacement event, the LSD updates the threshold to the cost of the largest weight leaf node remaining in the list.

When there are no more nodes left to be explored at a given layer in the tree either because of exhaustive search or all remaining nodes exceed the threshold, the algorithm backtracks up the tree to determine if there are any nodes left to be explored at higher layers in the tree. The LSD terminates when it can no longer backtrack.

The process just described for finding leaf nodes in the detection tree, where the search begins at the root node and proceeded as far as possible in the tree before backtracking,

is known as *depth-first* search. Fig. 2.7 summarizes the depth-first LSD using a flowchart. The only listed input to the algorithm is the list length ℓ. The output of the LSD is the list $\mathcal{L}$. In order to compute the branch metrics, as described in (2.15), we require the equalized received signal $\mathbf{y}$ and the triangular matrix $\mathbf{R}$, but we omit these as inputs for simplicity of notation.

2.4.1.2 The List Sequential Detector

An alternative to the LSD for finding the ℓ minimum cost leaf nodes in the tree is the list sequential (LISS) detector [11, 42]. In contrast to the LSD, which maintains only one node at a time in the detection tree, the LISS maintains multiple nodes in the tree simultaneously. During the search of the detection tree the node(s) that currently has the lowest cost(s) is/are extended. We call this type of search *metric-first* search [71]. We note that the nodes maintained need not be at the same layer in the tree.

The LISS algorithm is implemented using a stack to maintain the nodes currently under consideration. In this section we begin with an intuitive description of the LISS. We follow up with a flowchart description.

With the stated objective of finding the ℓ leaf nodes with minimum weights in the tree, the LISS algorithm begins by initializing the stack to be the root node and its associated cost to be 0. After this initialization, we remove this minimum cost node in the stack (at this point it is the only node in the stack) and replace it in the stack with all $q = |\mathcal{A}|$ of its child nodes. We then order the stack in terms of costs, placing the minimum cost node at the top of the stack.

Here the process begins to repeat itself. As before we remove the minimum cost node from the stack and replace it with its q children. We reorder the stack once more and replace the minimum cost node by its q children. Each time leaf nodes are reached we remove them from the stack and place them in a list.

The algorithm terminates when the node on top of the ordered stack (i.e. the one with

minimum cost) has weight greater than or equal to the cost of the ℓ^{th} minimum weight leaf node in the list[1]. Upon termination we can truncate the list to the ℓ minimum weight leaf nodes.

Fig. 2.8 summarizes the LISS algorithm using a flow chart. As before, we assume that all branch weights are known so that the only input required by the algorithm is the list length ℓ. The output of the algorithm is the list $\mathcal{L}$. Unlimited memory is assumed to avoid a discussion about truncation of the stack.

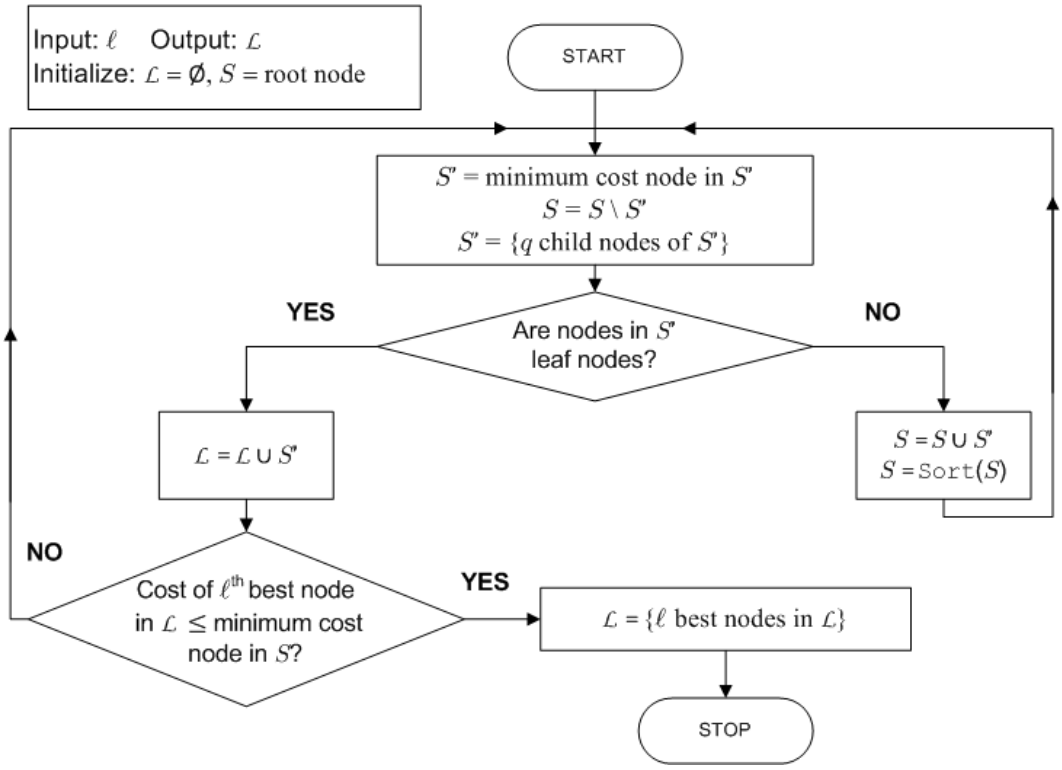

Figure 2.8: Flow chart for list sequential detector [11].

2.4.2 Approximate Minimum Cost Tree Search

The LSD and LISS algorithms just described are efficient ways to achieve the goal of finding the ℓ leaf nodes in a tree with minimum weights. What happens if we modify our goal by relaxing the constraint that we must find the ℓ leaf nodes with *minimum* weights

[1]This termination condition is slightly different than the one in [11], but is needed to ensure that we find the ℓ minimum weight leaf nodes.

and instead search for ℓ leaf nodes with *small* weights? The advantage of such an approach would be that our search could visit fewer nodes in the tree, thereby reducing complexity. The obvious disadvantage would be a suboptimal solution to our problem.

For the LSD and LISS algorithms described in the previous section we can solve the relaxed constraint problem through early termination. For the LSD this can be achieved by stopping once a certain number of nodes have been visited or by aggressively reducing the threshold value to search fewer nodes in the tree. For the LISS algorithm, we can terminate early once a certain number of leaf nodes are in the list or we could reduce the size of the stack to a predetermined fixed value such that it retains fewer nodes. A third option is to bias paths based on their layer in the tree to avoid extending nodes which are unlikely to produce a leaf node with small weight [41]. Additionally, to achieve our new goal of finding the ℓ leaf nodes with *small* weight, we can not only modify the LSD or LISS algorithms, but we can also consider other ways to search the tree.

An efficient way to search a tree, when a suboptimal list is allowed, is to search the tree layer-by-layer and at each layer remove nodes which are unlikely to produce a leaf node with small weight. Algorithms that search the tree using a layer-by-layer approach are *breadth-first*. Breadth-first search algorithms have higher complexity than depth-first or metric first approaches when the goal is to find the minimum cost(s) leaf node(s) in the tree. They are, however, a viable alternative when suboptimal search is allowed. This is because, in contrast to depth-first or metric first approaches, breadth-first algorithms have fixed complexity, meaning that they visit the same number of nodes in the tree independent of the branch weights on the tree. This is significant because it means that the algorithm has a regular structure which lends itself to practical implementation.

We now describe a breadth-first algorithm, which we call the soft M algorithm because of its close relation to the classical M algorithm [3]. The soft M algorithm begins by extending the $b \leq q$ minimum weight nodes from the root of the tree using Schnorr-Euchner enumeration. Assuming the algorithm would like to keep all of these nodes, it can then

extend the b best child nodes from each of the b parent nodes yielding b^2 nodes at the second layer in the tree. If b^2 is greater than some value, call it m, then the algorithm sorts the b^2 nodes and retains only the m best. This process of extending b nodes from each retained parent node and retaining only the m minimum weight nodes (assuming m is less than the number of retained nodes) continues until we reach the final layer in the tree, where ℓ leaf nodes are found for inclusion in the list (assuming $\ell \leq m$).

Fig. 2.9 summarizes the soft M algorithm using a flow chart. The inputs to the algorithm are the list length ℓ, the b parameter, which determines the number of nodes to extend from each retained parent node, and the m parameter, which is used to prune the tree when the number of nodes in the tree exceeds m. The output of the algorithm is the list $\mathcal{L}$.

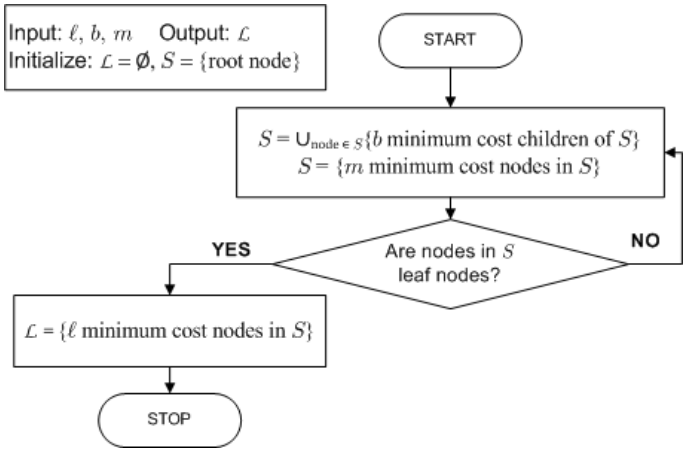

Figure 2.9: Flow chart for soft M algorithm.

2.4.3 Max-Log Optimal Tree Search

We now return to the problem statement at the end of section 2.3, namely (2.12). Ignoring for the moment the use of a list, one way to solve this expression exactly (i.e. *max-log optimally*) is to first find the $\hat{\mathbf{a}}^{\text{MAP}} \in \mathcal{Z}$ minimizing $\|\mathbf{r}-\mathbf{H}\hat{\mathbf{a}}\|^2$, where $\hat{\mathbf{a}}^{\text{MAP}}$ denotes the maximum a posteriori (MAP) solution. Then, perform a constrained search for all $j = \{1, \ldots \omega N_t\}$ for $\hat{\mathbf{a}}_j^{-\text{MAP}} \neq \hat{\mathbf{a}}^{\text{MAP}}$, where $\hat{\mathbf{a}}_j^{-\text{MAP}}$ denotes the constrained MAP solution subject to the constraint

that its j^{th} bit is the negation of the j^{th} bit for $\hat{\mathbf{a}}^{\text{MAP}}$. This negation is a boolean logic negation if we are talking about binary bits 1 and 0 and straightforward in the case of 1 and -1. The vectors $\hat{\mathbf{a}}^{\text{MAP}}$ and $\hat{\mathbf{a}}_j^{-\text{MAP}}$ are counterhypotheses of one another because the bit of interest is a different hypothesis (e.g. either 1 or -1) for each vector.

2.4.3.1 Smart Candidate Adding

Smart candidate adding (SCA) is the name given to approaches that search for candidate lists that either exactly or approximately include (1) $\hat{\mathbf{a}}^{\text{MAP}}$ and (2) $\hat{\mathbf{a}}_j^{-\text{MAP}}$ for each of the $j \in \{1, \dots \omega N_t\}$ transmitted bits [62]. Because there are ωN_t bits for each $\mathbf{a}$, the size of the list $\mathcal{L}$ optimally solving (2.12), for a given $\mathbf{r}$ and $\mathbf{H}$, is at most $\ell = 1 + \omega N_t$, where the 1 term corresponds to $\hat{\mathbf{a}}^{\text{MAP}}$ and the ωN_t terms stem from the fact that the constrained solutions for each of the ωN_t bits might be a unique element from $\mathcal{Z}$.

Smart candidate adding is effective for improving the error rate performance of soft-output MIMO detection algorithms. Early SCA approaches [62, 105, 94] focused on finding the maximum a posteriori (MAP) estimate (or an approximation thereof) and supplementing this estimate with directed searches for counterhypotheses.

In [50] an improvement over the SCA approaches of [62, 105, 94] was proposed that finds its list using a single pass through the detection tree, rather than using multiple searches. A state-of-the-art max-log optimal detection algorithm, motivated by the algorithm in [50] is the single-tree-search list sphere detector (STS-LSD) [87]. Specifically, the STS-LSD is an efficient depth-first algorithm that searches for the MAP estimate and each of the ωN_t optimal counterhypotheses without visiting nodes in the detection tree more than once. In contrast to the classical LSD, the STS-LSD prunes the detection tree only when a given node is unable to produce an improved MAP estimate or counterhypothesis to the current MAP estimate. This targeted search, focused on counterhypotheses rather than the ℓ minimum distance candidates, is efficient for finding the max-log optimal list of candidates.

2.5 Computational Complexity

Thus far in this chapter our list MIMO detection goal is the construction of low computational complexity tree-based soft-output detectors with near max-log optimal error rate performance. Assessing whether this goal is achieved requires knowledge of the system error rate performance *as well as* the computational complexity. A trade-off exists between these two salient properties [95].

Evaluating the error rate performance of a soft-output detector is straightforward. All that is required is to measure the average number of bit, symbol, or frame errors. Evaluating a detector's computational complexity is typically far more exacting because there are many ways by which complexity can be measured.

The receiver's computational complexity is critically important because it affects the chip size, execution time and power requirements for practical systems. Analyzing computational complexity is often a difficult undertaking because there are many ways by which we measure complexity. Despite these obstacles there are a number of effective techniques for roughly analyzing computational complexity. In this section we provide an overview of methods for measuring computational complexity, followed by a discussion of the computational complexity metrics used throughout this dissertation. We then establish computational complexity bounds and reference complexities for tree-based detection. Finally, a collection of results demonstrate the utility of the aforementioned reference complexities.

2.5.1 Measuring Computational Complexity

Measuring computational complexity for the purpose of constructing a fair comparison amongst different solutions to the tree-based soft-output MIMO detection problem is a challenge. Nuanced trade-offs between latency, power consumption and silicon area all play important roles in the resultant quality of a detector's architectural implementation. These trade-offs are often jumbled by design constraints that force system engineers to

tailor implementations for target architectural platforms. For example, an optimized architectural implementation used for a fixed-point ASIC would differ significantly from one tailored for a floating point DSP.

The number of design considerations necessary for an initial architectural implementation, and the more time-consuming task of design optimization, render a direct and fair comparison between competing algorithms a formidable task, although some system attributes are quantifiable without a complete architectural implementation. An example of such an architecture agnostic attribute is the order of an algorithm's execution time. However, in most cases, an architectural implementation is required in order to accurately assess the computational complexity. This is particularly true for the material design considerations of power consumption and silicon area.

Another factor complicating a direct comparison of computational complexity is that, even for the same algorithm, many possible architectural implementations exist. For example, just a sampling of the many architectural implementations for the sphere detector, i.e. the LSD with $\ell = 1$ [47], are found in [35, 18, 19, 80, 87, 64]. Given that even for the same detection algorithm there are many architectural implementations means that a comprehensive complexity analysis falls outside the scope of this dissertation. However, we still desire the ability to compare algorithms using computational complexity as a salient design criterion. We therefore require a simple and architecture independent metric for computational complexity that is useful in guiding our algorithmic construction.

One architecture independent metric for computational complexity is the number of real operations, be they fixed or floating point operations. The number of real operations, while often cumbersome to compute, is a useful metric for computational complexity across a wide range of algorithms. On occasion in this dissertation we consider the number of real operations, although we prefer an easier to calculate metric. Fortunately, tree-based detectors possess a fundamental metric of computational complexity that is easily calculated. We now discuss this metric.

A well-suited measure of computational complexity for tree-based detectors is the number of branch metrics computed during a tree search. Specifically, the number of branch metric computations is invariant to the architectural implementation, well accepted, and easy to calculate relative to other previously mentioned metrics and provides valuable insight into the overall system complexity [19, 108, 8, 68].

The number of branch metric computations corresponds to an upper bound on the number of visited nodes in the detection tree, since visiting a node in the tree requires calculating the corresponding branch metric. Using the number of visited nodes was proposed (and gained increased popularity due to the one node per cycle hardware implementation) in [19]. The advantage of using the number of branch metric computations, rather than the number of nodes visited, is that computing branch metrics avoids a complexity comparison that is unbalanced in favor of schemes that calculate metrics for nodes they later discard (as, e.g. the M algorithm [3]) as opposed to algorithms that only calculate metrics for nodes they do visit (as, e.g. the Schnorr-Euchner sphere decoder).

The disadvantages of using the number of branch metric computations as a measure of complexity is that it does not tell the entire story. First, the number of branch metric computations does not explicitly tell us about the time required to search through the detection tree or the memory required to store nodes in a stack. Second, the number of branch metric computations omits the complexity of the preprocessing algorithm. Such an omission may not be acceptable for fast-fading scenarios where the computational complexity of the detection ordering dominates, but is more appropriate for scenarios where the coherence time of the channel is long. Despite these disadvantages, the aforementioned benefits (i.e. invariance to the architectural implementation and ease of calculation) manifest the utility of using the number of branch metric computations as a complexity metric.

A first example of the utility of using the number of branch metric computations as a measure of computational complexity is to measure the complexity of the worst-case brute-force detector. Specifically, the brute-force detector must compute all $\left(q^{N_t+1} - q\right) / (q-1)$

branch metrics for a given detection tree. We can do better. We next quantify how much better.

2.5.2 Computational Complexity Bounds and Non-bound Reference Complexities

An ideal communications receiver would enable capacity achieving performance while requiring a negligible amount of computational complexity. Such a receiver is, of course, impossible to realize. For this reason, establishing computational complexity bounds and non-bound reference complexities is of great practical importance. The number of branch metric computations allows us to establish the following bounds and non-bound reference complexities for tree-based detectors. These reference complexities are useful for system engineers designing MIMO receivers:

- A lower bound on the number of branch metric computations required to ensure the JML solution and fixed number of counterhypotheses to this solution, up to and including ωN_t counterhypotheses.
- The maximum and minimum computational complexities required to obtain max-log optimal performance, i.e. max-log optimal complexity bounds.
- A 99th percentile computational complexity for the minimum number of branch metrics required to find the ℓ lowest cost leaf nodes in the detection tree.

We next discuss each of these ideas in detail.

Minimum Spanning Tree Bound: This is a lower bound on the number of branch metric computations required to ensure the JML solution and a fixed number of counterhypotheses to this solution, up to and including ωN_t counterhypotheses. This bound can be derived by considering a *minimum spanning tree*, where we define the minimum spanning tree to be a detection tree with the minimum number of branches required to generate one leaf node

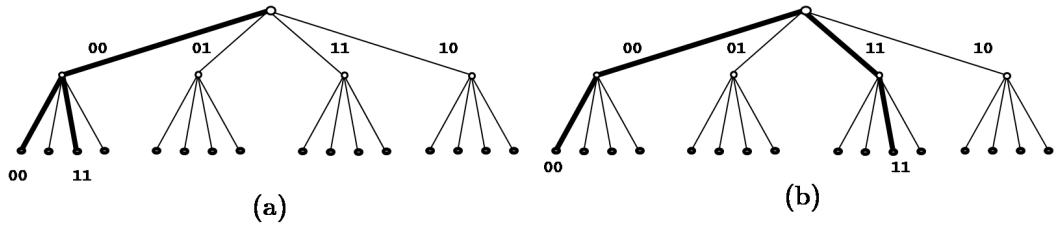

Figure 2.10: Example of a minimum spanning tree for (a) $C = 2 \Rightarrow (\mu = 3)$ and (b) $C = 4 \Rightarrow (\mu = 4)$ for a 2×2 MIMO system employing QPSK transmission.

and an additional $C - 1$ counterhypotheses. Consequently, when using this definition, C may not exceed $1 + \omega N_t$.

The minimum spanning tree bound for $C = 2$ consists of the JML solution and a counterhypothesis for exactly one bit, i.e. the bit pattern for the JML solution and the bit pattern for the lone counterhypothesis differ in only one position. Fig. 2.11-a depicts an example minimum spanning tree when $C = 2$ for a system with 2 input streams and QPSK modulation. The minimum number of branch metrics required to obtain the JML solution is N_t. Given $C = 2$, ensuring that exactly one counterhypothesis is found, and simultaneously computing the minimum number of branch metrics requires that the leaf node corresponding to the counterhypothesis be a "sibling"node of the JML solution at the final layer in the detection tree. For this case, the minimum spanning tree corresponds to $\mu = (N_t - 1) + 2 = N_t + 1$ branch metric computations (c.f. Fig. 2.11-a). Note, however, that the performance of a detector simply employing the minimum spanning tree would suffer relative to an algorithm generating the C nodes with minimum metrics (as, e.g. the LSD). Fig. 2.11-b depicts the minimum spanning tree for the same system as Fig. 2.11-a except that $C = 4$. For $C = 2$ the minimum spanning tree possesses three branches, whereas for $C = 4$ the minimum spanning tree possesses four branches.

The minimum spanning tree bound, in terms of the number of branch metric computations μ as a function of the parameters C, ω and N_t, is given by:

$$\mu = \begin{cases} N_t & C = 1 \\ N_t + 1 & C = 2 : 1 + \omega \\ N_t + 2 & C = 2 + \omega : 1 + 2\omega \\ \cdots & \cdots \\ 2N_t - 1 & 2 + (N_t - 2)\omega : 1 + \omega(N_t - 1) \\ 2N_t & C = 2 + (N_t - 1)\omega : 1 + \omega N_t \end{cases} . \tag{2.17}$$

The fact that $\mu = N_t$ for $C = 1$ is obvious: One leaf node requires a branch metric from each of the N_t layers in the detection tree. The minimum spanning tree for the case where each bit has a counterhypothesis, i.e. $C = 1 + \omega N_t$, corresponds to the situation where the bit pattern corresponding to each leaf node in the detection tree is the logical negation of the other, e.g. 010010 and 101101. Consequently, any two leaf nodes in the tree whose bit patterns are the boolean negation of each other represent the potential minimum spanning tree for $C = 1 + \omega N_t$. This implies $\mu = 2N_t$.

Max-Log Optimal Bounds: The number of leaf nodes required to compute the soft-output for max-log optimal detection ranges from a minimum of 2, in the case where, for all bit positions, the same candidate vector is used to provide the counterhypothesis to the JML decision, to a maximum of $\omega N_t + 1$, in the case where, for each bit position, a different leaf node provides the counterhypothesis. In the former case, which is unlikely, at least $\mu = 2N_t$ branch metrics must be computed. We will refer to this complexity as the "max-log best case" computational complexity. Note that this best case max-log optimal complexity corresponds to the minimum spanning tree bound for $C = \omega N_t + 1$. The "max-log worst case" computational complexity is found by considering the spanning tree for the case where each bit position requires a unique leaf node as the best counterhypothesis to the JML estimate. The resulting computational complexity for the max-log worst case is given

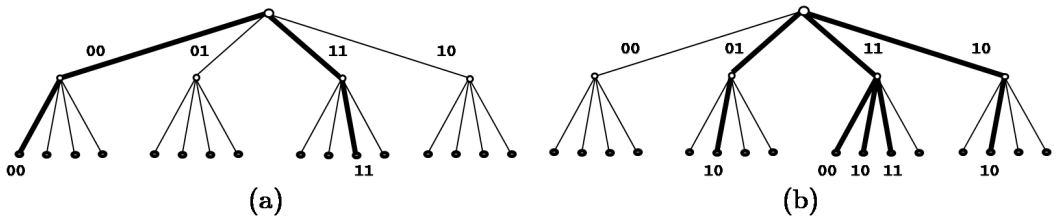

Figure 2.11: Example of an (a) best case max-log complexity ($\mu = 4$) and (b) worst case max-log complexity ($\mu = 8$) for a 2×2 MIMO system employing QPSK transmission.

by:

$$\mu = \sum_{1 \leq i \leq N_t} 1 + i\omega = N_t \left(1 + \frac{\omega\,(N_t + 1)}{2}\right). \tag{2.18}$$

Fig. 2.11-a and Fig. 2.11-b depict the best case and worst case max-log optimal complexities, respectively, for a 2×2 MIMO system employing QPSK transmission.

99**th Percentile Computational Complexity:** For the 99th percent computational complexity, we use the LISS as a practical algorithm for finding the ℓ leaf nodes in the detection tree with minimum costs. We elect to use the LISS algorithm because in [111] the LISS employing a Schnorr-Euchner enumeration strategy computes the minimum number of branch metrics necessary to find a given number of hypotheses on the transmit signal which maximize the a posterior probability. It it thus the tree search scheme with lowest complexity for solving the soft output detection task *optimally* for a given list size, where optimality is defined as maximizing the a posteriori probability. If the number of branch metric computations is the sole measure of complexity (i.e. ignoring the storage and sorting overhead of the LISS which are often high), it would therefore make no sense to employ any other tree search algorithm – which can only require more branch metric computations to achieve the same goal.

One drawback for the LISS is that its complexity is variable and so, in the worst case, the number of branch metrics computed by the LISS is equivalent to the exponential complexity of the brute force max-log optimal detector. A useful metric on this variable complexity is a 99th percentile metric on the number of branch metric computations calculated.

The 99th percentile metric is used to denote the number of branch metric computations calculated in at most 1% of the tree search realizations for a list of length ℓ. Using the 99th percentile (as opposed to, say, the 90th percentile) is motivated by the fact that a LISS whose complexity is upper bounded to this number of branch metric computations will stop its search prematurely in at most 1% of the cases. For state-of-the-art error correction codes, this will have a negligible impact on performance [108, 113]. We will refer to the 99th percentile metric as the 1% *upper bound* on computational complexity.

We conclude by observing that the 99th percentile computational complexity must be found via simulation and is therefore non-deterministic. Consequently, we do not depict a detection tree for this reference complexity.

Genie Spanning Tree: Consider the situation of a genie tree search, i.e. the ℓ minimum cost leaf nodes are known a priori. Pruning this tree to only consist of nodes that are ancestor nodes (e.g. parent or grandparent nodes) for any of the ℓ minimum cost leaf nodes. This genie spanning tree differs from the previously described minimum spanning tree bound in so far as the generated branches for the genie spanning tree are not automatically drawn from the lowest possible layers in the tree. Rather, the ℓ minimum cost leaves in the tree are found and all unnecessary branches are pruned. The remaining paths are used to calculate the total number of branches in the detection tree.

2.5.3 Results

For simulation we use a setup equivalent to the one in [47]: transmission occurs over a spatially and temporally i.i.d. fading 4×4 MIMO channel using 4-QAM and 64-QAM modulation alphabets. The information block size (including tail bits) is 9216 bits, using a rate $1/2$ PCCC based on $(7_R, 5)_{\text{octal}}$ convolutional codes for channel coding and 8 internal iterations of logMAP decoding, where R denotes which generator is in the denominator. The LLR magnitudes for bits without a counterhypothesis were set to the values found in

Table I in [70], for the 4×4 MIMO setup with 4-QAM and 64-QAM transmission[2].

The choice of the coding scheme is relevant to the overall system performance in MIMO detection. If near-capacity performance is desired then the channel code has to be designed to fit the EXIT characteristic of the detector [90], and multiple iterations between the detector and decoder are required. In this work we use the aforementioned system setup for ease of comparison with previous works, e.g. [47].

Fig. 2.12 depicts the complexity bounds outlined in the previous section for a 4×4 MIMO channel with 4-QAM transmission using the simulation setup just described, where equalization is performed using the zero-forcing (ZF) sorted QR decomposition (SQRD) [103] ordering and performance is measured as the SNR required to obtain a bit error rate (BER) of 10^{-5}. The lowest complexity curve, denoted with diamond markers, is the minimum spanning tree, where the marked points correspond to $C = \{2, 4, 8, 16\}$. For all other curves each marker corresponds to a unique list length $\ell = \{2, 4, 8, 16\}$, where the list used is the minimum cost list. The highest complexity curve in Fig. 2.12, denoted with a solid curve and upward facing triangular markers, is the 99% percentile computational complexity for the LISS. The average LISS complexity is also shown, using a dark dashed curve with square markers. Also shown are the best and worst case max-log optimal detector complexities, the genie spanning tree mean and genie spanning tree 99% percentile complexities. The best and worst case max-log optimal computational complexity for the given configuration correspond to $\mu = 8$ and $\mu = 24$, respectively.

As becomes clear from Fig. 2.12, the variance in complexity of the LISS is relatively small. Specifically, there is roughly a factor of 2 between the average and the 99th percentile complexity. This is consistent with results in [108]. Note, however, that for a list size of 16, the 99% percentile LISS complexity is already around $\mu \approx 150$ branch metric computations – about $2/3$ of the brute force max-log optimal detector. Considering

[2]Justification and the original derivation for these clipping levels can be found in [108]. Additionally, chapter 5 describes the LLR clipping problem in detail.

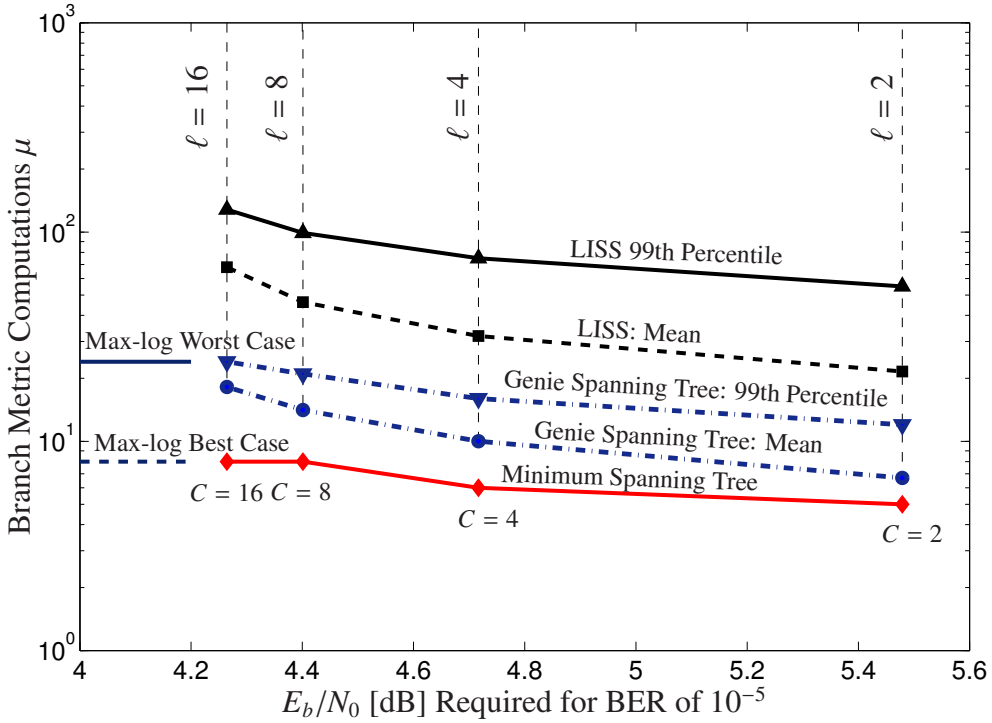

Figure 2.12: Complexity Bounds for 4×4 MIMO in Rayleigh fading using 4-QAM transmission and ZF-SQRD equalization.

the relative complexity of the practical LISS algorithm and the genie spanning tree for the LISS, there is roughly a 3-4 times complexity difference between the two cases if the average complexity is evaluated, and around a factor of 5, if the 99% LISS complexity is considered to be the most relevant design criterion.

As we have seen, the use of a minimum mean-square error (MMSE) effective channel matrix for detection ordering can reduce the complexity required to find a list of length ℓ [103]. However, when using MMSE preprocessing it is important to calculate LLR values using unbiased detection (2.19) to avoid performance degradation.

Fig. 2.13 depicts results for the same system as Fig. 2.12, except that we use MMSE-SQRD preprocessing. The minimum spanning tree does not change, but results for the remaining curves vary dramatically. Additionally, the best and worst case max-log optimal computational complexities do not change. However, the complexities of the reference complexities for the LISS (i.e. genie spanning tree mean and 99% percentile computational

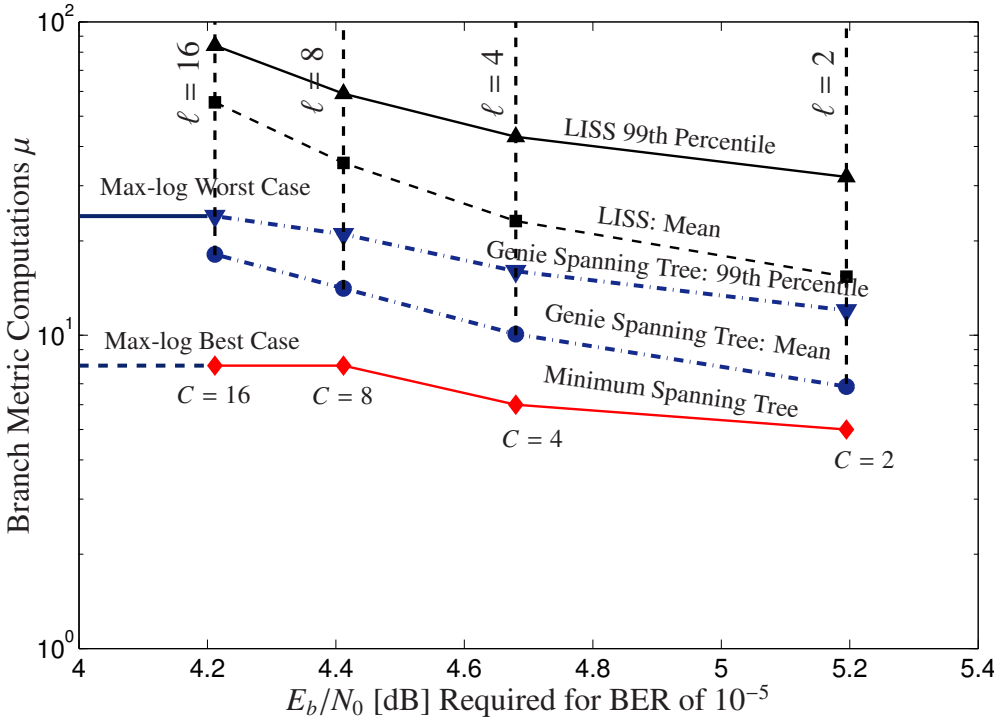

Figure 2.13: Complexity Bounds for 4×4 MIMO in Rayleigh fading using 4-QAM transmission and MMSE-SQRD equalization.

complexities and LISS detector mean and 99% percentile computational complexities) undergo a reduction in the computational complexity due to the MMSE preprocessing. The impact of the MMSE-SQRD ordering is even more substantial for tree search schemes with higher variability in detection complexity than the LISS, such as the sphere decoder, and/or for larger problem sizes [108]. We next consider the situation where we employ a 64-QAM modulation alphabet.

Fig. 2.14 depicts the same reference complexities as in the previous diagrams, but for the case of 64-QAM transmission using MMSE-SQRD equalization. For this higher modulation scenario, the difference between the genie spanning tree (genie LISS) and the actual number of branch metric computations for LISS detection are more evident – for a list length of 64, there is a factor of around 10 between the two cases, for the 99% percentile computational complexities. The best case max-log optimal computational complexity remains unchanged at $\mu = 8$ while the worst case grows, due to the increased constellation size, to $\mu = 64$.

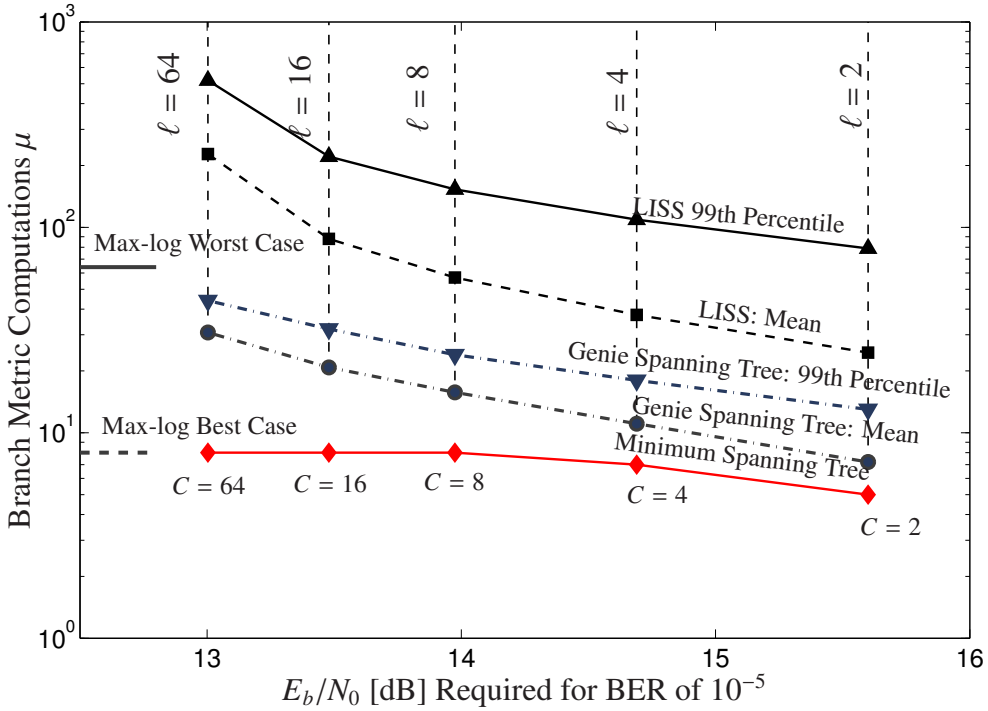

Figure 2.14: Complexity Bounds for 4×4 MIMO in Rayleigh fading using 64-QAM transmission and MMSE-SQRD equalization.

2.5.3.1 Variable vs. Fixed Computational Complexity

As we will see throughout this dissertation, MIMO detection algorithms can possess either variable or fixed computational complexity. For variable complexity algorithms, the number of branch metric computations is a random variable whose probability mass function is a strong function of the SNR. Typically, as the SNR increases the number of branch metric computations decreases. For many algorithms, the worst-case complexity can be very high (i.e. comparable to the worst-case brute-force complexity), but the average complexity can be extremely small.

Because the number of branch metric computations performed by the LSD is a random variable whose probability mass function is a strong function of the SNR, as the SNR increases the number of branch metric computations decreases. While the worst-case complexity of the list sphere detector can be very high (i.e. comparable to the brute-force detector), the average complexity can be extremely small. We will now illustrate this using

an Example 1.

Example 1: LSD Branch Metric Example: Consider an 8-input, 8-output memoryless spatially and temporally i.i.d. fading channel in AWGN, and assume the inputs are independent uncoded 64-QAM symbols. An exhaustive search would have to consider $64^8 = 2^{48} > 2.8 \times 10^{14}$ leaf nodes. Let N denote the number of branch metric computations performed by a the list sphere detector, assuming the inputs are ordered according to the near-optimal Minimum Mean-Square-Error (MMSE) sorted-QR decomposition [103]. The probability mass function for N is easy to estimate using simulation. Two examples are shown in Fig. 2.15. When the SNR is 20 dB as in Fig. 2.15-a, the list sphere detector computes $\bar{N} = 550.7$ and $\bar{N} = 2010.4$ branch metrics on average for $\ell = 7$ and $\ell = bN_t + 1 = 49$, respectively. When the SNR is 30 dB, as shown in Fig. 2.15-b, the list sphere detector computes only $\bar{N} = 462.2$ and $\bar{N} = 1861.4$ branch metrics on average for $\ell = 7$ and $\ell = 49$. From these results we observe that the average number of branch metric computations decreases with an increase in SNR and increases with the list length.

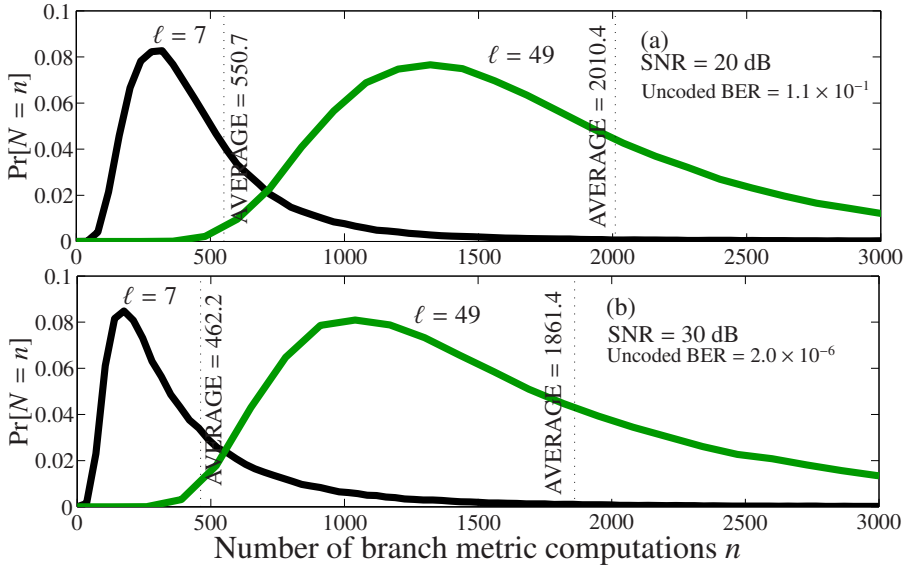

Figure 2.15: Estimated probability mass functions for N the number of branch metrics computed by the depth-first list sphere detector at (a) SNR = 20 dB and (b) SNR = 30 dB, assuming 8×8 Rayleigh-fading channel with 64-QAM inputs. These results were found by simulating the list sphere detector $T = 2 \times 10^5$ times, with independent noise, channel, and symbol realizations for each trial, then estimating the pmf for N according to $\Pr[N = n] = I_n/T$, where I_n is the number of trials for which n nodes were visited.

In contrast to variable complexity approaches, tree-based algorithms which calculate a fixed number of branch metric computations exist. For these algorithms, the number of branch metrics can be expressed deterministically. Our objective in this dissertation is the construction of soft-output MIMO detection algorithms possessing low and fixed computational complexity. Specifically, in our setting of tree-based list detectors we state our overarching dissertation goal as:

> **Dissertation Objective:** Find ℓ leaf nodes in the tree with low path cost using low and fixed computational complexity.

2.6 Ordering of the Channel Matrix – Preprocessing

The mapping of layers in the detection tree to transmitted symbols is a critical factor in determining either the performance or the computational complexity (or both) for soft-output MIMO detection. This mapping, which is a direct consequence of the channel matrix ordering during the QR decomposition, deserves careful attention.

An ordered QR decomposition leads to the detection tree layer mapping, i.e. $\mathbf{HP} = \mathbf{QR}$, where $\mathbf{P}$ is a permutation of $\mathbf{I}_{N_t}$. The BLAST ordering [34] is the optimal detection order for hard-output decision-feedback detection, where only a single path of the tree from root to leaf is traversed, since it maximizes the SNR at each layer. An attractive alternative to the BLAST ordering is the sorted QR decomposition (SQRD) [102], which achieves nearly the same hard-output decision-feedback performance as the BLAST ordering with reduced complexity. Specifically, the BLAST ordering is roughly twice as complex as the SQRD, but with improved performance as a result of its better ordering criterion [95]. Other useful orderings which we will discuss in more detail in the next chapter include the one employed by the parallel detector (PD) [58], the fixed-complexity sphere decoder (FSD) [9, 51], and B-Chase preprocessing [96, 97, 95, 98].

All of the preprocessing algorithms for which we present results in this dissertation have a fixed computational complexity on the order of $\Theta(N_t^3)$. We note, however, that highly effective preprocessing algorithms with variable computational complexity do exist. An important class of variable complexity preprocessing algorithms are based on using the idea of lattice-reduction-aided (LR-aided) detection [46, 56, 57], with [57] achievable in polynomial time. Because we focus on fixed-computational complexity in this dissertation, we will logically only consider preprocessing algorithms with fixed-computational complexity.

The use of a minimum mean-square error (MMSE) effective channel matrix when performing the ordered QR decomposition can further improve performance and/or reduce complexity [103]. Unlike the ZF detector, which minimizes interference while neglecting noise effects, the MMSE linear detector achieves an optimal balance of noise enhancement and interference suppression [12]. While MMSE detection is well known to be an effective detection technique, it is important to compute the path metrics for the detection tree in an unbiased way. As reported in [109] this is accomplished by considering the following system model:

$$\left\|\bar{\mathbf{r}} - \bar{\mathbf{H}}\mathbf{a}\right\|^2 = \left\|\begin{bmatrix}\mathbf{r}\\ \mathbf{0}_{N_t\times 1}\end{bmatrix} - \begin{bmatrix}\mathbf{H}\\ \sigma\mathbf{I}_{N_t}\end{bmatrix}\mathbf{a}\right\|^2 = \|\mathbf{r} - \mathbf{H}\mathbf{a}\|^2 + \sigma^2\|\mathbf{a}\|^2, \tag{2.19}$$

where $\bar{\mathbf{r}}$ is the effective received signal and $\bar{\mathbf{H}}$ is the MMSE effective channel matrix, and $\sigma^2 = E/(N_t N_0)$. It is the term $\sigma^2\|\mathbf{a}\|^2$ that must be removed for unbiased MMSE detection of the path metrics.

2.7 Enumeration for Breadth-First Detection

The order in which child nodes are extended from a parent nodes in the detection tree is referred to as enumeration. Enumeration can have a significant impact on the overall computational complexity of the tree search. A Fincke-Pohst [32] enumeration considers the child nodes in a natural order, say from left to right, with no regard to their weights. Fewer

nodes will be visited if a Schnorr-Euchner [76] enumeration is adopted, which explores child nodes in an order determined by their weights, with the best first. All results and algorithms in this dissertation assume Schnorr-Euchner enumeration.

A recently proposed approach for implementing a Schnorr-Euchner enumeration is the one in [19], where for QAM alphabets the constellation is mapped onto several concentric circles each corresponding to a phase shift keying (PSK) alphabet. Within each PSK subset the preferred child node is established and the PSK subsets are compared to determine the next child node from the QAM parent node. For QPSK modulation only one subset is necessary and thus no comparison is required. For 16-QAM and 64-QAM three and nine parallel subset instantiations are required, respectively.

Enumeration is an important consideration for breadth-first detection as well, but here the goal is to enumerate the b_i best nodes at the given layer in the detection tree, from each of the retained survivor nodes. An enumeration technique well-suited for breadth-first detection was proposed in [63]. This is accomplished by mapping the detection problem onto a geometrical approach: For a known relative position of the received symbol to an initial reference point within a given grid, an unique sequence of favorable nodes can be identified. More specifically, this enables a heuristic determination of favorable child nodes without their calculation, by only requiring a few inexpensive comparisons for a given parent node, independent of the constellation size [63]. The result of using this approach is that the number of required metric calculations is reduced to a minimum of one calculation per examined node. Consequently any sort operations for node selection are redundant and our use in this dissertation of the number of branch metric computations is supported as an acceptable measure of computational complexity.

2.8 Discussion

Before advancing to the contributions in this dissertation, let us first step back to compare the universe of soft-output MIMO detection algorithms. Table 2.1 classifies soft-output

MIMO detectors into four quadrants based on whether or not a list $\mathcal{L} \subseteq \mathcal{Z}$ and/or a tree is used in the detection process. In this dissertation we elect to focus on the upper left quadrant of this grid, i.e. tree-based list detectors. We made the decision to focus on list detectors because they are a viable solution to the suboptimal soft-output MIMO detection problem with lower computational complexity than an exact solution. Detection algorithms without a list are feasible, e.g. [78, 24, 15], but their performance often is far from the exact solution. Soft-output detection algorithms that do not use a list and obtain strong performance therefore represents a potential area for further research. Our decision to focus on tree-based approaches for finding the list is based on the fact that tree-based detectors efficiently solve the list detection problem and have a desirable performance-complexity trade-off.

Table 2.1: Classification of soft-output MIMO detection algorithms.

	LIST	NO LIST
TREE	List Sphere Detector [47] List Sequential Detector [11, 42] Soft M Algorithm [3, 68]	
NO TREE	Monte Carlo Methods [30, 38, 31] Semidefinite Programming [86, 73] Soft Sphere Projection [78]	Soft ZF/MMSE [29, 20] Soft Interference Cancelation[24, 15]

2.9 Further Reading

Other soft-output tree-based MIMO detection algorithms include iterative tree search [28], the list fixed-complexity sphere detector [8], and the soft fixed-complexity sphere detector [6]. Detailed treatment of tree search algorithms can be found in [3] and [55]. Additionally, as detailed in Table 2.1 there are many soft-output MIMO detection algorithms that fall outside the scope of list and tree based detection algorithms. Some of these approaches include Monte-Carlo methods [30, 38, 31], semidefinite programming [86, 73], space-time Chase detection [59], and soft sphere projection [78]. Details on architectural issues

pertaining to soft-output MIMO detection algorithms are provided in [39, 22, 87]. Recent advances relating to Schnorr-Euchner enumeration can be found in [19] and [63].

2.10 Summary

In this chapter we motivated the problem of soft-output MIMO detection. MIMO channels enable greater reliability in the presence of fading and/or increased throughput via spatial multiplexing gains. Because MIMO systems employ error control coding and soft-output detection algorithms achieve lower error rates in the presence of error control coding than hard-output detectors, the soft-output detection problem is critically important. While a variety of approaches for the solving the soft-output MIMO detection problem exist (see section 2.8), we motivated our reasoning for focusing on tree-based list detectors in this dissertation. Specifically, it is because these detectors are efficient and capable of near max-log optimal performance. As stated in this chapter, our dissertation objective is to find a list of low cost leaf nodes in the detection tree with low and fixed computational complexity. The remainder of this dissertation is used to achieve this objective.

CHAPTER 3

SMART-ORDERED CANDIDATE-ADDING ALGORITHM

3.1 Introduction

The design of multiantenna detection algorithms that simultaneously achieve low error rate performance and low computational complexity is a challenge. This challenge is exacerbated as the spectral efficiency increases. In fact, the computational complexity of the optimal multiantenna detector grows exponentially with increased spectral efficiency. In this chapter we propose a soft-output detection algorithm, known as the smart-ordered candidate-adding (SOCA) algorithm, that allows for multiantenna detection with low error rate performance *and* low computational complexity. In fact the SOCA algorithm not only obtains near max-log optimal error rate performance with low computational complexity, it does so with *fixed* computational complexity as well.

The SOCA algorithm employs a smart-ordered QR decomposition and parallel smart candidate adding to achieve its desirable performance-complexity tradeoff. The SOCA algorithm's fixed computational complexity is a function of the fact that it uses a breadth-first search of the detection tree to perform candidate enumeration. Moreover, the deterministic nature of the SOCA's computational complexity and a careful architectural implementation have the potential to produce a parallel architectural structure with a low and predictable latency.

In the next section we lay the groundwork for the construction of the SOCA algorithm by first combining the idea of smart candidate adding with breadth-first tree-based detection. We follow this motivation with the proposed SOCA algorithm in section 3.3. Next, we extend traditional ways of classifying breadth first detection algorithms to include allowing variable parameterization for each layer of the detection tree, as well as the inclusion of

parallel smart candidate adding in section 3.4. Results are presented in section 3.5, suggestions for further reading in 3.6 and a chapter summary in section 3.7.

3.2 Motivation: Towards Fixed-Complexity Smart Candidate Adding

Recall the breadth-first M algorithm from chapter 2 [3]. As any other breadth first scheme, it traverses the tree layer-by-layer; extending the b "best" children (with minimum branch metrics) from all nodes retained from the previous layer in the tree and subsequently maintains the m nodes with minimum path cost from the set of extended nodes. Consequently, the M algorithm can be parameterized by two scalar values, m and b.

Many detection algorithms are special cases of the classical M algorithm with specific parameterizations and preprocessing. The simplest example is the hard-output decision feedback detector, for which $m = b = 1$. With $b = q$ and $m = \infty$, where an m value of ∞ implies that all nodes extended are retained, the algorithm turns into the maximum complexity, brute-force, approach which enumerates all possible transmit vectors. For $b = q$, and arbitrary positive m, the M algorithm is also known as the K-best algorithm [101, 39].

As described in chapter 2, an effective way to improve the error-rate performance of soft-output MIMO detection algorithms is through the use of smart candidate adding. We next show how SCA can be incorporated into a breadth-first tree search by not only considering branch metrics for the detection tree, but also the corresponding bit mappings.

Recall from section 2.4.3.1 that SCA algorithms search for candidate lists that either exactly or approximately include (1) $\hat{\mathbf{a}}^{\text{MAP}}$ and (2) $\hat{\mathbf{a}}_j^{-\text{MAP}}$ for each of the $j \in \{1, \ldots \omega N_t\}$ transmitted bits [62]. Incorporating SCA into a breadth-first detector was proposed in [110]. This so-called SCA-M algorithm combined the classical M algorithm with an approximate candidate adding algorithm. Specifically, the SCA-M algorithm can be broken into two stages [110]:

- **Stage 1:** is a breadth-first list tree-search for the MAP estimate. This search, which finds a list of candidates with low cost (and which does not necessarily include the

exact MAP solution) can potentially search the entire signal set $\mathcal{Z}$ depending on the selection of $m^{(1)}$ and $b^{(1)}$, where the superscript (1) is used to denote stage 1. The parameters $m^{(1)}$ and $b^{(1)}$ should be selected so that the MAP estimate is an element of the stage 1 list, denoted $\mathcal{L}_1$, with high probability. The list size at the end of stage 1 is $\ell_1 = |\mathcal{L}_1|$.

- **Stage 2:** searches for counterhypotheses for bits in $\mathcal{L}_1$ that do not yet have a counterhypothesis, i.e. all candidates in $\mathcal{L}_1$ possessing the same logical bit (0 or 1) for each given bit $j \in \{1, \dots \omega N_t\}$. Consequently, stage 2 searches over only a constrained signal set $\mathcal{Z}_j^{\bar{}\text{MAP}}$ for each bit $j \in \{1, \dots \omega N_t\}$ in $\mathcal{L}_1$ without a counterhypothesis, and is referred to as a *constrained search*. An M-Algorithm with $m^{(2)}$ and $b^{(2)}$ is used to find the list $\mathcal{L}_2$, where the final list for the SCA-M algorithm is $\mathcal{L} = \mathcal{L}_1 \bigcup \mathcal{L}_2$.

Achieving low error rates using the SCA-M algorithm is possible even when the stage 2 list length, as well as the parameter $m^{(2)}$ and $b^{(2)}$, are small. In fact, results in [110] show that low error rates are possible when $\ell_2 = \mathcal{L}_2 = m^{(2)} = b^{(2)} = 1$. This implies that once the MAP estimate is found, the search for counterhypotheses need not require high computational complexity.

A key drawback of the SCA-M algorithm, like many prior SCA algorithms [62, 105, 94], is that it requires multiple searches through the detection tree. Additionally, we observe that depending on the stage 1 list $\mathcal{L}_1$, stage 2 constrained searches may be required for all $j \in \{1, \dots \omega N_t\}$ bits (only possible when $\ell_1 = 1$) or a stage 2 constrained search may not be required (in the case where all ωN_t bits in $\mathcal{L}_1$ have a counterhypothesis). This observation implies that stage 2 for the SCA-M algorithm has a variable computational complexity that ranges from requiring between 0 and ωN_t constrained searches, depending on the number of counterhypotheses in $\mathcal{L}_1$ relative to the minimum cost estimate in $\mathcal{L}_1$. Out of the two aforementioned drawbacks of the SCA-M algorithm, namely multiple searches through the detection tree (i.e. stage 1 and stage 2) and the variable complexity of stage 2, we

consider the multiple searches through the detection tree to be the more critical drawback because it leads to higher than necessary computational complexity. In the next section we propose a single-tree-search low- and fixed- computational complexity algorithm, known as the SOCA algorithm, that solves both the problem of multiple searches and variable computational complexity.

3.3 Fixed Complexity Smart Candidate Adding - SOCA Algorithm

The computational complexity of breadth-first smart candidate adding can be fixed by performing constrained searches for counterhypotheses *concurrently* with the MAP estimate as the search proceeds through the detection tree, rather than through supplemental searches. This is similar to the variable-complexity depth-first "parallel sphere detector" approach taken in [50] and improved upon in [87]. Enabling concurrent counterhypothesis searches requires a single pass through the detection tree. We accomplish this by electing to perform a single pass of the detection tree using the fixed-complexity breadth-first M-Algorithm. A direct consequence of this decision is that the counterhypothesis for a bit of interest can only be found relative to the best *partial* MAP (PMAP) estimate at the current level in the detection tree – as opposed to the exact MAP estimate in prior approaches. We therefore obtain a complexity reduction by using this proposed *parallel* smart candidate adding (PSCA) approach at the cost of a small loss in performance.

We next propose an algorithm for solving the soft-output MIMO detection problem called the *smart ordering and candidate adding* algorithm. The SOCA algorithm consists of two stages and allows for a tradeoff between error-rate performance and computational complexity. These two stages are (1) a preprocessing stage and (2) a core processing stage. The preprocessing stage is used to determine the mapping between layers in the detection tree and the transmitted vector of information symbols. The core processing stage finds the list $\mathcal{L}$, the output of the SOCA algorithm. Because the preprocessing algorithm can be considered a performance enhancement, we begin by describing the core processing.

3.3.0.2 SOCA Core Processing

The SOCA algorithm finds $\mathcal{L}$ using a standard detection tree like the one described in section 2.3.1. The foundation of the SOCA algorithm is a simple breadth-first strategy for searching the tree that is closely related to the M algorithm [3]. Like the M algorithm, the SOCA algorithm moves through the tree one layer at a time, discarding all but a subset of "surviving" nodes from a given layer before moving to the next. One difference is how many surviving nodes are retained at each layer; rather than keeping this fixed, the SOCA algorithm allows for the possibility that this number m_i may depend on the layer index i. Another difference is how many children from each surviving node are extended; rather than keeping this fixed, the SOCA algorithm allows for the possibility that this number b_i may also depend on the layer index i.

We proposed the idea of varying the number of surviving nodes extended from retained parent nodes for each layer in the detection tree, i.e. the b_i's, in [68]. We called the detector in [68] the channel-based layer-adaptive M (CLAM) algorithm. The CLAM algorithm attempted to allocate the most computational resources to the layer most likely to be in error based on each of the per-layer SNRs. Hard-output results demonstrated that the CLAM algorithm is on average less complex than the M algorithm while achieving significantly improved performance. A key finding from our work on the CLAM algorithm is that, similar to the parallel detector and the fixed-complexity sphere detector, it is most important to allocate computational resources to the first (or first few) layers in the detection tree. Consequently, while the SOCA algorithm does not allocate the b_i's based on channel state information, the idea of varying these b_i's is motivated by the ideas we proposed in [68].

The SOCA algorithm builds upon its breadth-first foundation by inserting a new step. Before pruning away (if necessary) all but the m_i best surviving nodes from a current set of candidate nodes at layer i, the SOCA algorithm identifies the candidate node with the best metric as the partial MAP node. Once identified, the SOCA algorithm adds new nodes to the candidate set so that each of the ω bits corresponding to the current symbol a_i has

a counterhypothesis. Specifically, if $\hat{\mathbf{c}}^{\mathrm{PMAP}}$ denotes the ωi-bit pattern corresponding to the node with the best metric, with the last ω of these bits corresponding to a_i, then the SOCA algorithm adds the ω sibling nodes of the partial-MAP node by simply flipping each of the last ω bits of $\hat{\mathbf{c}}^{\mathrm{PMAP}}$ in turn. This bit flipping strategy was chosen because of its low complexity, despite the facts that (1) the counterhypotheses so generated may not be the ones having the best metric, and (2) a counterhypothesis for the bit in question may already be represented in the candidate set. Once added, these counterhypotheses may be immediately pruned, although our results indicate that for MIMO system sizes at least as large as 4×4, the performance benefit of protecting these added counterhypotheses combined with the increased computational complexity of a candidate sort to determining which nodes to prune mean that protecting all enumerated nodes for the SOCA algorithm is advised.

In the case of gray mapping and QAM alphabets, while the sibling nodes are not guaranteed to have small metrics, they are likely to have small metrics because at least two, and at most four, of the siblings are nearest neighbors of the transmitted symbol estimate. In fact, exactly two, three and four siblings correspond to nearest neighbors in the case of an estimated corner, border, and interior point for a gray-mapped QAM alphabet, respectively.

Because the SOCA algorithm is breadth-first and possesses fixed computational complexity it lends itself well to architectural implementation. For reasons we will discuss later, namely the preprocessing algorithm, the SOCA algorithm does not need to consider the problem of missing counterhypotheses to the children extended from the root node, i.e. the SOCA algorithm does not concern itself with missing counterhypotheses at the first layer of the detection tree.

Like any other algorithm built on the foundation of the M algorithm, the tree for the SOCA algorithm can be pruned using a sort-and-select procedure, reducing the number of nodes to the m_i best nodes whenever m_i is less than the number of nodes enumerated at the current layer in the tree. When m_i is larger than the number of nodes extended at a given layer in the tree, this sort-and-select stage is omitted for reduced complexity. In this case,

instead of a sort-and-select procedure, all that is required is to determine the minimum cost node at each layer in the tree. If a sort-and-select is necessary, one option is the heapsort algorithm [55]. The heapsort algorithm, at the i^{th} layer of the tree, can be achieved with computational complexity $\Theta(m_i \log m_i)$.

A concise description of the `SOCA` algorithm is provided in Fig. 3.1. In summary, the SOCA algorithm takes as input the received signal $\mathbf{r}$, the MIMO channel $\mathbf{H}$, the constellation $\mathcal{A}$, and two vectors $\mathbf{b} = [b_1\ b_2\ \ldots\ b_{N_t}]$ and $\mathbf{m} = [m_1\ m_2\ \ldots\ m_{N_t}]$, where $\mathbf{b}$ grows the tree by adding nodes and $\mathbf{m}$ prunes the tree by deleting nodes. The set $\mathcal{F}$ is used to denote the surviving nodes at the current layer in the tree. We recommend keeping the elements of $\mathbf{b}$ small (1 if possible), with an exception for the first detection layer (i.e. $b_1 > 1$) due to the fact that the diversity order of the first symbol to be detected is $N_r - N_t + 1$ and a mistake here leads to error propagation. In many practically relevant system configurations $\mathbf{b} = [b_1\ 1\ \ldots\ 1]$ with b_1 set to between 25% and 50% of q yields excellent performance at very low complexity. The reason we do not need to set b_1 equal to q is because of our use of a smart-ordered QR algorithm, i.e. the first line in Fig. 3.1. Without lines 6 through 11 and the assurance of a smart-ordered QR decomposition in line 1, the rest of the pseudocode is simply the M algorithm with variable $\mathbf{b}$ and $\mathbf{m}$.

We will discuss the preprocessing for the SOCA algorithm momentarily. First, we provide a simple example of the core processing for a two transmitter system employing QPSK modulation. Specifically, we are interested in finding the leaf nodes resulting from searching the tree in Fig. 3.2-a for $\mathbf{b} = [3\ 1]$ and $\mathbf{m} = [3\ 5]$. The bit mappings for each decision are, from left to right, 00, 01, 11 and 10, as noted above and below the detection tree in Fig. 3.2-a, and the branch metrics are labeled on each branch for Fig. 3.2-a.

Fig. 3.2-b begins the algorithm by enumerating the $b_1 = 3$ best nodes from the root of the tree, i.e. the nodes corresponding to branch metrics of 2, 3 and 7, and excluding the branch with metric 8. Then, because $m_1 = 3$ and only three nodes have been enumerated, no pruning occurs. These three remaining nodes serve as parent nodes for the next layer in

Algorithm: SOCA
Input: $\mathbf{r}, \mathbf{H}, \mathbf{b}, \mathbf{m}$
Output: $\mathcal{L}$

$[\mathbf{Q}, \mathbf{R}, \mathbf{P}] = \text{SOQR}(\mathbf{H}, b_1)$
$\mathbf{y} = \mathbf{Q}^* \mathbf{r}$
$\mathcal{F}$ = root node
for $i = 1 : N_t$ **do**
    $\mathcal{F} = \cup_{\text{node} \in \mathcal{F}} \{b_i \text{ best children of node}\}$
    if $i > 1$ **then**
        for $j = (i-1)\omega + 1 : i\omega$ **do**
            Flip bit j of $\hat{\mathbf{c}}^{\text{PMAP}}$ and add the corresponding node to $\mathcal{F}$
        end
    end
    $\mathcal{F} = m_i$ best of $\mathcal{F}$
end
$\mathcal{L} = \mathbf{P}\mathcal{F}$

Figure 3.1: SOCA Algorithm Description.

the detection tree.

We continue on at the next layer in the tree by enumerating the $b_2 = 1$ best child node from each of the retained parent nodes, as shown in Fig. 3.2-c. This yields $3 \cdot 1 = 3$ leaf nodes in the detection tree. We then perform parallel candidate adding on the best estimate in the tree, i.e. 0000 as shown in Fig. 3.2-d yielding five leaf nodes. Because there are five leaf nodes and $m_2 = 5$ we retain all leaf nodes for our list and we are done. Observe that the bit patterns corresponding to the five nodes are 0000, 0010, 0001, 0111 and 1000. As desired, given that no pruning occurs (i.e. $m_2 = 5$), every bit, in the layers after the first, has an associated counterhypothesis. In fact, it turns out that for this example *all* bits have a counterhypothesis, although for the SOCA algorithm this is not generally guaranteed at the first detection layer. Observe that for this example the node 0001, with cost 7, is unnecessary because node 0111, with its lower cost of 6, serves as the counterhypothesis to the MAP node 0000 with cost 3 for both the second and fourth bits. For completeness we remark that node 0010 serves as the counterhypothesis for bit three and node 1000 serves

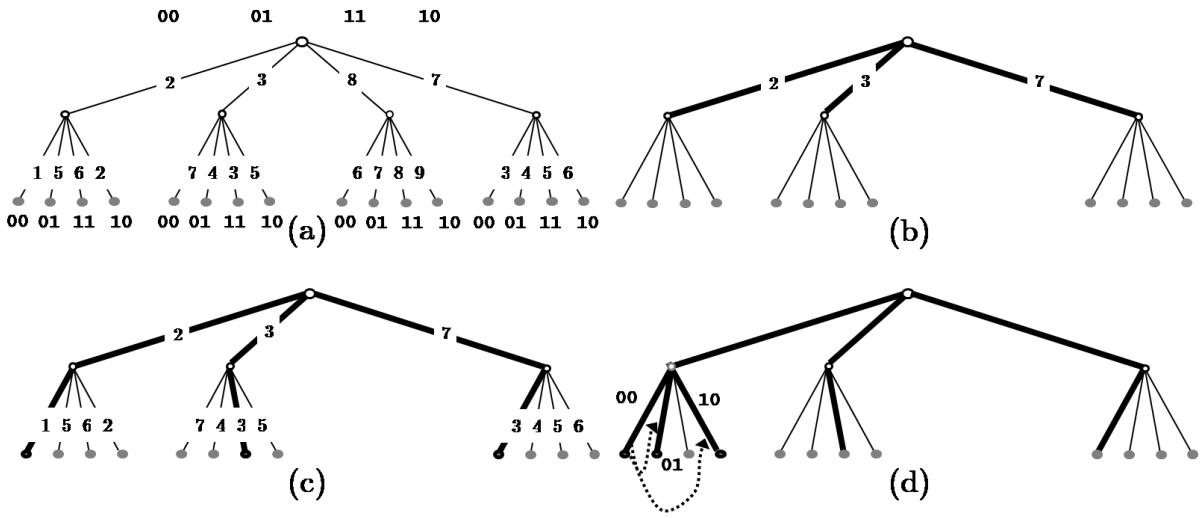

Figure 3.2: Example of SOCA algorithm for a 4-ary tree with two layers.

as the counterhypothesis for bit one.

3.3.0.3 Ordering of the Channel Matrix

As we have seen in previous chapters, the mapping of layers in the detection tree to transmitted symbols is a critical factor in determining either the performance or the computational complexity (or both) for soft-output MIMO detection. An ordered QR decomposition is used to achieve the desired ordering, i.e. $\mathbf{HP} = \mathbf{QR}$, where $\mathbf{P}$ is a permutation of $\mathbf{I}_{N_t}$.

The BLAST ordering [34] presented in chapter 2 is not generally optimal when more than one node is enumerated at any stage in the detection tree. For example, the parallel detector (PD) of [58] enumerates all q child nodes of the root node and extends each of these nodes using decision feedback to obtain q leaf nodes. The parallel detector works best when the *weakest* received signal component is detected first, so that its contribution is completely removed from the detection problem. Intuitively, this is because there is no possibility for an error to occur in a layer where all child nodes are enumerated. Consequently, it is desirable to enumerate all child nodes in the layer with the largest noise enhancement to minimize performance loss. In [9] the parallel detector ordering was extended by employing the weakest-first parallel detector ordering for layers where all q child

nodes are enumerated and the strongest-first BLAST ordering for all other layers. In [51] it was shown that the ordering of [9] maintains the diversity order of the maximum-likelihood detector with a fixed complexity and order $\Theta(q^{\sqrt{N_t}})$ if $N_r = N_t$, when all nodes in the first $\lfloor \sqrt{N_t} \rfloor$ layers are enumerated.

A detection order for cases where the number of child nodes to be enumerated from each parent is between 1 and q is given by the B-Chase detector [98]. B-Chase preprocessing has been shown to gracefully trade off between the opposing design goals of maximizing (as in the BLAST ordering) vs. minimizing (as in the PD ordering) the SNR of the first detection layer by allowing the ordering algorithm to consider an increase in the number of child nodes enumerated from the root node as an effective SNR gain for the receiver.

We now present a particular B-Chase preprocessing configuration that we found to perform well. We call this configuration the smart-ordered QR (SOQR) decomposition. A SOQR decomposition takes as inputs $\mathbf{H}$ and b_1 and produces the outputs $\mathbf{Q}$, $\mathbf{R}$, and $\mathbf{P}$. The key step in the SOQR is to determine which layer to detect first. This decision is a function of the per-layer SNRs and b_1. As b_1 is increased from 1 to q the layer selected to be detected first moves from the one with the highest SNR to the one with the lowest. This is done so that as b_1 is increased to approach q we order the detection based on the assumption that detection errors in the first layer in the tree are unlikely. Indeed, they are impossible when $b_1 = q$.

We propose that the index k of the first layer to be detected be chosen according to the B-Chase criterion [98]:

$$k = \begin{cases} \underset{n\in\{1,\ldots N_t\}}{\arg\max} \|\mathbf{Y}_n\|^2, & b_1 = q \\ \underset{n\in\{1,\ldots N_t\}}{\arg\max} \min\left\{ \frac{\gamma_{b_1}^2}{\|\mathbf{Y}_n\|^2},\ \frac{1}{\min_{s\neq n}\left\{\|\mathbf{Y}_s\|^2 - |g_{s,n}|^2\right\}} \right\}, & \text{otherwise} \end{cases} \tag{3.1}$$

where $\mathbf{Y}^* = \mathbf{R}^{-1}$ is determined by a QR decomposition of the channel matrix, i.e. $\mathbf{QR} = \mathbf{H}$. Additionally, $g_{s,n} = \mathbf{Y}_s^* \mathbf{Y}_n / \|\mathbf{Y}_n\|$, where $\mathbf{Y}_n$ is the nth column of $\mathbf{Y}$. The parameter $\gamma_{b_1}^2$ is the

effective SNR gain (see [98]) at the first detection layer when b_1 child nodes are enumerated from the root node. For QPSK transmission $\gamma_1^2 = 1$, $\gamma_2^2 = \gamma_3^2 = 2$ and $\gamma_4^2 = \infty$. Values of $\gamma_{b_i}^2$ for 16 and 64-QAM transmission are found in [98]. However, because the value for γ can be determined using a lookup table that is a function of the parameter b_1, the selection for b_1 does not influence the complexity of the SOQR decomposition. The complexity of (3.1) is dominated by computing the squared column norm $\|\mathbf{Y}_n\|^2$ a total of N_t times and the $N_t(N_t - 1)$ vector multiplications $\mathbf{Y}_s^*\mathbf{Y}_n$ to compute all $g_{s,n}$ values.

After selecting the index of the first layer to be detected, the remainder of the SOQR is essentially a SQRD [102], where the ordering of the first detection layer is forced. The SOQR can be achieved with complexity order $\Theta(N_t^3)$. Pseudocode for the `SOQR` algorithm is provided in Fig. 3.3. Pseudocode for the `SOQR` algorithm is provided in Fig. 3.3. Note that the forced ordering in line 1, the initialization of k_2 in line 3, and the forced ordering of lines 5-12 ensures the first layer detected is chosen according to (3.1).

In the next section we classify a number of hard- and soft-output breadth-first detection algorithms of which the SOCA algorithm is an example. Results for the SOCA algorithm, as well as many others detailed in the next section, will be presented following this classification.

3.4 Classifying Breadth-First Detectors

Clarifying and extending the presentation in the previous section, we propose to classify breadth-first detectors by specifying the following parameters individually for each layer in the detection tree: (1) the number of child nodes enumerated from each retained parent node, (2) the number of nodes to retain before extending child nodes, and (3) whether or not to perform parallel smart candidate adding. Additionally, it is essential to specify the preprocessing algorithm in order to accurately describe a breadth-first detector.

We specify the three parameters for a breadth-first tree search using $1 \times N_t$ vectors $\mathbf{m} = [m_1\ m_2\ \ldots\ m_{N_t}]$, $\mathbf{b} = [b_1\ b_2\ \ldots\ b_{N_t}]$ and $\mathbf{s} = [s_1\ s_2\ \ldots\ s_{N_t}]$, respectively, where b_i

Algorithm: SOQR
Input: $\mathbf{H}, b_1$
Output: $\mathbf{Q}, \mathbf{R}, \mathbf{P}$

Find k using (3.1): a function of $\mathbf{H}$ and b_1
$\mathbf{Q} = \mathbf{H}$, $\mathbf{R} = \mathbf{0}_{N_t}$, $\mathbf{P} = \mathbf{I}_{N_t}$
$\mathbf{d} = \texttt{diag}(\mathbf{Q}^*\mathbf{Q})$; $k_2 = k$
for $i = N_T : -1 : 1$ **do**
    if *i==1* **then**
        $k = 1$
    else
        $k = \underset{k \neq k_2}{\arg\min}\ \mathbf{d}$
    end
    if $i == k_2$ **then**
        $k_2 = k$
    end
    Swap columns i and k in $\mathbf{Q}$, $\mathbf{R}$, $\mathbf{P}$
    Swap elements i and k in $\mathbf{d}$
    $R_{i,i} = \sqrt{d_i}$
    $\mathbf{q}_i = \mathbf{q}_i / R_{i,i}$
    for $j = i - 1 : -1 : 1$ **do**
        $R_{i,j} = \mathbf{q}_i^* \mathbf{q}_j$
        $\mathbf{q}_j = \mathbf{q}_j - R_{i,j}\mathbf{q}_i$
        $d_j = d_j - |R_{i,j}|^2$
    end
    $d_i = \infty$
end

Figure 3.3: Smart-Ordered QR (SOQR) Decomposition.

represents the number of children extended from each parent node at the ith tree layer of the detection tree, where m_i represents the number of nodes retained at the ith layer and the boolean vector $\mathbf{s} = [s_1\ s_2\ \ldots\ s_{N_t}]$ determines whether or not to perform parallel smart candidate adding at the ith layer. The union of the **m**, **b** and **s** vectors, along with a specification for the preprocessing algorithm, leads to a framework that allows for the characterization of a large class of fixed complexity, single-pass tree search, breadth-first

MIMO detectors. While this framework is simple to describe, it enables a myriad of possibilities and brings to light many new design considerations. An appropriate configuration is crucial to achieving a desirable performance-complexity tradeoff.

A concise description of the generalized framework is provided in Fig. 3.4. In fact the framework is very close to the pseudocode description for the SOCA algorithm provided in Fig. 3.1, with the modification that the ordered QR decomposition is algorithmic specific and the vector $\mathbf{s}$ is introduced. Our generalized framework takes as input the received signal $\mathbf{r}$, the MIMO channel $\mathbf{H}$, the alphabet $\mathcal{A}$, and three vectors $\mathbf{b} = [b_1\ b_2\ \ldots\ b_{N_t}]$, $\mathbf{m} = [m_1\ m_2\ \ldots\ m_{N_t}]$, and $\mathbf{s} = [s_1\ s_2\ \ldots\ s_{N_t}]$. The set $\mathcal{F}$ is used to denote the surviving nodes at the current layer in the tree. More detail concerning the detection ordering is provided in section 3.3.0.3.

Algorithm: `Generalized Breadth-First Soft-Output Detection`
Input: $\mathbf{r}, \mathbf{H}, \mathbf{b}, \mathbf{m}, \mathbf{s}$
Output: $\mathcal{L}$

```
1  [Q, R, P] = OrderedQR(H)
2  y = Q*r
3  F = root node
4  for i = 1 : N_t do
5  |   F = ∪_{node∈F} {b_i best children of node}
6  |   if s_i = 1 then
7  |   |   for j = (i − 1)ω + 1 : iω do
8  |   |   |   Flip bit j of ĉ^PMAP and add the corresponding node to F
9  |   |   end
10 |   end
11 |   F = m_i best of F
12 end
13 L = PF
```

Figure 3.4: Algorithm for Generalized Breadth-First Soft-Output Detection.

3.4.1 Placement

We will now place existing fixed-complexity breadth-first detectors into the framework just presented and discuss the design considerations that accompany each detection scheme, as well as relationships amongst them [68]. While the list of detection algorithms in this subsection does not claim to be complete, it does provide insight into many of the most common and effective fixed complexity breadth-first schemes.

Decision-Feedback: The simplest scheme to be captured by the presented framework is the decision-feedback equalizer (DFE), or successive interference cancelation (SIC) detector. After removing ("canceling") the signal contribution of previous layers, this scheme will recursively determine the single best candidate at the currently considered layer, i.e. single enumeration, and proceed with this decision to the next layer. Obviously, the tree width is minimized by this scheme. Good DFE performance thus heavily depends on making the correct decision in the initial detection layer (having the lowest diversity order). BLAST (or SQRD) should therefore be used to obtain the optimal (or nearly optimal) DF performance. DF detection is captured through the parameterization $\mathbf{b} = [1 \ \ldots \ 1]$, $\mathbf{m} = [\infty \ \ldots \ \infty]$, and $\mathbf{s} = [0 \ \ldots \ 0]$ and the specification of the preprocessing algorithm. Consequently $\mu = N_t$ and $\ell = 1$.

Parallel Detector [58]: Rather than using single enumeration at the first detected layer, i.e. $b_1 = 1$, the parallel detector uses full enumeration at this layer, i.e., $b_1 = q$ so that all candidates are enumerated. All subsequent layers are detected from the q parent nodes at the first layer using decision feedback detection. The PD also adopts a special ordering, where the weakest received signal component is detected first. Subsequent layers use the BLAST ordering. The intuitive justification for such an approach is that in each layer where all nodes are enumerated, no decision errors can occur. It is therefore more desirable to enumerate all candidates for the layers with the largest noise enhancement, to

minimize performance loss, rather than waste complexity on layers which do not enhance the noise significantly. The PD uses the parameterization $\mathbf{b} = [q\ 1\ \ldots\ 1]$, $\mathbf{m} = [\infty\ \ldots\ \infty]$, $\mathbf{s} = [0\ \ldots\ 0]$ and PD preprocessing. Thus, $\mu = qN_t$ and $\ell = q$.

B-Chase Detection [96, 97, 95, 98]: The B-Chase(ℓ) detector is a hard-output detector that generates a list of ℓ tentative decisions for the first detected symbol, and implements a bank of ℓ ordered decision-feedback detectors in parallel, one for each element of the list. In the case of hard-output detection, the final decision vector is the DF equalized output that minimizes the mean-squared error (MSE). As described in section 3.3.0.3, the performance-complexity trade-off for Chase detection is easily adapted by adjusting ℓ, as Chase detection reduces to ordered DF when $\ell = 1$ and the PD when $\ell = q$. Increasing ℓ improves performance at the cost of a complexity growing linearly in ℓ. The B-Chase detector uses the parameterization $\mathbf{b} = [\ell\ 1\ \ldots\ 1]$, $\mathbf{m} = [\infty\ \ldots\ \infty]$, $\mathbf{s} = [0\ \ldots\ 0]$ and B-Chase preprocessing. Hence, it computes $\mu = \ell N_t$ branch metrics and the list size is ℓ.

Fixed-Complexity Sphere Decoder [9, 51]: The FSD extends the PD to handle cases when the number of candidates enumerated at a detection layer is neither 1 or q. Specifically, when all nodes are enumerated at a given layer in the detection tree, the FSD adopts the ordering criterion of the PD, otherwise it uses the BLAST ordering. Similar to the PD and the B-Chase detector, paths once generated are never pruned. The FSD is capable of many parameterizations, where $\mathbf{s}$ is always the zero vector. The most effective parameterizations, however, are those of the PD and, for large dimensions such as 8×8 [7], the parameterization $\mathbf{b} = [q\ q\ 1\ \ldots\ 1]$ and $\mathbf{m} = [\infty\ \ldots\ \infty]$.

List Fixed-Complexity Sphere Decoder[8]: The list fixed-complexity sphere decoder (LFSD) is a soft-output extension of the FSD. It builds on the FSD approach by typically

computing more branch metrics than the FSD, in order for $\mathcal{L}$ to include more counter-hypotheses to the hard-output FSD decision vector. In [8] this was typically done using balanced powers of 2 for $b_2, \ldots b_{N_t}$. In the event that this was not possible due to list length constraints, these powers of 2 are weighted to earlier layers in the detection tree.

Parallel Smart Candidate Adding [113]: The parallel smart candidate adding algorithm is very similar to the SOCA algorithm except that $s_1 = 1$ and a SQRD preprocessing is used. Consequently, the SOCA represents an improvement over the PSCA algorithm [113]. The PSCA algorithm is captured through the parameterization $\mathbf{m} = [\infty \ \ldots \ \infty]$, $\mathbf{s} = [1 \ \ldots \ 1]$, and an appropriate selection of the preprocessing algorithm and the vector $\mathbf{b}$. Note that in the case $b_i > 3$ ($b_i > 2$ for the real-valued model) a slight variance in complexity is possible, since the b_i closest points will then generate a varying number of counter-hypotheses to the partial MAP estimate.

Smart Ordered Candidate Adding [67]: The proposed SOCA algorithm has been discussed in detail in this chapter. As described, two key parameter choices allow for the near max-log optimal performance of the SOCA algorithm. First, the smart-ordered QR decomposition provided in Fig. 3.3 is employed. Second, the SOCA algorithm is realized by keeping the elements of $\mathbf{b}$ small (1 if possible), with an exception for the first detection layer (i.e. $b_1 > 1$) due to the fact that the diversity order of the first symbol to be detected is $N_r - N_t + 1$ and a mistake here leads to error propagation. Due to the selection of $b_1 > 1$ the PSCA parameter is set to $\mathbf{s} = [0 \ 1 \ \ldots \ 1]$.

3.4.2 Computational Complexity

We now describe the core processing computational complexity of breadth-first detectors, as classified in section 3.4, using the total number of branch metric computations μ. Specifically, μ is a function of the number of nodes retained ζ_i for a given layer in the detection tree [68]:

$$\zeta_i = \min\left(\zeta_{i-1} b_i + s_i \kappa_i, m_i\right) \tag{3.2}$$

where i is the detection layer, ζ_0 is initialized to 1, and κ_i denotes the number of branch metrics calculated as part of the smart candidate adding at layer i. The term κ_i is given by:

$$\kappa_i = \max\left(\omega - 2(\sqrt{b_i} - 1), 0\right). \tag{3.3}$$

For the complex-valued system model it is assumed that b_i is the square of an integer [68]. The total number of branch metric computations for schemes in our framework becomes:

$$\mu = \sum_{i=1}^{N_t} \left(\zeta_{i-1} b_i + s_i \kappa_i\right). \tag{3.4}$$

The final list size is $\ell = |\mathcal{L}| = \zeta_{N_T}$. Consequently, all that is required to determine the core computational complexity for breadth-first detectors, as classified in section 3.4, in terms of the number of branch metrics are the vectors $\mathbf{b}$, $\mathbf{m}$ and $\mathbf{s}$. Large entries for the vector $\mathbf{b}$, particularly at the early layers in the detection tree, and $\mathbf{m} = \infty$ lead to a large detection tree, whereas small entries for the vector $\mathbf{b}$ and $\mathbf{m} = \infty$ yield a smaller detection tree. The desire to keep the entries in $\mathbf{b}$ small (as close to 1 as possible) is in many ways the motivation behind the SOCA algorithm.

The SOCA algorithm has the property that when $\mathbf{m} = \infty$ and $\mathbf{b} = [b_1\ 1\ \ldots\ 1]$, as is the case for the 4×4 results presented in section 3.5.2, only

$$\mu = b_1 + \sum_{i=2}^{N_t} b_1 + \omega(i-1) = N_t\left(b_1 + \frac{\omega(N_t - 1)}{2}\right) \tag{3.5}$$

branch metrics are computed. The intuition behind (3.5) is that first there are b_1 nodes enumerated from the root of the detection tree and each of these traverses each of the N_t

layers of the detection tree resulting in $N_t b_1$ leaf nodes. Additionally, at each layer of the detection tree, excluding the first (i.e. $N_t - 1$ layers) ω additional nodes are added for which decision feedback detection occurs until ω leaf nodes are obtained. Multiplying out the right hand side of (3.5) demonstrates that the number of branch metric computations for the SOCA algorithm has order $\Theta(N_t^2)$. As a sanity check, for the formula in (3.5) and $b_1 = 3$ and $N_t = 2$ the SOCA algorithm computes exactly eight branch metrics. This is the same result as in the example in Fig. 3.2.

The number of candidate vectors in the list $\mathcal{L}$ for the SOCA algorithm when $\mathbf{b} = [b_1\ 1\ \ldots\ 1]$ is given by:

$$\ell = \min\left(m_{N_t}, b_1 + \omega\left(N_t - 1\right)\right). \tag{3.6}$$

When $m_{N_t} = \infty$ then $\ell = b_1 + \omega\left(N_t - 1\right)$. This is simply a mathematical statement saying that, for each detection layer after the first, ω additional nodes are enumerated in addition to the decision feedback detection occurring from each node retained from the previous layer in the tree, ultimately resulting in an additional $\omega(N_t - 1)$ leaf nodes. Consequently, for the SOCA algorithm, the list size grows linearly with the number of input streams.

3.5 Results and Analysis

This section is used to provide performance versus computational complexity results for the proposed SOCA and PSCA algorithms, as well as many of the soft-output detection algorithms detailed so far in this dissertation. We begin by describing the simulation setup, followed by the specific parameterizations used for both fast and slow fading channel conditions. We present results and analysis for both fast and slow fading scenarios in order to consider situations where the ability to extract either (a) the time diversity or (b) the spatial diversity from the channel is essential.

3.5.1 Simulation Setup

For both fast and slow fading scenarios the detector is run only once, i.e. we do not employ iterative detection-decoding. For all scenarios, detection is performed using the complex-valued system model, a random interleaver is used, and we employ unbiased MMSE detection, as detailed in section 2.6. Additionally, LLRs were clipped at a magnitude of ±6 for all investigated algorithms. LLR clipping based on channel state information improves performance and is the focus of chapter 4.

3.5.1.1 Fast Fading

For the fast-fading scenario we use temporally i.i.d. fading, i.e., each transmitted vector symbol sees a new channel realization. For coded results, the information block size (including tail bits) is 9216 bits and a setup equivalent to the one in [47] is employed: a rate 1/2 parallel concatenated convolutional code (PCCC) based on memory 2 constituent convolutional codes with generator polynomials $(7_R, 5)_{\text{octal}}$ using 8 internal iterations of logMAP decoding, where R denotes which generator is in the denominator. Fast-fading performance is measured in terms of the averaged E_b/N_0 in dB to achieve an uncoded BER (uBER) of 10^{-2} or a coded BER (cBER) 10^{-5} to match [47].

3.5.1.2 Slow Fading

Here we assume the channel does not change during the duration of the entire transmitted codeword and that the channel matrix entries are drawn anew with the transmission of each new codeword. A convolutional code with code polynomial [133 171] and constraint length 7, punctured to code rate $3/4$ is employed and the information block size (including tail bits) is 3456. The convolutional decoder employed is MaxLog(MAP). Performance is measured in terms of the E_b/N_0 in dB required to achieve a frame error-rate (FER) of 10^{-2}. We used the FER to measure slow-fading performance because for this scenario, where we employ a weak code and the channel offers no time diversity, BER results can often be

misleading. The target FER of 10^{-2} was selected because it is common to design systems for this error rate [87].

3.5.2 Results

A summary of the algorithmic placements from the previous subsections is provided in Table 3.5.2 for a 4×4 fast fading MIMO channel [68]. In addition to specifying the parameterization of the aforementioned algorithms, it provides the number of branch metric calculations for a 4×4 MIMO system using 16-QAM and 64-QAM transmission alphabets, as well as the SNR required to achieve the target uncoded and coded BERs. Typical parameterizations for the M, LFSD, PSCA, and SOCA algorithms are provided. Additionally, the preferred channel decomposition for each algorithm is provided.

The algorithms listed in the first three results columns of Table 3.5.2, i.e. the BLAST-ordered decision feedback detector [91, 34, 60], the parallel detector [58], and the B-Chase detector [98] were all designed as hard-output detectors. As the B-Chase detector is a generalization of both the BLAST-ordered decision feedback detector and the parallel detector, we recommend it to those seeking a hard-output MIMO detector that has a good performance-complexity tradeoff. Results in [68] support this recommendation.

Out of the soft-output detectors in Table 3.5.2, i.e. the LFSD, the PSCA and the SOCA algorithms, the SOCA algorithm parameterized with $\mathbf{b} = [8\ 1\ 1\ 1]$ and $\mathbf{b} = [16\ 1\ 1\ 1]$ has the most desirable performance-complexity tradeoff. These two SOCA realizations are within 0.3 and 0.2 dB of the LFSD with $\mathbf{b} = [q\ 2\ 2\ 2]$ for 64-QAM transmission, and require only 6% and 10% of the computational complexity of the realized LFSD, respectively.

We now present a series of error-rate performance versus computational complexity results, with an emphasis on the proposed SOCA algorithm. In addition to the fixed computational complexity SOCA algorithm, results are presented for the LSD [47] and the LISS [11]. The LISS employs a bias parameter to perform statistical tree pruning set to 1

	BLAST B-Chase(1) FSD(1)	PD B-Chase(q) FSD(Q)	B-Chase(ℓ)	LFSD	LFSD	PSCA	PSCA	SOCA	SOCA
m	∞	∞	∞	∞	∞	∞	∞	∞	∞
b	[1 1 1 1]	[q 1 1 1]	[ℓ 1 1 1]	[q 2 1 1]	[q 2 2 2]	[1 1 1 1]	[4 4 1 1]	[8 1 1 1]	[16 1 1 1]
s	[0 0 0 0]	[0 0 0 0]	[0 0 0 0]	[0 0 0 0]	[0 0 0 0]	[1 1 1 1]	[1 1 1 1]	[0 1 1 1]	[0 1 1 1]
μ 16-QAM	4	64	4ℓ	112	240	44	96	56	88
SNR[dB] 10^{-2} uBER	16.05	12.85	–	–	–	13.06	12.81	–	–
SNR[dB] 10^{-5} cBER	13.08	9.67	–	9.33	8.84	9.37	8.95	9.10	9.03
μ 64-QAM	4	256	4ℓ	448	960	64	134	62	94
SNR[dB] 10^{-2} uBER	21.66	17.88	–	–	–	18.44	18.06	–	–
SNR[dB] 10^{-5} cBER	18.01	13.84	–	13.37	12.96	13.72	13.22	13.64	13.31

Table 3.1: Comparison of various fixed-complexity breadth-first MIMO detection schemes for a 4×4 MIMO channel.

at each level of the detection tree [16]. This bias parameter reduces computational complexity at the cost of a small performance penalty relative to the LSD with the same list size. We also compare against the single-tree-search LSD algorithm [87]. Finally, results for a soft-output implementation of the M algorithm [3] are provided, where we form $\mathcal{L}$ from the m_{N_t} best leaf nodes at the final detection layer. Finally, all algorithms employ the best-first Schnorr-Euchner enumeration [76], rather than Fincke-Pohst enumeration [32].

Fig. 3.5 depicts performance versus computational complexity results for 16-QAM transmission in fast Rayleigh fading. The average computational complexities for the LSD, LISS, and STS-LSD are represented using dashed lines and the 99.9th percentile computational complexities are represented using solid lines. The LSD is represented by square markers, the STS-LSD by diamond markers and the LISS by circular markers. For the LSD the list size ℓ is provided for each marker. The same list sizes are represented for the LISS, although the performance results differ due to the statistical tree pruning performed by the LISS [16]. For the STS-LSD the numbers next to the markers represent the value of the clipping/pruning parameter $L_{\max}$ as described in [87], instead of the list length, because this parameter determines the computational complexity for the STS-LSD. In addition to the variable complexity algorithms, the solid curve denoted with pentagram markers represents the proposed fixed complexity SOCA algorithm. The numbers corresponding to each SOCA marker denote the number of nodes enumerated at the first detection layer b_1. Finally, we note that for all 4×4 SOCA results we use the parameterization $\mathbf{b} = [b_1\ 1\ 1\ 1]$, where b_1 is the number of child nodes enumerated from the root of the tree, and $\mathbf{m} = \infty$ so that no tree pruning occurs. A consequence of omitting tree pruning is that we do not need a sort-and-select stage to determine the m_i nodes to retain at the i^{th} layer of the tree. Instead, all that is needed is the selection of the lowest cost node at any layer in the tree so that parallel smart candidate adding can be applied to this node.

Fig. 3.5 shows that for the fast-fading case, the averaged computational complexity for the STS-LSD (i.e. $\overline{\text{STS-LSD}}$) algorithm achieves a desirable performance-complexity

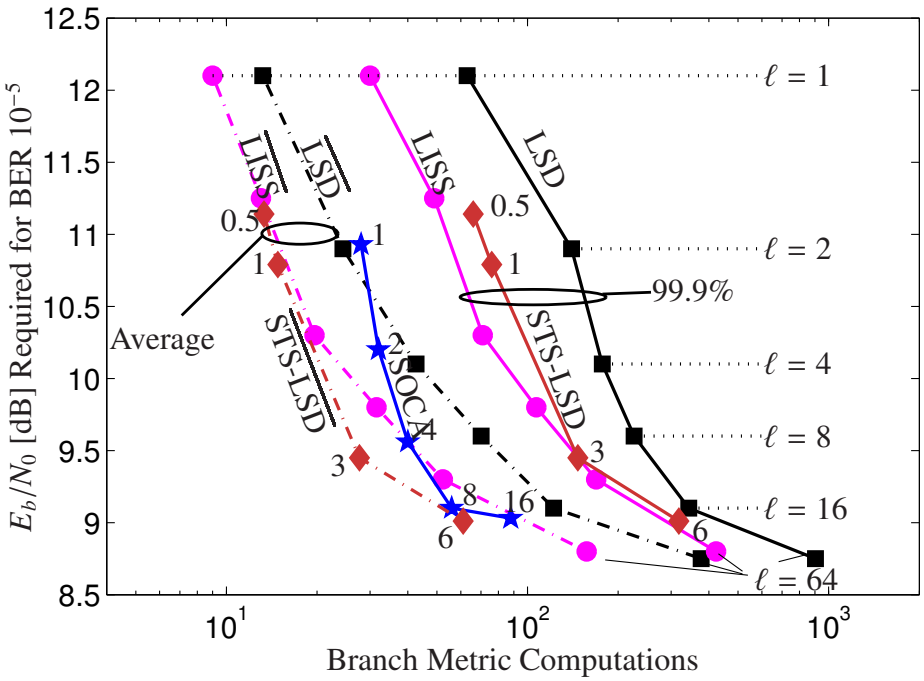

Figure 3.5: Performance vs. complexity for soft-output 4×4 MIMO detection schemes using 16-QAM transmission in fast Rayleigh fading. The numbers corresponding to the SOCA curve represent the value for b_1 and the numbers corresponding to STS-LSD curves represent $L_{\max}$.

tradeoff. Often the worst-case (or bounded worst-case) computational complexity is more important in terms of system design. The performance-complexity curve for the $\overline{\text{STS-LSD}}$ therefore serves as a somewhat idealized reference to which other detection algorithms should aspire. Here the fixed-complexity SOCA algorithm with $b_1 = 8$ is an attractive option because, while it performance-complexity profile is worse than the $\overline{\text{STS-LSD}}$, it significantly outperforms the 99.9th percentile STS-LSD. Fig. 3.5 also shows that for $b_1 > 4$ the SOCA achieves a better performance-complexity tradeoff than the $\overline{\text{LISS}}$ or $\overline{\text{LSD}}$ algorithms.

In contrast to all variable complexity algorithms, the SOCA algorithm achieves its performance-complexity tradeoff with fixed computational complexity, an advantage from an architectural standpoint because it leads to a regular design structure. Additionally, the breadth-first layer-by-layer nature of the SOCA algorithm means that it is possible to construct a parallel architectural implementation with low latency [75].

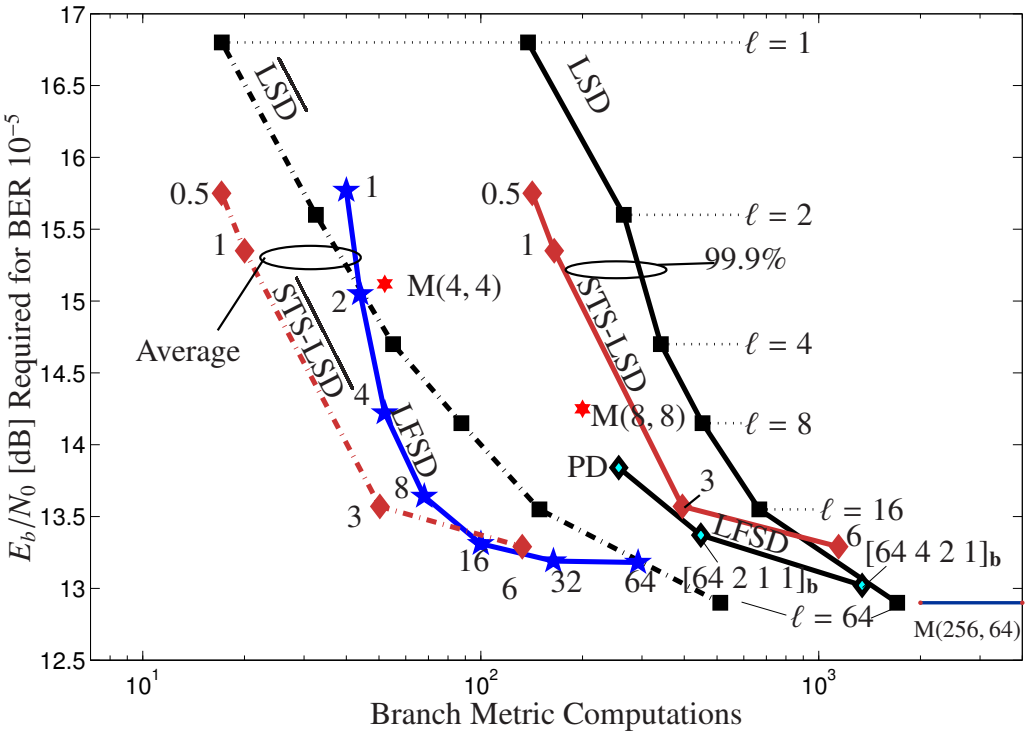

Figure 3.6: Performance vs. complexity for soft-output 4×4 MIMO detection schemes using 64-QAM transmission in fast Rayleigh fading. Results for the LFSD [8] are provided for $\mathbf{b} = [64\ 1\ 1\ 1]$, $\mathbf{b} = [64\ 2\ 1\ 1]$ and $\mathbf{b} = [64\ 4\ 2\ 1]$.

Fig. 3.6 provides the same performance-complexity plot as Fig. 3.5, but for 64-QAM transmission. Once more the SOCA performance-complexity curve falls between that of the average and 99.9th percentile computational complexity for the STS-LSD, with the SOCA having fixed computational complexity. Fig. 3.6 also compares against the soft-output M algorithm [3] and the LFSD [8]. The reference performance provided was found using the K-best algorithm and $m = 256$. Such a realization would compute over 36000 branch metrics and so the computational complexity is not shown. The LFSD is denoted by lightly shaded circular markers with dark edges. In its minimum configuration the LFSD reduces to a soft-output parallel detector, i.e. $\mathbf{b} = [64\ 1\ 1\ 1]$ with all leaf nodes in the tree used to form $\mathcal{L}$. LFSD results are also provided for $\mathbf{b} = [64\ 2\ 1\ 1]$ and $\mathbf{b} = [64\ 4\ 2\ 1]$, where the subscript $\mathbf{b}$ is used to denote that the vector to which it is attached is $\mathbf{b}$. One reason the SOCA algorithm outperforms the LFSD in terms of the performance-complexity tradeoff is because of the way it adds counterhypotheses. Specifically, rather than increasing the elements of $\mathbf{b}$ like the LFSD (i.e. $b_i > 1$), the SOCA simply bit flips around the estimate that is currently best, thereby growing the tree by addition of nodes rather than a multiplicative factor of nodes. Additionally, because of its use of the SOQR, the SOCA does not need to extend all $q = 16$ child nodes at the first layer of the tree to achieve good performance.

Fig. 3.7 provides a performance-complexity plot for 16-QAM transmission in slow fading, where the curves, algorithms and markers are the same as outlined previously, with LFSD results provided for $[16\ 1\ 1\ 1]_{\mathbf{b}}$, $[16\ 2\ 2\ 1]_{\mathbf{b}}$, $[16\ 2\ 2\ 1]_{\mathbf{b}}$ and $[16\ 4\ 2\ 2]_{\mathbf{b}}$. From Fig. 3.7 it can be observed that an increase in SNR is required for the slow-fading scenario to achieve comparable error-rate performance to the fast-fading scenario. In slow fading the SOCA algorithm remains an attractive option, even though its performance-complexity profile is never superior to the average computational complexities of the LSD, LISS, or STS-LSD. However, the fixed computational complexity of the SOCA algorithm is again significantly lower than the worst-case (or bounded worst-case) computational complexity of the LSD and LISS. Finally, the 99.9th percentile computational complexity for the

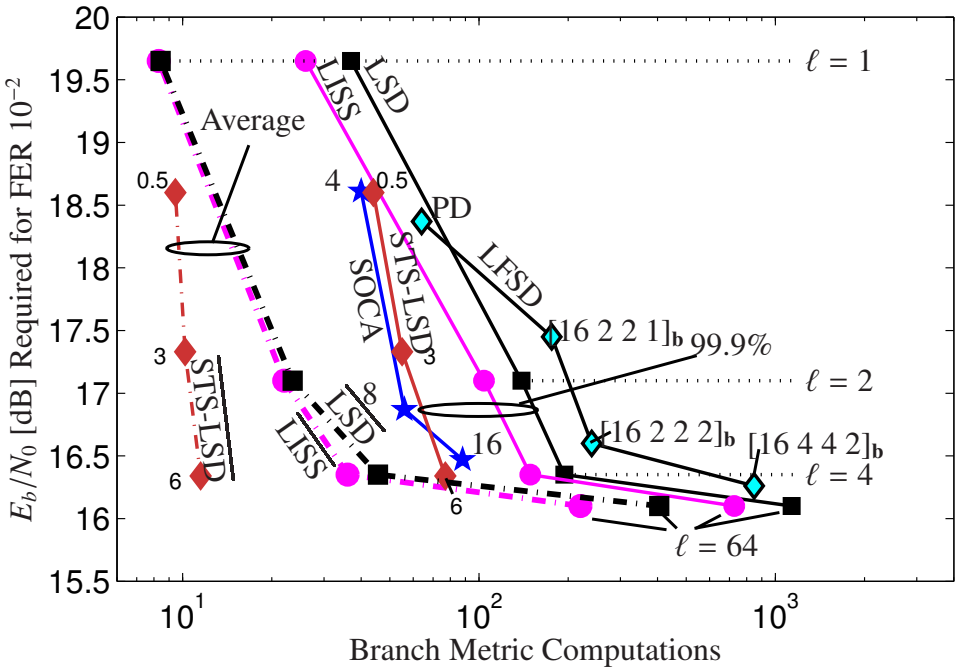

Figure 3.7: Performance vs. complexity for soft-output 4×4 MIMO detection schemes using 16-QAM transmission in slow Rayleigh fading.

STS-LSD has almost the same computational complexity as the SOCA algorithm for the algorithmic realizations presented. Here, the STS-LSD employing upper bounded computational complexity is an attractive alternative to the SOCA.

Fig. 3.8 provides results for a 4×4 channel employing 64-QAM transmission in slow fading and is used to demonstrates the importance of the SOQR on the overall error-rate performance. The solid curve with left facing triangular markers, denoted SQRD-CA, represents the SOCA algorithm except that instead of using a SOQR decomposition the algorithm employs the commonly used sorted-QR decomposition [102]. Ignoring the forced detection ordering in the first layer, the SQRD-CA and SOCA have identical computational complexities, yet the SOCA algorithm outperforms the SQRD-CA algorithm by 1.2 dB when $b_1 = 16$.

We now look at a larger 8×8 communication channel. Fig. 3.9 provides performance versus computational complexity results for a fast-fading 8×8 MIMO channel. The performance of the K-best algorithm with $m = 512$ is the reference performance for this system

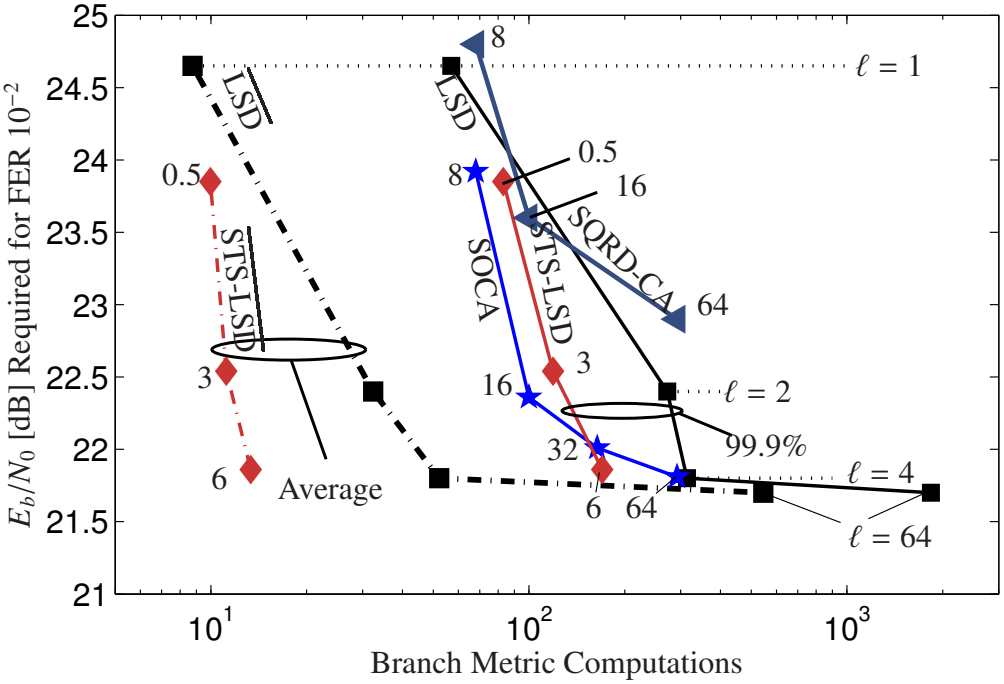

Figure 3.8: Performance vs. complexity for soft-output 4×4 MIMO detection schemes using 64-QAM transmission in slow Rayleigh fading.

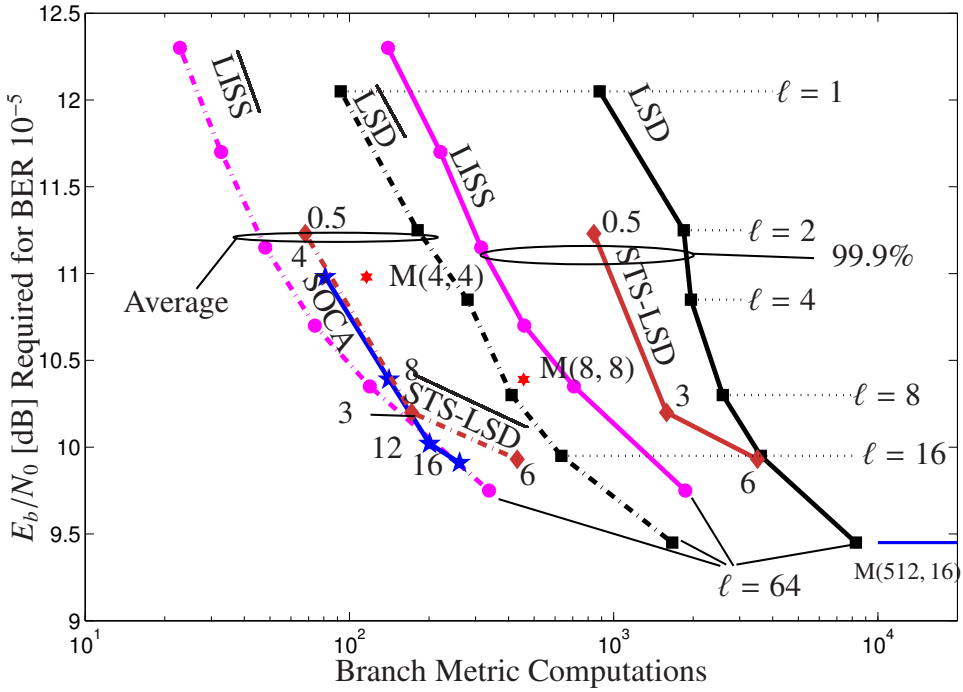

Figure 3.9: Performance vs. complexity for soft-output 8×8 MIMO detection schemes using 16-QAM transmission in fast Rayleigh fading.

configuration. We also depict the M algorithm with $m = b = 4$ and $m = b = 8$. In order to achieve a desirable performance versus computational complexity tradeoff for this larger system size, the SOCA algorithm requires a change to $\mathbf{b}$ such that $\mathbf{b} = [b_1\ 2\ \ldots\ 2]$, where b_1 is the number of child nodes enumerated from the root of the tree. This is because the performance drops off significantly when $\mathbf{b}$ is maintained at $\mathbf{b} = [b_1\ 1\ \ldots\ 1]$. A second important change is the incorporation of tree pruning. For the results shown in Fig. 3.9, at each level of the tree the survivor nodes were pruned to $m_i = b_1$, i.e. the $\mathbf{m}$ vector for the SOCA algorithm was set to $\mathbf{m} = [b_1\ b_1\ \ldots\ b_1]$. This means that in Fig. 3.9 the corresponding value next to each marker for the SOCA represents the algorithmic realization when $b_1 = m_1 = \ell$. Without tree pruning the performance of the SOCA algorithm is slightly improved relative to the SOCA without tree pruning. However, these results are not shown because the computational complexity would increase prohibitively when tree pruning is omitted. This increase is due to the large system size which, without tree pruning, allows for extra layers of tree growth. Finally, we note that the SOCA algorithm has roughly the same performance-complexity curve as the average complexity of the STS-LSD.

Fig. 3.10 provides the same 8×8 16-QAM results as Fig. 3.9 but for the slow-fading scenario. The SOCA algorithm with $b_1 = m_1 = 16$ has roughly the same performance as the LISS with $\ell = 4$ but its computational complexity is 45% of the 99.9th percentile computational complexity. Additionally, the SOCA algorithm with $b_1 = 12$ has roughly the same performance as the M algorithm with parameterization $m = b = 8$, but with 57% of the complexity. This savings reduction come from the fact that, for layers 2 down to N_t of the tree, we have a multiplier of $b_i = 2$ for the SOCA and a significantly larger $b_i = 8$ for the M algorithm. Finally, we observe that the SOCA algorithm has a fixed performance-complexity curve that sits between the average and 99.9th percentile computational complexity of the STS-LSD. Thus, even for the most challenging scenario presented (i.e. 8×8 16-QAM in slow fading) the SOCA algorithm remains a good choice for soft-output MIMO detection.

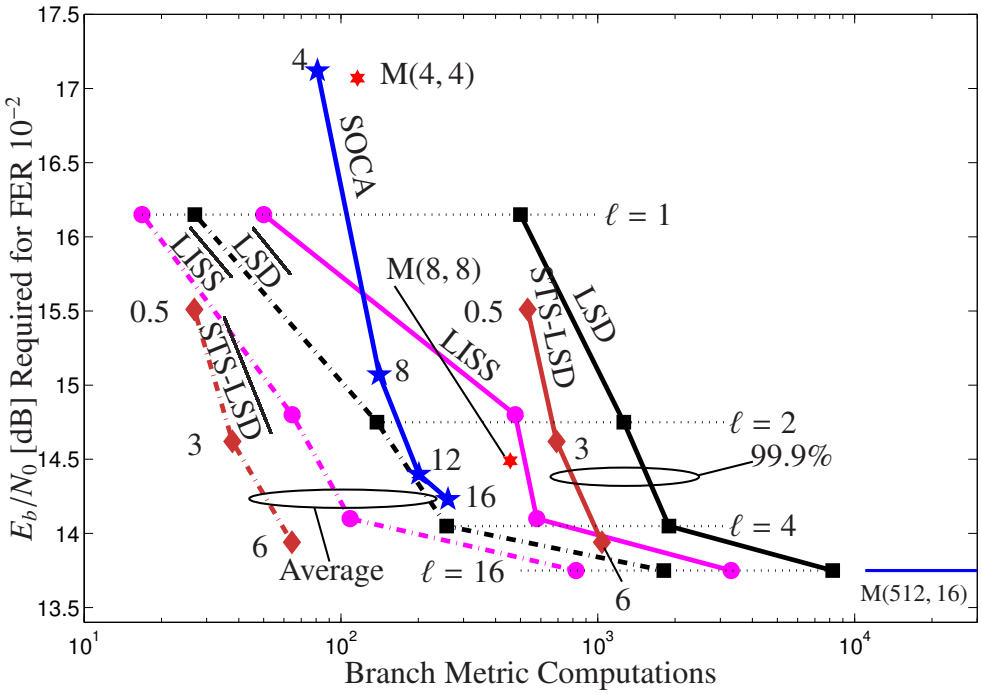

Figure 3.10: Performance vs. complexity for soft-output 8×8 MIMO detection schemes using 16-QAM transmission in slow Rayleigh fading.

3.6 Soft Fixed-Complexity Sphere Decoder

A recently proposed important soft-output MIMO detection algorithm related to several detectors in this dissertation is the soft fixed-complexity sphere decoder (SFSD). The SFSD, while not a single-pass approach like the SOCA algorithm and other algorithms classified in section 3.4, is a soft-output extension of the FSD. Unlike the LFSD which is also a soft-output extension of the FSD, the SFSD is more similar to previously reported SCA approaches [62, 105, 94, 110, 113], with the process of smart candidate adding being referred to as "bit-negating" and "path augmentation" [6]. Specifically, the SFSD can be thought of as the combination of the hard-output FSD approach, used to generate the set $\mathcal{L}_{\mathcal{FSD}}$, with an iterative SCA type approach used to generate the set $\mathcal{L}_{\mathcal{SCA}}$, where $\mathcal{L} = \mathcal{L}_{\mathcal{FSD}} \cup \mathcal{L}_{\mathcal{SCA}}$. Unlike other SCA approaches, the SFSD typically employs FSD ordering and enumerates all nodes in the first detection layer(s). When only one iteration is performed (SCA augmentation of a single path), the SFSD approach is similar to an algorithmic realization in [110]. With multiple iterations (SCA extended paths), SFSD performance can be

improved, but the performance improvements are relatively small and detection complexity is substantially increased. Additionally, there exist many other fixed (or quasi-fixed) complexity breadth-first algorithms that are related to algorithms in this chapter that we did not treat due to their hard-output nature. Examples of these other algorithms include [65, 52, 82, 53].

3.7 Summary

In this chapter we presented an algorithm capable of near max-log optimal error-rate performance that, in contrast to prior soft-output breadth-first algorithms, possessed low and fixed computational complexity. This is an important result because it invalidates what is often the most relevant criticism against breadth-first approaches, i.e. that they suffer severe performance loss when the available computational complexity is small. Additionally, the fact that the proposed SOCA algorithm is breadth-first is an advantage because its layer-by-layer structure means that it is possible to construct parallel architectural implementations with low latency. Such architectural implementations for the proposed SOCA algorithm, however, do not yet exist and are therefore an interesting area for future research.

Results in this chapter were provided for a spatially multiplexed BICM system in slow and fast fading environments. The slow fading environment, where no time diversity was available, proved to be far more challenging than the fast fading environment in terms of the required computational complexity to achieve near max-log optimal performance. One transmission scheme to combat slow fading environments not discussed in this chapter is the use of diversity-multiplexing schemes such as algebraic space-time codes. In the next chapter we will present an algorithm for soft-output MIMO detection of an important algebraic space-time code known as the golden code.

CHAPTER 4

SOFT-OUTPUT DETECTION OF THE GOLDEN CODE

4.1 Introduction

Space-time codes are an effective means of providing a diversity gain, and potentially a coding gain, to multiantenna communications systems [2, 89]. A class of space-time codes that combines a diversity gain with a multiplexing gain are the algebraic space-time codes. Algebraic space-time codes are an area of significant current research, where much of this research focuses on the design of codes achieving the optimal diversity-multiplexing trade-off described in [107].

An important algebraic space-time code, proposed independently in [13] and [27], is the golden code for two transmit and two receive antennas. The golden code offers many benefits. First, the golden code is a full-rate code, meaning that the ratio of the number of distinct symbols transmitted to the total transmission time is one. Second, the golden code is a full-diversity code, implying that the difference between all real codeword matrices is full-rank (four in the case of the golden code). Third, the golden code has maximal coding gain. Specifically, the golden code performs better than all previously reported full-rate codes with two transmit antennas in terms of the SNR required to achieve a target error probability [85]. For these and other reasons the golden code has been incorporated into the 802.16e WiMAX standard [49].

Soft-output detection of the golden code is an important but computationally difficult task. In this chapter we propose a low- and fixed- complexity soft-output detection algorithm for the golden code. Prior work in this area includes a soft-output detection algorithm [81] that has a fixed computational complexity of $\Theta(q^2)$, where q is the alphabet size, which can be prohibitive for large alphabets.

Our proposed algorithm uses linear equalization to simplify the task of finding a list of candidate values for one pair of information symbols, and then – for each pair on the list – it uses decision-feedback equalization to find candidate values for the remaining pair of information symbols. A simple ordering algorithm, as proposed in [66], is used that exploits the golden code's structure to ensure that the overall algorithm performs well. Like [81] our detector has computational complexity $O(q^2)$ when the list length $\ell = q^2$. However, for $\ell << q^2$, our algorithm achieves comparable performance to [81] at much lower complexity.

4.2 Golden Code System Model

Due to the specific structure of the golden code, which we will describe momentarily, we constrain our system model more than in previous chapters. Specifically, we consider the transmitter shown in Fig. 4.1-a [47]. As before, the input is a vector $\mathbf{u}$ of i.i.d. uniform information bits that is encoded and interleaved. Unlike previous chapters, however, the coded bit stream is partitioned into blocks $\mathbf{c}$ of exactly 4ω bits, where $\omega = \log_2 q$ is the number of bits per symbol. Each block is then mapped and encoded onto a vector of four complex information symbols $\mathbf{a} = [a_1, a_2, a_3, a_4]^{\mathrm{T}}$ whose components are taken from a QAM alphabet $\mathcal{A}$ of size $q = |\mathcal{A}| = 2^{\omega}$ and energy $E/2$. The golden code encodes and transmits these four information symbols over two symbol periods from two antennas, so that the rate of the space-time code is two symbols per signaling interval. The transmitted codeword can be expressed as a 2×2 matrix:

$$\mathbf{G} = \begin{bmatrix} g_1[1] & g_2[1] \\ g_1[2] & g_2[2] \end{bmatrix}, \tag{4.1}$$

where $g_i[k]$ denotes the symbol transmitted from antenna $i \in \{1, 2\}$ at time $k \in \{1, 2\}$.

The received signal $r_l[k]$ at receive antenna l at time k is given by:

$$r_l[k] = \sum_{i=1}^{2} g_i[k] h_{i,l}[k] + n_l[k] \tag{4.2}$$

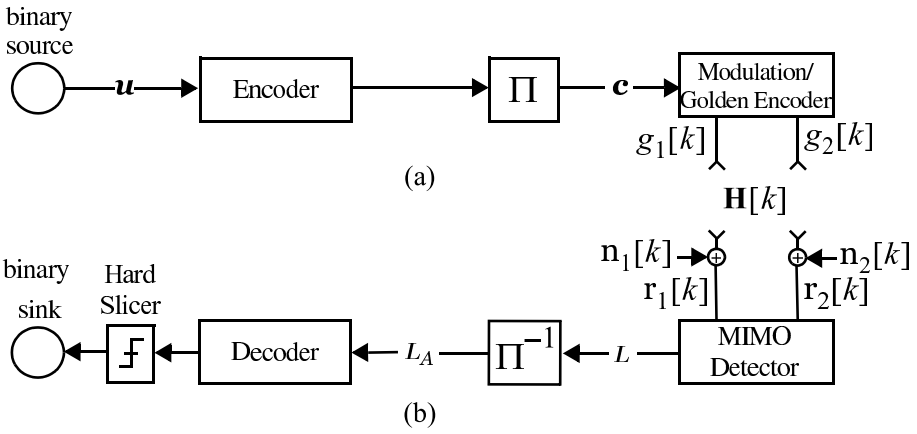

Figure 4.1: System model with (a) MIMO transmitter and (b) MIMO receiver.

where $n_l[k]$ is the complex additive-white Gaussian noise at receive antenna l at time k, $\mathbb{E}|n_l[k]|^2 = N_0$, and $h_{i,l}[k]$ is the channel coefficient between the i-th transmit antenna and l-th receive antenna at time k.

In Fig. 4.1-b we show the MIMO receiver. The receiver has the same soft-output detection objectives as in prior chapters, albeit with the constraint of golden code transmission.

4.3 Effective Channel Matrix

The Dayal-Varanasi golden code [27] encodes one pair of information symbols $\mathbf{v} = [a_1, a_2]^{\mathrm{T}}$ onto the main diagonal of $\mathbf{G}$, and it encodes a second pair of information symbols $\mathbf{w} = [a_3, a_4]^{\mathrm{T}}$ onto the off-diagonal, so that:

$$\mathbf{G} = \begin{bmatrix} \tilde{v}_1 & 0 \\ 0 & \tilde{v}_2 \end{bmatrix} + \phi \begin{bmatrix} 0 & \tilde{w}_1 \\ \tilde{w}_2 & 0 \end{bmatrix} \tag{4.3}$$

where:

$$\tilde{\mathbf{v}} = \mathbf{M}\mathbf{v}, \quad \tilde{\mathbf{w}} = \mathbf{M}\mathbf{w}, \quad \mathbf{M} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix},$$
$$\theta = \frac{1}{2}\tan^{-1}(2), \quad \phi = e^{j\pi/4}. \tag{4.4}$$

Substituting (4.3) and (4.4) into (4.2), the vector of received samples

$\mathbf{r} = [r_1[1],\ r_1[2],\ r_2[1],\ r_2[2]]^{\mathrm{T}}$ at a receiver with two antennas at the two time instances can be written as the output of an effective four-input four-output channel [85]:

$$\mathbf{r} = \mathbf{H}\mathbf{a} + \mathbf{n}, \tag{4.5}$$

where $\mathbf{n} = [n_1[1],\ \ldots\ n_2[2]]^{\mathrm{T}}$ is the noise and $\mathbf{H} = \bar{\mathbf{H}}\Psi$ is the *effective channel matrix:*

$$\mathbf{H} = \underbrace{\begin{bmatrix} h_{11}[1] & 0 & \phi h_{21}[1] & 0 \\ 0 & h_{21}[2] & 0 & \phi h_{11}[2] \\ h_{12}[1] & 0 & \phi h_{22}[1] & 0 \\ 0 & h_{22}[2] & 0 & \phi h_{12}[2] \end{bmatrix}}_{\bar{\mathbf{H}}} \underbrace{\begin{bmatrix} t & s & 0 & 0 \\ -s & t & 0 & 0 \\ 0 & 0 & t & s \\ 0 & 0 & -s & t \end{bmatrix}}_{\Psi}, \tag{4.6}$$

where $s = \sin(\theta)$ and $t = \cos(\theta)$. Using (4.5) as our effective channel model allows us to perform soft-output tree-based list detection as described in chapter 2. That is to say, we perform an ordered QR decomposition on the effective channel matrix $\mathbf{H}$ and use the resultant detection tree to perform soft-output list detection.

4.4 Soft-Output Detection of the Golden Code

Finding a list of low-cost candidates, when employing the golden code, is a challenge. This section presents a low- and fixed- computational complexity algorithm for finding a list of information vectors for soft-output detection of the golden code. The first step is an *ordering* task that determines which two of the four information symbols are to be detected first. The second step is to find a candidate list of possible values for this first pair of information symbols; this task is facilitated by a linear zero-forcing filter. The final task is to find, for each candidate pair from the list, a complementary pair of values for the remaining information symbols using decision-feedback detection. The proposed algorithm can be viewed as an application of the Chase framework [98], except that we are choosing a pair of information symbols to detect first, rather than just one symbol.

4.4.1 Ordering of the Effective Channel Matrix

We first determine the initial pair of information symbols detected, and then the order of detection for the remaining pair of symbols. The first step in the proposed algorithm is to determine which pair of information symbols should be detected first, and to furthermore determine the order in which the remaining pair of symbols is to be detected. Consider the system in (4.5), which can be rewritten as:

$$\mathbf{r} = \bar{\mathbf{H}}\mathbf{x} + \mathbf{n}, \tag{4.7}$$

where $\bar{\mathbf{H}} = \mathbf{HP}$ is a permuted channel, where $\mathbf{P}$ is a permutation matrix, and where $\mathbf{x} = \mathbf{P}^{-1}\mathbf{a}$ is the permuted vector of information symbols. The ordering can be represented by the matrix $\mathbf{P}$, which approximately maximizes the SNR for the first pair of detected symbols $(\hat{x}_1, \hat{x}_2)$.

The pseudocode for an efficient ordering algorithm, first presented in [66] and reprinted with the permission of the collaborating authors, is shown in Fig. 4.2. As described in [66], the SNR for the first detected symbol is maximized by selecting the column with largest norm to be detected first. Additionally, to maximize the SNR of the second symbol detected, we choose the second column index k_2 such that $\bar{\mathbf{h}}_1$ and $\bar{\mathbf{h}}_2$ (or $\mathbf{h}_{k_1}$ and $\mathbf{h}_{k_2}$) are as orthogonal as possible, i.e. to minimize [66]:

$$\delta = |\mathbf{h}_{k_1}^{*}\mathbf{h}_{k_2}|/(||\mathbf{h}_{k_1}||||\mathbf{h}_{k_2}||). \tag{4.8}$$

For additional details on the ordering algorithm and inherent complexity savings due to the symmetry of the golden code the reader is referred to [66].

4.4.2 ZF Equalization, List Enumeration, and DF Detection

Consider the system in (4.7). After a QR decomposition $\bar{\mathbf{H}} = \mathbf{QR}$ and multiplying the channel output by $\mathbf{Q}^{*}$, we obtain the effective channel output $\mathbf{y} = \mathbf{Q}^{*}\mathbf{r}$. Applying a linear filter $\mathbf{R}^{-1}$ and performing a symbol slicing operation yields:

$$\hat{\mathbf{x}} = Q\left(\mathbf{R}^{-1}\mathbf{y}\right) = Q\left(\mathbf{x} + \tilde{\mathbf{n}}\right), \tag{4.9}$$

Input: $\mathbf{H}$ **Output**: $\mathbf{P}$

if $\|\mathbf{h}_1\| > \|\mathbf{h}_2\|$ **then**
  $k_1 = 1$
  $C_2 = [2, 3, 4], C_3 = [4, 4, 2], C_4 = [3, 2, 3]$
else
  $k_1 = 2$
  $C_2 = [1, 3, 4], C_3 = [3, 1, 3], C_4 = [4, 4, 1]$
end
$\delta_{min} = \infty$
for *i from* 1 *to* 3 **do**
  $k_2 = C_2[i], k_3 = C_3[i], k_4 = C_4[i]$
  $\delta = |\mathbf{h}_{k_1}^* \mathbf{h}_{k_2}|/(\|\mathbf{h}_{k_1}\|\|\mathbf{h}_{k_2}\|)$
  if $\delta < \delta_{min}$ **then**
    $\delta_{min} = \delta$
    $\mathbf{P} = [\mathbf{e}_{k_1}, \mathbf{e}_{k_2}, \mathbf{e}_{k_3}, \mathbf{e}_{k_4}]$
  end
end

Figure 4.2: Ordering Algorithm

where $Q(\cdot)$ rounds its input to the nearest element of $\mathcal{A}$ and $\tilde{\mathbf{n}} = \mathbf{R}^{-1}\mathbf{Q}^*\mathbf{n}$. The proposed enumeration algorithm exploits $\hat{x}_1$ and $\hat{x}_2$, ignoring the outputs $\hat{x}_3$ and $\hat{x}_4$. Specifically, we use $\hat{x}_1$ to form a candidate list $\mathcal{L}_1$ for the symbol x_1 and $\hat{x}_2$ to form a candidate list $\mathcal{L}_2$ for x_2. Then, the list $\mathcal{L}_{12}$ is formed by combining all possible symbols from the candidate lists $\mathcal{L}_1$ and $\mathcal{L}_2$. Specifically, $\mathcal{L}_{12} = \{(\mathcal{L}_1[m], \mathcal{L}_2[n])\}\ \forall m \in \{1, \ldots, \sqrt{\ell}\}$ and $\forall n \in \{1, \ldots, \sqrt{\ell}\}$, where $\mathcal{L}_1[m]$ (resp. $\mathcal{L}_2[n]$) is the m-th (resp. n-th) element of the list $\mathcal{L}_1$ (resp. $\mathcal{L}_2$).

Our enumeration algorithm for the list $\mathcal{L}_1$ approximates the $\sqrt{\ell}$ closest points to the decision $\hat{x}_1$ by separating $\hat{x}_1$ into its real and imaginary components. We do so by first finding the β closest points on the real axis to $\Re\{\hat{x}_1\}$ and imaginary axis to $\Im\{\hat{x}_1\}$, respectively, where β is assumed to be an odd integer and equal to $\lfloor \ell^{1/4} \rfloor$. These β points along the real axis and imaginary axis form a square grid of β^2 candidates which form our initial approximation of $\mathcal{L}_1$. If $\sqrt{\ell}$ is larger than β^2 then additional points are enumerated beyond this grid. We propose to enumerate additional points by simply searching further

along the real and imaginary axis for $\hat{\mathbf{x}}_1$. The list $\mathcal{L}_1$ is then formed via the expression: $\mathcal{L}_1 = \{\mathcal{L}_{1^R}[m] + j\mathcal{L}_{1^I}[n]\}\ \forall m \in \{1, \ldots, \sqrt{\ell}\}$ and $\forall n \in \{1, \ldots \sqrt{\ell}\}$, where $R[m]$ and $I[n]$ denote enumerated points along the real and imaginary axes, respectively.

The intuition behind our enumeration algorithm for the list $\mathcal{L}_1$ is that we seek to approximate the $\sqrt{\ell}$ closest points to the decision $\hat{x}_1$. We approximate these $\sqrt{\ell}$ closest points by separating $\hat{x}_1$ into its real and imaginary components. We then find the β closest points on the real axis to $\Re\{\hat{x}_1\}$ and the β closest points on the imaginary axis to $\Im\{\hat{x}_1\}$, respectively, where $\beta = \lfloor \ell^{1/4} \rfloor$ and is assumed to be an odd integer. Taking all combinations of these points yields a square grid of β^2 points. If $\sqrt{\ell}$ is larger than β^2 then additional points are enumerated beyond the square grid. We propose to enumerate these points by searching along the real axis for $\Re\{\hat{x}_1\}$ and the imaginary axis for $\Im\{\hat{x}_1\}$. The list $\mathcal{L}_1$ is then formed by combining all points enumerated along both the real and imaginary axes, i.e. $\mathcal{L}_1 = \{\mathcal{L}_{x_1^R}[m] + j\mathcal{L}_{x_1^I}[n]\}\ \forall m \in \{1, \ldots, \sqrt{\ell}\}$ and $\forall n \in \{1, \ldots \sqrt{\ell}\}$, where $x_1^R[m]$ and $x_1^I[n]$ denote enumerated points along the real and imaginary axes, respectively.

Fig. 4.3 depicts the list $\mathcal{L}_1$ for a decision $\hat{x}_1 = 3 - 3j$ when $\hat{x}_1$ is an (a) 16-QAM corner point and (b) 64-QAM interior point and $\beta = 3$. The decision $\hat{x}_1$ is denoted using a gray square and the rest of the list is denoted using solid circles. Points not in the list are denoted using white circles. If $\sqrt{\ell} = 11$ in Fig. 4.3-a then the two additional points beyond $\beta^2 = 9$ are found by simply enumerating one additional candidate along the real and imaginary axes, respectively. These points are denoted using triangle markers. In Fig. 4.3-b we enumerate two additional points along both the real and imaginary axes so that $\sqrt{\ell} = 13$.

We find $\mathcal{L}_2$ in exactly the same way as $\mathcal{L}_1$. Enumerating all combinations of the $\sqrt{\ell}$ elements of $\mathcal{L}_1$ and $\sqrt{\ell}$ elements of $\mathcal{L}_2$ yields a list of pairs $\mathcal{L}_{12}$ for $\{\hat{x}_1, \hat{x}_2\}$ of size ℓ. Finally, note that the entire process for enumerating $\mathcal{L}_{12}$ may be accomplished with a lookup table to form $\mathcal{L}_1$ and also $\mathcal{L}_2$ followed by a hardware combiner to form all ℓ combinations.

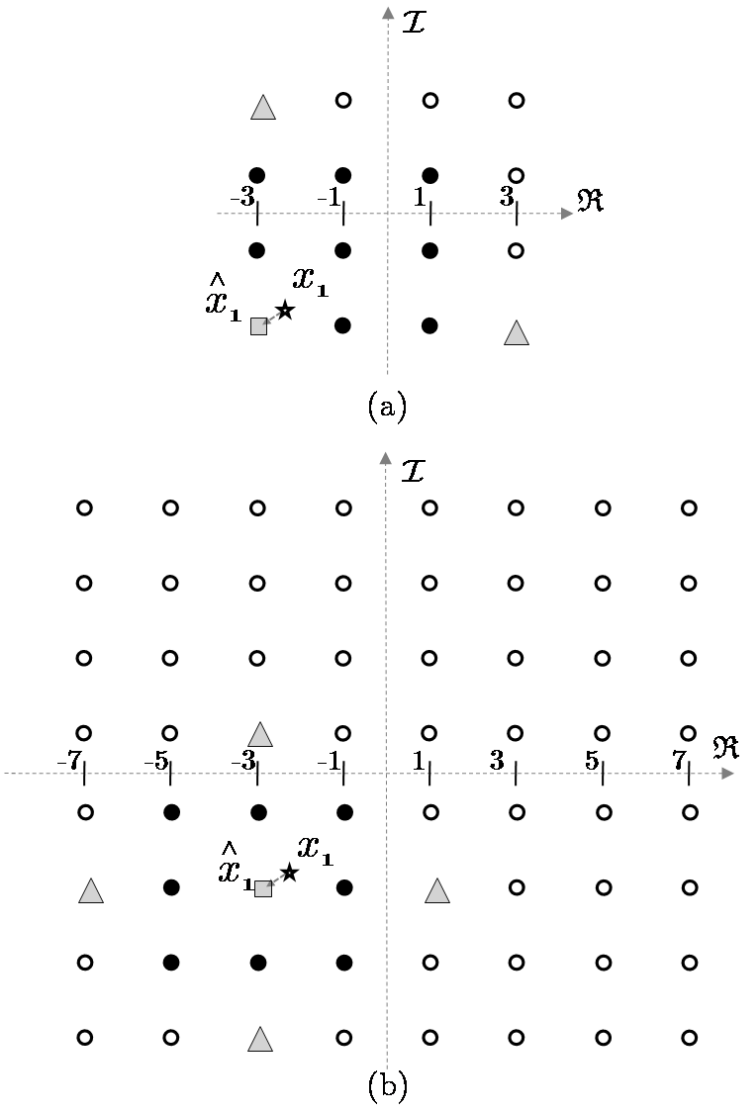

Figure 4.3: Enumerated list when $\hat{x}_1 = 3 - 3j$ and $\beta = 3$ for (a) 16-QAM with $\sqrt{\ell} = 11$ and (b) 64-QAM with $\sqrt{\ell} = 13$.

For each of the ℓ candidates in $\mathcal{L}_{12}$ we observe that:

$$\begin{bmatrix} y_3 - R_{3,1}x_1 - R_{3,2}x_2 \\ y_4 - R_{4,1}x_1 - R_{4,2}x_2 \end{bmatrix} = \begin{bmatrix} R_{3,3} & 0 \\ R_{4,3} & R_{4,4} \end{bmatrix} \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} \tilde{n}_3 \\ \tilde{n}_4 \end{bmatrix}. \tag{4.10}$$

From this we can find ℓ vectors $\mathbf{x} = [\hat{x}_1\ \hat{x}_2\ \hat{x}_3\ \hat{x}_4]^{\mathrm{T}}$ by simple decision feedback for each of the ℓ pairs of $(\hat{x}_1, \hat{x}_2)$:

$$\begin{aligned} \hat{x}_3 &= Q\left((y_3 - R_{3,1}\hat{x}_1 - R_{3,2}\hat{x}_2)/R_{3,3}\right) \\ \hat{x}_4 &= Q\left((y_4 - R_{4,1}\hat{x}_1 - R_{4,2}\hat{x}_2 - R_{4,3}\hat{x}_3)/R_{4,4}\right). \end{aligned} \tag{4.11}$$

A summary of the proposed algorithm is provided in Fig. 4.4. The inputs are the received vector $\mathbf{r}$, the effective channel matrix $\mathbf{H}$ and the list length ℓ. The output is soft-output information for each bit of the transmitted information vector.

Input: $\mathbf{r}, \mathbf{H}, \ell$ **Output**: $L(c_j|\mathbf{r})\ \forall j \in \{1, \ldots 4\omega\}$

1 $\mathbf{P} =$ `Ordering`$(\mathbf{H})$ as described in Fig. 4.2
2 $\mathbf{HP} = \mathbf{QR}$, $\mathbf{y} = \mathbf{Q}^*\mathbf{r}$, $\beta = \lfloor \ell^{1/4} \rfloor$
3 $\hat{\mathbf{x}} = Q(\mathbf{R}^{-1}\mathbf{y})$
4 $\mathcal{L}_{1^R} = \beta$ closest points on real axis to $\Re\{\hat{x}_1\}$
5 $\mathcal{L}_{1^I} = \beta$ closest points on imag axis to $\Im\{\hat{x}_1\}$
6 $\mathcal{L}_1 = \{\Re\mathcal{L}_{1^R}[m] + j\Im\mathcal{L}_{1^I}[n]\}\ \forall m \in \{1, \ldots, \beta\}$ and $\forall n \in \{1, \ldots \beta\}$
7 **if** $\sqrt{\ell} > \beta^2$ **then**
8 Alternate adding next closest candidates in row and column of $\hat{\mathbf{x}}_1$ until $\sqrt{\ell}$ candidates enumerated
9 **end**
10 Repeat steps 4 – 9 replacing x_1 with x_2 and $\mathcal{L}_1$ with $\mathcal{L}_2$ (execute only once)
11 $\mathcal{L}_{12} = \{(\mathcal{L}_1[m], \mathcal{L}_2[n])\}\ \forall m \in \{1, \ldots, \sqrt{\ell}\}$ and $\forall n \in \{1, \ldots, \sqrt{\ell}\}$
12 **for** *each of the* ℓ $\{\hat{x}_1, \hat{x}_2\}$ *in* $\mathcal{L}_{12}$ **do**
13 $\hat{x}_3 = Q\left((y_3 - R_{3,1}\hat{x}_1 - R_{3,2}\hat{x}_2)/R_{3,3}\right)$
14 $\hat{x}_4 = Q\left((y_4 - R_{4,1}\hat{x}_1 - R_{4,2}\hat{x}_2 - R_{4,3}\hat{x}_3)/R_{4,4}\right)$
15 $\mathcal{L} = \mathcal{L} \bigcup \mathbf{P}[\hat{x}_1,\ \hat{x}_2,\ \hat{x}_3,\ \hat{x}_4]^{\mathrm{T}}$
16 **end**
17 Evaluate (2.11) using $\mathcal{L}\ \forall j \in \{1, \ldots 4\omega\}$

Figure 4.4: Proposed Algorithm

4.4.3 Quantifying Complexity

We measure computational complexity using real operations, which can be multiplies, additions, comparisons, quantizers, and divisions. We assume complex multiplies require 4 real multiplies and 2 real additions, and that complex additions require 2 real additions. The computational complexity of the proposed algorithm can be separated into the preprocessing step, which is an ordered QR decomposition (line 1 from Fig. 4.4), and a core processing step. The additional complexity of the ordering algorithm over a QR decomposition is 48 real operations, where 3 of the operations are real divides.

The core processing begins by inverting $\mathbf{R}$, multiplying $\mathbf{R}^{-1}\mathbf{y}$ and quantizing the resultant vector $\hat{\mathbf{x}}$ (line 3 from Fig. 4.4). The computation of $\mathbf{R}^{-1}$ requires 104 real operations where 4 of the operations are real divides. As $\mathbf{R}^{-1}$ is triangular, the multiplication $\mathbf{R}^{-1}\mathbf{y}$ requires 44 real operations. Then the quantization to $\hat{\mathbf{x}}$ requires 8 real quantizers and so line 3 requires 156 real operations for the entire information vector or $156/2 = 78$ real operations per signaling interval.

The enumeration of the list $\mathcal{L}_{12}$ (lines 4-11) can be accomplished with a lookup table and a dedicated hardware combiner used to form all combinations of the lists for each symbol, i.e. $\mathcal{L}_{12} = \{(\mathcal{L}_1[m], \mathcal{L}_2[n])\}\ \forall m \in \{1, \ldots, \sqrt{\ell}\}$ and $\forall n \in \{1, \ldots, \sqrt{\ell}\}$. The remaining steps (lines 13-17) involve decision feedback, permutation and the computation of the soft-output information, where many of the intermediate variables used in the decision feedback computation may be reused for computation of the soft-output information. Line 13 requires 18 real operations (2 complex multiplies, 2 complex adds and 2 real divides) and line 14 requires 26 real operations (3 complex multiplies, 3 complex additions and 2 real divides) for a total of 44ℓ real operations per information vector.

In total lines $13 - 14$ require 44ℓ real operations per information vector. Computing the cost for each of the ℓ candidates in line 17, assuming reuse of computations between lines $13 - 14$ and line 17, requires an additional 44ℓ real operations per information vector for a total of $44\ell + 44\ell = 88\ell$ real operations per information vector or $88/2 \cdot \ell = 44\ell$

real operations per signaling interval. Consequently, the total number of real operations for the core processing plus the overhead required by the ordering algorithm over that of a QR decomposition, for one signaling interval, is $78 + 44\ell$.

4.5 Results

We now present results for the proposed algorithm. We assume the channel does not change during the duration of an entire frame and that the channel matrix entries are drawn anew with the transmission of each new frame. A convolutional code with code polynomial [133 171] and constraint length 7, punctured to code rate 3/4 is employed and the information block size (including tail bits) is 3456. We employ a max-log convolutional decoder. Performance is measured in terms of the E_b/N_0 in dB required to achieve a frame error rate (FER) of 10^{-2}.

Fig. 4.5 depicts performance results for a system employing 16-QAM transmission for $\ell = 121$. Also depicted is the joint maximum-likelihood (JML) detector and the K-best detector [101] employing MMSE-SQRD preprocessing for $K = 256$. We observe that the proposed algorithm with $\ell = 121$ is 1.7 dB worse than the K-best detector, where the K-best detector requires over 3 times the complexity, in terms of real operations. Also depicted in Fig. 4.5 is the G-LORD detector [81] with parameter $k = 2$, i.e. full enumeration at the first two layers which implies $\ell = 256$. The proposed algorithm is 0.9 dB worse than the G-LORD detector with $\ell = 256$, but the G-LORD detector is over 80% more complex. We also provide a curve listed as "No approx, $\ell = 121$, SQRD" demonstrating the significant impact of our proposed ordering algorithm. The curve was found by forming $\mathcal{L}_{12}$ with the 11 *best* decisions, relative to the received signal $\mathbf{r}$, for x_1 and x_2, respectively.

Fig. 4.6 depicts performance results for our proposed algorithm in a system employing 64-QAM transmission for $\ell = 169$ (i.e. $\beta = 3$ and $\ell_{x_1} = \ell_{x_2} = 13$). Also depicted is the hard-output joint maximum-likelihood (JML) detector, the K-best detector [101] employing MMSE-SQRD preprocessing and $K = 512$, and the G-LORD detector [81] with

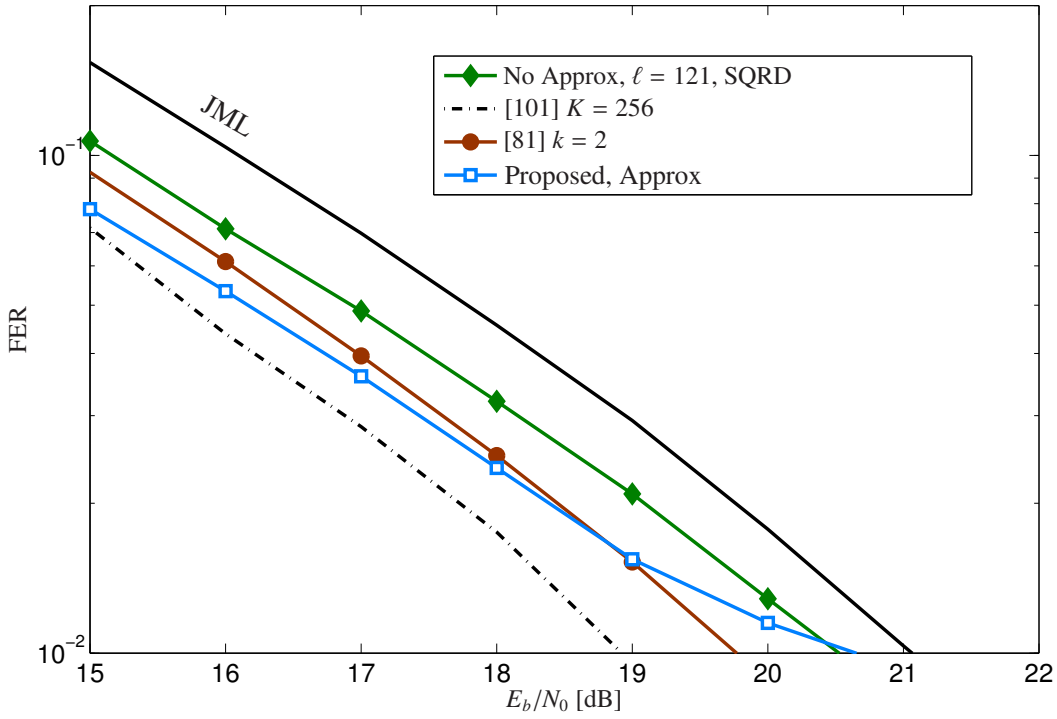

Figure 4.5: Soft-output golden code performance for 16-QAM.

parameter $k = 2 \Rightarrow \ell = 4096$. We observe that the proposed algorithm with $\ell = 169$ is 1.2 dB worse than the K-best detector and 0.8 dB worse than the G-LORD detector. We note however, that the G-LORD detector is roughly 21 times more complex. Here the K-best reference curve is only slightly better performing than the G-LORD detector and requires more than 140 times the complexity of the proposed algorithm with $\ell = 169$.

Table 4.5 depicts the E_b/N_0 required to achieve a FER of 10^{-2} and the computational complexity for [81], [101] and the proposed algorithm, where complexity is measured in terms of the number of real operations performed per signaling interval. The complexity of the SQRD is ignored for all algorithms, but for the proposed algorithm we include the additional overhead of our ordering algorithm in the computational complexity. Table 4.5 supports the claim that the proposed algorithm has a desirable performance-complexity trade-off. Specifically, results indicate that the proposed algorithm is particularly attractive for 64-QAM transmission.

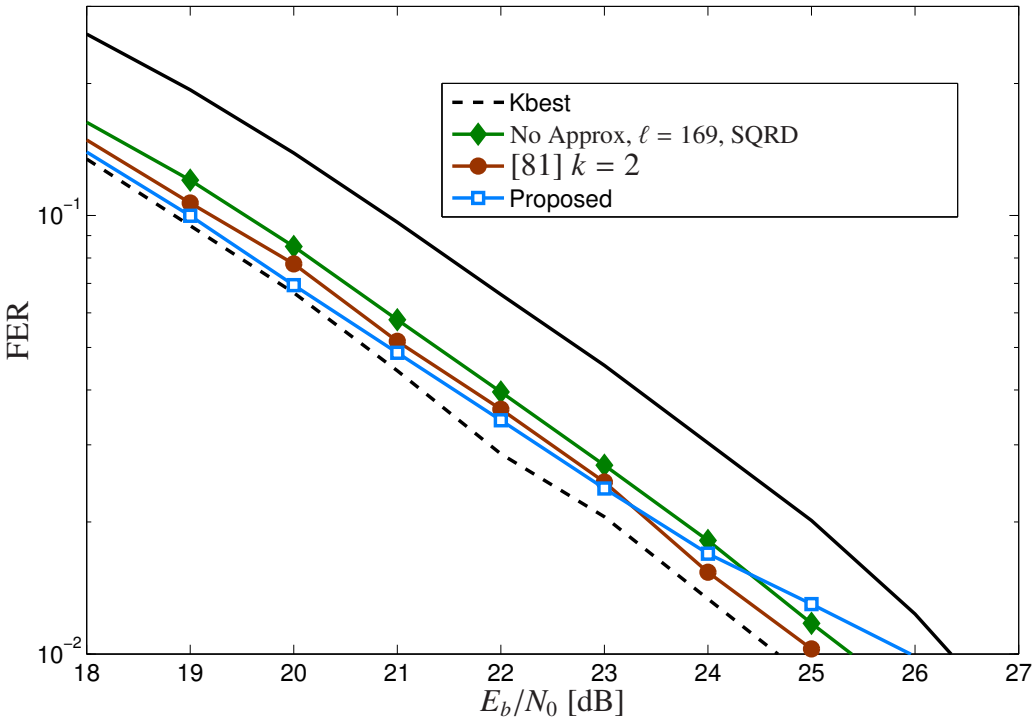

Figure 4.6: Soft-output golden code performance for 64-QAM.

16-QAM			
Algorithm	Prop.	[81]	[101]
ℓ	121	256	256
E_b/N_0 [dB] for FER 10^{-2}	20.6	19.8	18.9
Real Ops.	4696	9032	16738
64-QAM			
Algorithm	Prop.	[81]	[101]
ℓ	169	4096	512
E_b/N_0 [dB] for FER 10^{-2}	25.9	25.1	24.7
Real Ops.	6520	137440	931587

4.6 Summary

The golden code is an important algebraic space time code that achieves the full diversity-multiplexing frontier of Zheng and Tse [107, 85] and is included in the 802.16e WiMAX standard. In this chapter we proposed a low- and fixed- computational complexity soft-output detection algorithm for an important algebraic space-time code known as the golden code. The detector involved a preprocessing algorithm, as proposed in [66], that approximately maximize the SNR for the first pair of detected information symbols. Next, we employed linear equalization to simplify the task of finding a list of candidate values for these first two information symbols, and then – for each pair of symbols on the list – using decision-feedback equalization to find candidate values for the remaining pair of information symbols. Numerical results indicated that the proposed soft-output list detection algorithm is significantly less complex than previously reported algorithms, yet performs comparably.

CHAPTER 5

LOG-LIKELIHOOD RATIO CLIPPING

5.1 Introduction

The reliability of the soft-output information produced by a MIMO detector significantly impacts system error rate performance. An ideal soft-output detector computes the exact a posteriori probabilities. However, as we have seen in prior chapters, practical MIMO receivers achieving high spectral efficiencies must resort to sub-optimal detection schemes in order to avoid the burdensome computational complexity of exact APP detection. Many modern communication devices, such as inexpensive mobile phones, employ suboptimal MIMO detection thereby sacrificing significant error rate performance for the sake of saving computational complexity (and thus production cost).

Suboptimal list MIMO detectors often suffer from the inability to compute exact soft-output information for some (or even any) of the transmitted bits. However, for even small list sizes, a reasonably precise estimate of the optimal detector's hard-output can be determined. For each hard detected bit the sign and magnitude for the LLR can be determined. The *sign* of the log-likelihood ratio can be determined with a reasonably low probability of error and, as the list length increases, the sign of the LLR is known with a diminishing amount of uncertainty. The LLR *magnitude*, however, often remains imprecise for small list sizes. This is particularly relevant problem for list MIMO detection, where there is a non-zero probability that counterhypotheses are missing for some of the detected bits. In such a situation, one has to resort to estimation for the corresponding LLR value. Numerous techniques have been proposed to address this issue [70] such as:

- LLR clipping [47, 28, 108]: the maximum magnitudes for LLR information is assumed to be fixed to a certain predefined value. This strategy is extremely simple

to implement, but the achievable performance depends crucially on an appropriate selection of the clipping level.

- Bit flipping/Chase decoding [93]: a counter-hypothesis is generated by taking the MAP estimate, flipping the bit of interest and calculating the Euclidean distance of the resulting hypothesis. Achieving good performance with this approach requires high computational complexity. This is because each additional bit flipped requires an exponential increase in computational complexity. As the detection layers are coupled the counterhypotheses resulting from bit flipping are often of poor quality.

- Last list entry: one may use the last entry of the list (the one with largest Euclidean distance) as a lower bound on the Euclidean distance of the counter-hypothesis. However, this bound is rather loose such that using it will cause the LLRs to be "clipped" very aggressively, causing a significant performance loss [61].

- Path augmentation [10]: the dead-ends of the tree structure can be extended based on the output of a linear filter and/or available a priori information. This method is also rather complex and yields poor estimates if no a priori information is available [108].

- Worst case distance [87]: An effective technique is to add the fixed worst-case distance to an ML/MAP distance metric. Based on the results in [87] this approach appears to perform better than [61] with similar complexity

While all of the aforementioned approaches address the problem of an unknown (or unreliable) LLR magnitude, LLR clipping is the simplest solution.

The selection of the clipping level has a significant impact on the achievable performance in coded communication systems [28]. Choosing the clipping level too high induces the decoder to assume overly high reliability for bits with missing counter-hypotheses, potentially preventing decision errors occurring at these bit positions from being corrected. Conversely, setting the clipping level too low substantially distorts the soft output for bits

with counter-hypotheses, also leading to a performance loss [70]. In other words, small clipping values tend to degrade error-correction capabilities and large clipping values can result in error propagation.

Most previous approaches for computing the LLR clipping level rely upon selecting a fixed clipping level, usually based on an attempt to maximize the mutual information at the detector output over the choice of the clipping level. Such fixed LLR clipping (FLC) level schemes possess the obvious advantage that, once the FLC level has been selected, they are easy to implement. Drawbacks of FLC schemes include an involved selection process for the clipping level [28, 108]. Finally, and perhaps most importantly, these approaches are limited in the error rate they can achieve.

In [72] it was shown that LLR clipping has more effect on system performance when using suboptimal algorithms such as the M algorithm with $b = \infty$ (i.e. the K-best algorithm [3, 101, 39]). Intuitively this makes sense because when the soft information is exact it is unnecessary to clip what is already an optimal LLR value. Consequently, in this chapter we adopt the K-best algorithm as our representative suboptimal detection algorithm. Results show that the proposed SLC scheme outperforms even the best FLC schemes for coded MIMO communication systems. A promising recent submission [106] provides an approach for calculating the LLR clipping level that is similar to the SNR-aware LLR clipping approach described in this chapter. More details on this work are found in the further reading section at the end of this chapter.

In this chapter, after reviewing prior art on FLC schemes in section 5.2, we propose a low complexity approach for computing *SNR-aware LLR clipping (SLC)* levels in section 5.3, based on an estimate of the bit error probability at the detector output. Specifically, for a given channel realization and list length, we calculate the instantaneous error probability on the different detected layers. This information is then used to predict the LLR clipping level. The additional complexity of the proposed approach can be considered almost

negligible in coded systems. Section 5.4 presents results demonstrating the improved performance of the SLC approach over FLC. Suggestions for future work and further reading are provided in section 5.5 and a summary is provided in 5.6.

5.2 Fixed LLR Clipping Level (FLC)

For suboptimal list detectors employing fixed LLR clipping it is essential that the clipping level, denoted L_{clip}, be selected carefully. If the clipping level is chosen too high, this prevents decision errors occurring at some bit positions from being corrected, resulting in poor performance. Conversely, setting the clipping level too low limits the mutual information at the detector output, also leading to decreased performance [70].

Early FLC approaches relied on trial and error for choosing L_{clip}. An example of this is [47], which set $L_{\text{clip}} = 8$, after observing good performance results for this choice. Setting L_{clip} to 8 implies extreme confidence when detecting a transmitted bit to be a 0 (-8 LLR) or a 1 (+8 LLR). In fact, $L_{\text{clip}} = 8$ can be considered to be a reasonable upper bound, as clipping the LLRs to this level has a negligible impact on the mutual information if the LLRs are exact i.e., obtained from an optimal detector [108].

More recent approaches for determining FLC values have relied upon maximizing the average mutual information at the detector output for specific system configurations [28, 108]. Assuming genie knowledge of the transmitted bits c_i in the code bit stream, this mutual information can be determined from the calculated LLRs $L_p(c_i)$ using [40]:

$$I(\mathbf{c}; L_p(\mathbf{c})) \approx 1 - \frac{1}{N_C} \sum_{n=1}^{N_C} \log_2 \left(1 + \exp\left(-c_i \cdot L_p(c_i)\right)\right),$$

where $N_C \gg 1$ is the number of bits in the codeword. Using a mutual information based approach, it was shown in [28] that for many cases, $L_{\text{clip}} = 3$ is a reasonably good choice. This work was elaborated upon in [108] to show that the optimal clipping level depends on the list size and modulation alphabet. This is because the probability that counter-hypotheses are not available increases as the list size decreases. Analysis in [108] shows that the mutual information between channel input and detector output is a suitable figure

of merit for optimization, and that the optimal LLR clipping level will strongly depend on the system setup as well as the chosen detector configuration.

5.3 SNR-aware LLR Clipping (SLC)

We can extend the fixed clipping approach just described to one where L_{clip} is based on the available channel state information (CSI). Consider again the definition of the LLRs from (2.2), which can be restated as:

$$\begin{aligned} L(c_j|\mathbf{r},\mathbf{H}) &:= \ln \frac{\Pr\left[c_j = +1|\mathbf{r},\mathbf{H}\right]}{\Pr\left[c_j = -1|\mathbf{r},\mathbf{H}\right]} \\ &= \hat{c}_j \ln \frac{\Pr\left[c_j = \hat{c}_j|\mathbf{r},\mathbf{H}\right]}{\Pr\left[c_j \neq \hat{c}_j|\mathbf{r},\mathbf{H}\right]} \\ &= \hat{c}_j \ln \frac{\Pr\left[c_j = \hat{c}_j|\mathbf{r},\mathbf{H}\right]}{1 - \Pr\left[c_j = \hat{c}_j|\mathbf{r},\mathbf{H}\right]} \end{aligned} \tag{5.1}$$

where $\hat{c}_j$ is the hard output estimate of the considered bit, as obtained from the detector. In the absence of a counter-hypothesis for this bit, the expression inside the logarithm of (5.1) is unknown. Furthermore, the knowledge of the Euclidean distance $d(\hat{\mathbf{a}}) = \|\mathbf{r} - \mathbf{H}\hat{\mathbf{a}}\|^2$ and associated $p(\mathbf{r}|\hat{\mathbf{a}})$ is not sufficient to establish a precise estimate for $\Pr\left[c_j = \hat{c}_j|\mathbf{r},\mathbf{H}\right]$, as calculating the normalization factor $\Pr(\mathbf{r})$ required to determine $\Pr[\hat{\mathbf{a}}|\mathbf{r},\mathbf{H}]$ is computationally complex.

We therefore propose to resort to an approximation of (5.1), by averaging out the influence of $\mathbf{r}$. With this simplification, expression (5.1) becomes:

$$\begin{aligned} L(c_j|\mathbf{r},\mathbf{H}) &\approx \hat{c}_j \ln \frac{\Pr\left[c_j = \hat{c}_j|\mathbf{H}\right]}{1 - \Pr\left[c_j = \hat{c}_j|\mathbf{H}\right]} \\ &= \hat{c}_j \ln \frac{1 - P_b(\mathbf{H})}{P_b(\mathbf{H})}, \end{aligned} \tag{5.2}$$

where P_b is the bit error probability at the detector output, which needs to take into account the CSI, the current SNR, the modulation format, and the configuration of the detector. This predicted bit error probability will be denoted as $P_b(\ell, \text{SNR}_i)$ in the following, where

SNR_i is the instantaneous SNR for the ith detection layer (ith component of the transmit signal) and given by:

$$\text{SNR}_i = \frac{E_s}{N_t N_0} R_{i,i}^2, \tag{5.3}$$

where $R_{i,i}$ is the ith diagonal element of a lower triangular matrix $\mathbf{R}$ resulting from a QR-decomposition of the channel matrix, $\mathbf{H} = \mathbf{QR}$. To model the improvement in the quality of the detector output for larger list sizes, $P_b(\ell, SNR_i)$ is therefore given as a function of the list length ℓ.

This predicted error probability $P_b(\ell, \text{SNR}_i)$ yields our SNR-aware L_{clip} value for bits in the ith detection layer:

$$L_{clip,i} := \ln \frac{1 - P_b(\ell, \text{SNR}_i)}{P_b(\ell, \text{SNR}_i)} \approx -\ln P_b(\ell, \text{SNR}_i). \tag{5.4}$$

In [112] it was proven that the optimal LLR clipping level, for BPSK transmission over the AWGN channel and repetition codes of arbitrary rate, is of the form of the definition (without approximation) in (5.4). In such a situation the list length is either one, in the absence of a counterhypothesis, or two, in which case the exact LLR is known.

To illustrate the capabilities for our SLC approach, we consider the error probabilities for PAM and QAM modulation. Specifically, the symbol error probability for the maximum-likelihood detector in the case of pulse amplitude modulation (PAM) transmission over an AWGN channel with effective signal-to-noise ratio $\sqrt{\ell}\text{SNR}_i$ is given by:

$$P_{s,PAM} = 2\left(1 - \frac{1}{\sqrt{q}}\right) Q\left(\sqrt{\frac{3}{q-1}\sqrt{\ell}\text{SNR}_i}\right). \tag{5.5}$$

Note that since one-dimensional PAM modulation is considered in (5.5), we elect to use $\sqrt{\ell} = \sqrt{K}$ to compute this expression, where K is the parameter for the representative K-best tree search detection scheme [101, 39]. Performance results support such a selection[1]. Plugging $P_{b,QAM}$ into (5.4) yields improved LLR values in the absence of a counter-hypothesis, relative to FLC.

[1] Another option, which yielded good results was to use γ_ℓ, the square root of γ_ℓ^2 as detailed in [98]

Following standard QAM extensions of the PAM SER expression yields the QAM symbol error rate:

$$P_{s,QAM} = 1 - (1 - P_{s,PAM})^2 \tag{5.6}$$

from which the QAM bit error rate can be easily obtained as $P_{b,QAM} \approx P_{s,QAM}/L$ (and equivalent for the PAM BER). Plugging these error rate expressions back into (5.4) allows for an improved LLR clipping level, relative to fixed LLR clipping. Note that (5.4) is a general expression capable of working with a host of modulation and detection schemes, provided that proper treatment is given to the application of the effective SNR gain due to an increase in the list length.

The probability density function for L_{clip} is shown in Fig. 5.1 for a 4×4 MIMO system in i.i.d. Rayleigh fading using (a) 4-QAM and (b) 64-QAM transmission with $K = \ell = \{1, 4, 8\}$ and $K = \ell = \{1, 4, 16\}$ [101, 39], respectively. Results shown are for SNR values corresponding to a coded BER of 10^{-5} for the given list length using sorted MMSE preprocessing [109] (cf. Fig 5.2 and Fig 5.3). The plot was obtained using L_{clip} values found for over $400,000$ distinct channel realizations and the approximation in (5.4) was used.

We now provide a brief description of the complexity aspects for the SLC approach. Specifically, the complexity of the approach is the complexity required to compute (5.4) for each detection layer and channel realization. While the complexity associated with such a computation typically involves complicated calculations such as the $Q(\cdot)$ function and square root operation, as in (5.6), the practical complexity of SLC can be significantly reduced through the use of a table lookup, or approximation like the one in (5.4).

5.4 Results

We consider transmission over a spatially and temporally i.i.d. fading 4×4 MIMO channel, using 4 and 64-QAM alphabets. The information block size (including tail bits) is 9216 bits. Detection is performed based on the real-valued system model. We employ unbiased

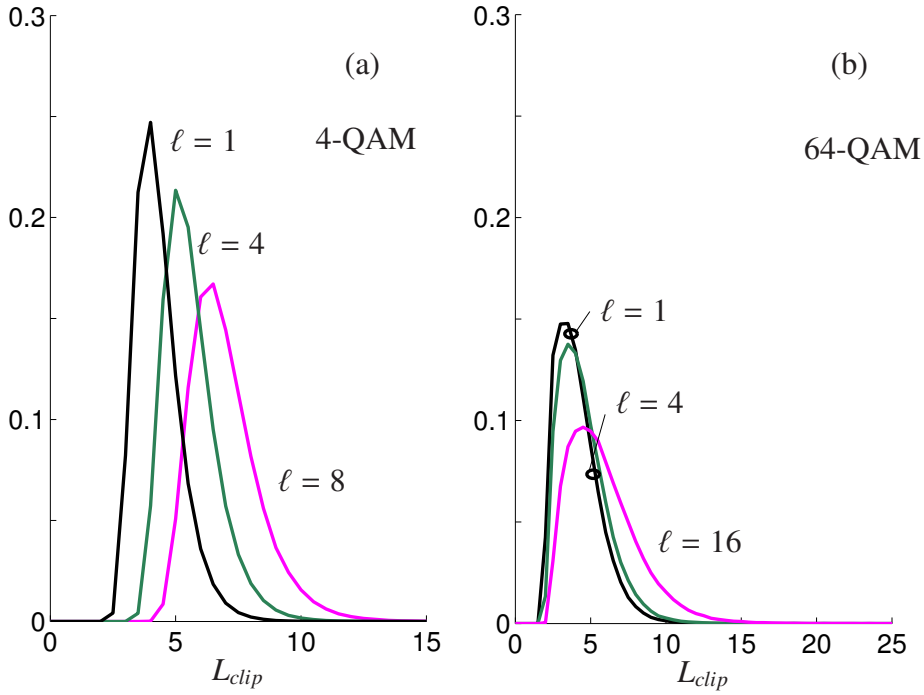

Figure 5.1: The pdf for L_{clip} for i.i.d Rayleigh fading for a 4×4 MIMO system using (a) 4-QAM and (b) 64-QAM transmission and $K = \ell = \{1, 4, 8\}$ and $K = \ell = \{1, 4, 16\}$, respectively. The maximum values for L_{clip} are not shown due to precision issues which force these values to be infinite.

MMSE detection as detailed in (2.19) with all techniques. For coded transmission, we use a setup equivalent to the one in [47]: a rate 1/2 PCCC based on $(7_R, 5)$ convolutional codes using 8 internal iterations of logMAP decoding.

Fig. 5.2 provides a performance comparison of the proposed SLC approach and fixed-valued LLR clipping using the simulation setup just described in the case when there are no iterations between the detector and the decoder (i.e. only decoder iterations) and 4-QAM transmission. All results shown are obtained using the K-best Algorithm for the case when $K = \ell = \{1,\ 2,\ 4,\ 8\}$ and employ the approximation found in (5.4). In all cases, for the same detection algorithm (i.e. same value of ℓ), our SNR-aware approach outperforms the FLC approaches. As an example, when $\ell = 1$, the SLC approach outperforms the clipping of ± 8 proposed in [47] by 0.5 dB and the clipping of ± 3 prosed in [28] by 0.3 dB at a BER of 10^{-5}. Furthermore, the K-best for $\ell = 8$ shown in Fig. 5.2 is used to demonstrate that the SLC approach with $\ell = 4$ is roughly equivalent to the performance of the higher

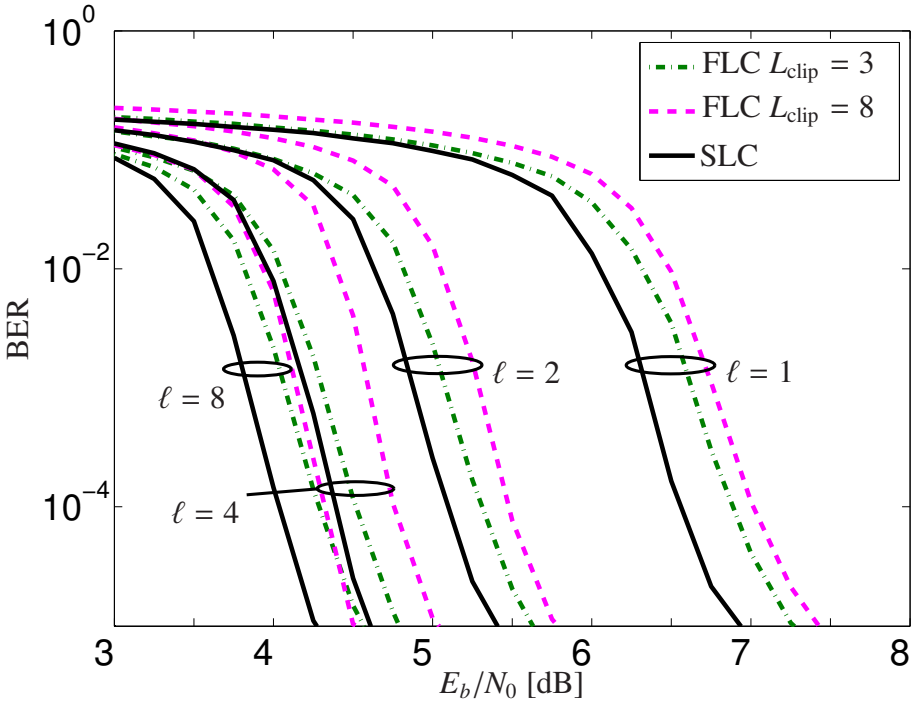

Figure 5.2: SNR-aware LLR clipping versus fixed LLR clipping for a 4-QAM coded non-iterative system for a 4×4 MIMO system under Rayleigh fading.

complexity $\ell = 8$ algorithm when using FLC.

Fig. 5.3 provides the same performance comparison between the SLC approach and fixed-valued LLR clipping, this time for the case of 64-QAM. Results are obtained for the K-best algorithm for $K = \ell = \{1,\ 2,\ 4,\ 16 \text{ and } 64\}$ and employ the approximation found in (5.4). Again, in all cases, for the same detection algorithm, the SLC approach outperforms the fixed LLR clipping approaches.

5.5 Further Reading

Recently, it is has been shown that the LLR clipping is closely related to the probability of error at the detector output. [112] uses an optimization problem to determine the LLR clipping level, where the clipping level should be chosen by the detector such that the probability of error at the output of the channel decoder is minimized. Results are provided for the case of repetition codes transmitted over the AWGN channel using antipodal signaling and repetition codes of arbitrary rates. Specifically, it is proven that for this setup, the

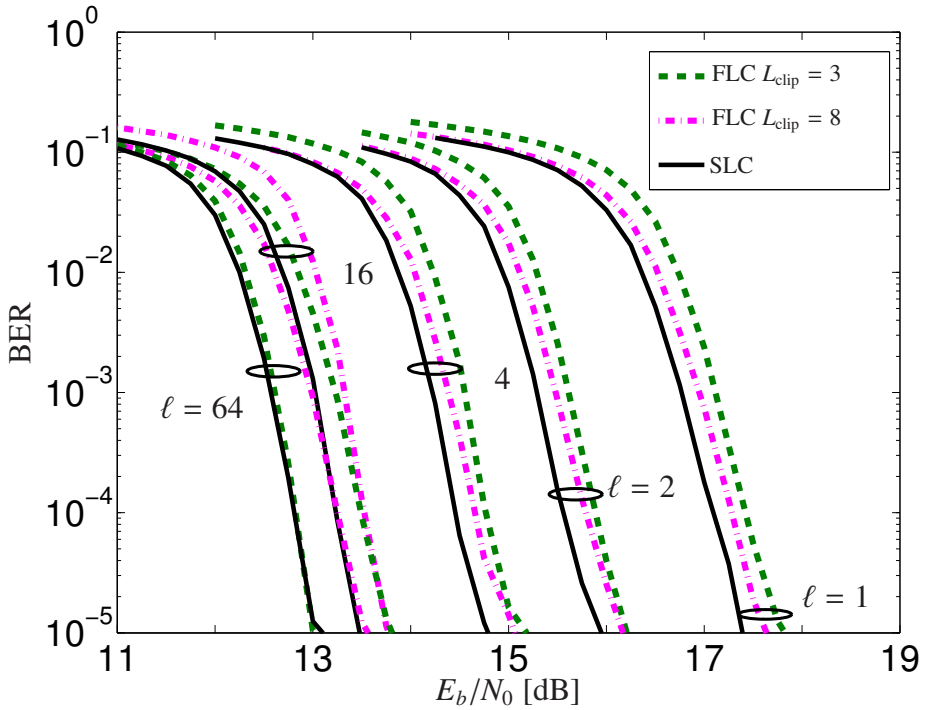

Figure 5.3: SNR-aware LLR clipping versus fixed LLR clipping for a 64-QAM coded non-iterative system for a 4×4 MIMO system under Rayleigh fading.

optimal LLR clipping level Λ^* is given by:

$$\Lambda^* = \ln \frac{1 - P_e}{P_e}, \tag{5.7}$$

where P_e is the probability of error at the detector output.

A promising recent submission [106] provides an approach for calculating the LLR clipping level that is similar to the SNR-aware LLR clipping approach described in this chapter. However, [106] differs from the SLC approach in a number of key areas. First, the iterative tree search (ITS) detector [28] is used as the representative detection algorithm for deriving the LLR clipping level instead of K-best detector [3, 101, 39] from this chapter. Second, [106] estimates the clipping value based on an approximation that a channel bit equals the log ratio of the maximum probability of a correct decision for the bit to the probability of an erroneous decision. This log ratio closely resembles (5.7) and is directly related to the plot of mutual information found in [28] in an elegant way.

5.6 Summary

In this chapter, we presented a scheme for computing the log-likelihood ratio clipping level for suboptimal MIMO list detection. This problem was framed as determining variable LLR clipping levels conditioned on both the CSI and the list length of the detector. QAM error rate performance over i.i.d. Rayleigh fading MIMO channels indicated that the proposed SNR-aware LLR clipping approach outperformed fixed LLR clipping schemes.

An open problem is extending the optimality proof of [112] to different codes, fading models and MIMO systems. The work in [106] is a promising start in this direction.

CHAPTER 6

NON-UNIFORM COMPUTATIONAL COMPLEXITY ALLOCATION

6.1 Introduction

So far in this dissertation we have concerned ourselves with the problem of list detection in the context of soft-output MIMO detection. In this chapter we again consider the problem of list detection, but in the context of hard-output detectors for multiantenna orthogonal frequency-division multiplexing (MIMO-OFDM) systems. Hard-output list detectors are those that first find a candidate list and then select the minimum cost element from the list to be the hard decision.

OFDM signaling is particularly useful for combatting wideband communication channels, where frequency-selective fading causes the quality of the channel to vary significantly from one OFDM subcarrier to the next. It is common practice for MIMO-OFDM detectors to implement the same detector at each subcarrier, in which case the overall performance is dominated by the weakest subcarrier. We propose a receiver strategy for MIMO-OFDM channels called *nonuniform computational complexity allocation (NCCA)*, whereby the receiver adapts the computational resources of the MIMO detector at each subcarrier to match a metric of the corresponding channel quality. For concreteness, we investigate this architecture in the special case when each subcarrier uses the B-Chase detector [98] with each list length is dependent on the subcarrier SNR. The performance of each B-Chase detector is compactly captured by the so-called list detection error probability. We compute an exact expression and upper bound for the list detection error probability in AWGN. We then propose an algorithm for assigning list lengths to subcarriers that aims to minimize the maximum list detection error probability over the different subcarriers.

Numerical results for a 4-input 4-output Rayleigh-fading channel with 16-QAM show that the proposed MIMO-OFDM detector outperforms a conventional detector with the same complexity by 4.4 dB.

Orthogonal frequency-division multiplexing (OFDM) is a proven strategy for communication over frequency-selective channels, whereby information is conveyed via multiple subcarriers that are mutually orthogonal [99, 5]. The use of OFDM transforms a frequency-selective MIMO channel into a parallel bank of flat-fading MIMO channels, one for each subcarrier. This transformation enables the use of a memoryless MIMO detection algorithm (i.e., one designed for a flat-fading channel) in a frequency-selective setting by simply applying the same detection algorithm at each subcarrier.

Unlike a conventional MIMO-OFDM receiver that uses the same detection algorithm at each subcarrier, and thus *uniformly* distributes computational processing across the different subcarriers, this paper proposes a *nonuniform* strategy for distributing a complexity budget across subcarriers called *nonuniform computational complexity allocation (NCCA)*. The idea is to use low-complexity detectors for those subcarrier channels having high quality and high SNR, and to reserve higher-complexity detectors for those channels having low quality and low SNR. In a sense, for a given overall complexity budget, the weaker subcarriers can borrow processing resources from the stronger ones. This concept is roughly analogous to the "just enough" philosophy that underlies power control in wireless communications, where a transmitter will adjust its signal power in accordance with the path loss to the receiver so that the received signal power is sufficient for reliable communication, but not greater [4]. Here we translate this philosophy to processing power instead of signal power.

Several existing detectors can be viewed as special cases of NCCA. For example, the MIMO-OFDM detector of [54] uses the M-algorithm for detection on each subcarrier, with the value of the M parameter chosen to match a measure of SNR for the corresponding subcarrier. Similarly, any OFDM detector that uses a variable (i.e. not fixed) complexity

detector at each tone, such as a sphere decoder [92, 26], lattice-reduction-aided detector [104, 79], or sequential decoder [3, 23], can be viewed as another instance of the NCCA approach, because the complexity at each subcarrier is a random variable dependant on channel conditions. In all previously reported cases, however, the complexity is assigned independently for each subcarrier, without coordination, and as a result the total complexity can vary with time.

The NCCA framework is streamlined when the MIMO detectors at the different subcarriers are all selected from the same family of detection algorithms, and when this family is parameterized by a single parameter that trades off performance for complexity. Examples of such detector families include the M-algorithm [3], the stack algorithm [3, 23], and the bit-level Chase algorithm [59]. For concreteness we investigate the NCCA framework using the *B-Chase* family of MIMO detectors [96, 97]. Recall that the B-Chase(ℓ) detector is a hard-output detector for memoryless MIMO channels that generates a *list* of ℓ tentative decisions for one of the transmitted symbols, and implements a bank of ℓ ordered decision-feedback detectors in parallel, one for each element of the list. The final decision vector is the decision-feedback detector output that minimizes mean-squared error.

The overall error probability of B-Chase detection is dominated by the list detection-error probability of the scalar list detector at the first stage, where we define a *list detection error* as the event that the transmitted symbol does not appear on the decision list [17]. *We therefore use the list detection error probability as a tool for assigning list lengths to the different subcarriers*. In contrast to prior works, the list lengths allocated to each subcarrier are optimized jointly in accordance with a fixed overall complexity budget. Initial analysis for the list detection error probability was provided in [96], which investigated the list detector decision regions for quadrature-amplitude modulation (QAM), and demonstrated that increasing the list length leads to an effective gain in SNR.

The remainder of this chapter is organized as follows. Section 6.2 incorporates OFDM into the MIMO system model. Section 6.3 computes the exact single-input single-output

(SISO) list detection error probability for AWGN channels and provides a minimum-distance approximation for the list detection error probability. Section IV describes the strategy, including an algorithm for allocating computational resources to the different subcarriers based on the list detection error probability. Section V presents numerical results for a Rayleigh-fading frequency-selective MIMO channel. Section VI concludes the paper.

6.2 OFDM System Model

We once more consider a frequency-selective MIMO channel with N_t transmit antennas and N_r receive antennas. Differing from prior chapters is the fact that transmitter utilizes J OFDM subcarriers, so that the frequency-selective channel is transformed into J memoryless N_t-input, N_r-output channels:

$$\mathbf{r}^{(i)} = \mathbf{H}^{(i)}\mathbf{a}^{(i)} + \mathbf{n}^{(i)}, \tag{6.1}$$

where the subcarrier index $i \in \{1, \ldots J\}$, $\mathbf{a} = [a_1^{(i)}, \ldots a_{N_t}^{(i)}]^{\mathrm{T}}$ is the vector of transmitted symbols for subcarrier i, $\mathbf{r}^{(i)} = [r_1^{(i)}, \ldots r_{N_r}^{(i)}]^{\mathrm{T}}$ is the vector of received samples for subcarrier i, $\mathbf{H}^{(i)}$ is the $N_r \times N_t$ memoryless channel matrix for subcarrier i , and $\mathbf{n}^{(i)}$ is the noise vector for subcarrier i. Once more we assume that the channel inputs are chosen uniformly and independently from the same q-ary QAM alphabet $\mathcal{A} = \{\pm\alpha, \pm 3\alpha, \ldots \pm (\sqrt{q} - 1)\alpha\} + \sqrt{-1}\{\pm\alpha, \pm 3\alpha, \ldots \pm(\sqrt{q}-1)\alpha\}$, where $q = |\mathcal{A}|$, $\alpha = \sqrt{3E_a/(2(q-1))}$, and E_a is the alphabet energy. In this chapter we define the SNR per bit to be $E_b/N_0 = \mathcal{E}[|H_{iv}|^2]E_a/(N_0 \log_2(q))$, where H_{iv} is the element in the ith row and vth column for a given subcarrier channel matrix.

6.3 List Detection Error Probability

The purpose of this section is to quantify the performance of the B-Chase detector by evaluating the list detection error probability; these performance results will be used in the next section to facilitate the allocation of computational resources to the different subcarriers. However, because the utility of the list detection error probability may extend to other applications unrelated to distributed complexity allocation, computing this probability is an

interesting and relevant problem in its own right.

6.3.1 Exact Analysis

A key problem for all decision-feedback detectors on fading channels is the minimal diversity gain for the first symbol detected, which leads to a large probability of error for this symbol. This larger error probability dominates the overall error-rate. The B-Chase detector overcomes this bottleneck by considering $\ell > 1$ possibilities for the first symbol, implementing a separate decision-feedback detector for each of the ℓ possibilities, and choosing the best of the resulting candidate decision vectors [96].

The B-Chase detector begins by identifying which of the N_t transmitted symbols will be detected first. It then uses a linear filter to null (in the ZF case) or partially null (in the unbiased MMSE case) the contributions of the $N_t 1$ interfering symbols, yielding a scalar

$$r = a + n, \tag{6.2}$$

where $a \in \mathcal{A}$ is the transmitted symbol of interest, and n is the combined noise and residual interference (if any). In the following we will assume that the real and imaginary components of n are i.i.d. $\mathcal{N}(0, \sigma^2)$ random variables and independent of the transmitted symbol a, an assumption that is strictly true only in the ZF case [77]. The SNR at this point is thus $SNR = \mathcal{E}[|a|^2]/\mathcal{E}[|n|^2] = E_a/(2\sigma^2)$.

After determining the order of detection, the B-Chase detector then implements a *list detector* for the first detection symbol, where a list detector is defined as a device that accepts a scalar input r and generates an ordered list of the ℓ alphabet symbols that are closest to r. In other words, if for each r we define $(\lambda_1, \lambda_2, \ldots \lambda_q)$ as a permutation of the alphabet $\mathcal{A} = \{a_1, a_2, \ldots a_q\}$ satisfying

$$|r - \lambda_1| \leq |r - \lambda_2| \leq \ldots \leq |r - \lambda_q|, \tag{6.3}$$

then the decision list is simply the first ℓ elements of the permutation, $\mathcal{L} = \{\lambda_1, \lambda_2, \ldots \lambda_\ell\}$. In the special case when $\ell = 1$, the list detector reverts to the familiar minimum distance (maximum likelihood) detector.

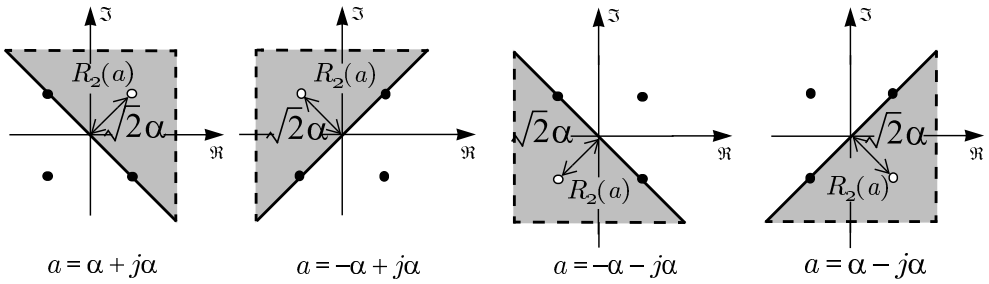

Figure 6.1: List detector decision regions of a 4-QAM list detector when $\ell = 2$.

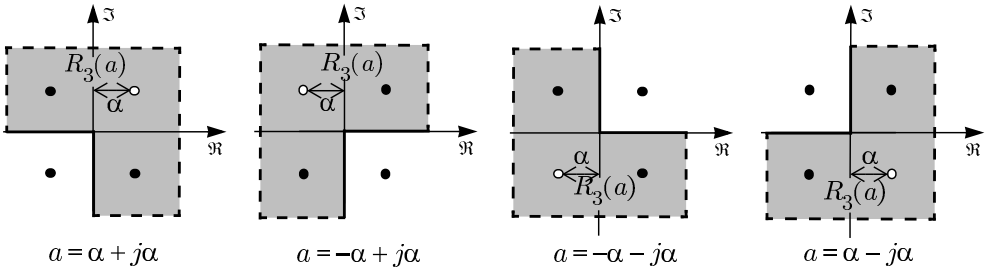

Figure 6.2: List detector decision regions of a 4-QAM list detector when $\ell = 3$.

A *list detection error* occurs whenever the actual transmitted symbol does not appear on the list produced by the list detector. In other words, a list detection error occurs if a is transmitted and $a \notin \mathcal{L}$, which implies that a is further from the received signal than all symbols on the list; that is [17]:

$$|r - \lambda_l| \leq |r - a|, \qquad l = 1,\ 2,\ \ldots\ \ell. \tag{6.4}$$

The performance of the B-Chase(ℓ) MIMO detector is largely determined by the probability of a list detection error, which we denote P_ℓ [98].

Let us associate with each transmitted symbol $a \in \mathcal{A}$ and for each list length $\ell \in \{1, \ldots q\}$ a list detector decision region $R_\ell(a)$, defined as the set of complex numbers r that would result in the length-ℓ decision list $\mathcal{L}$ containing the symbol a [96], i.e. $R_\ell(a) = \{r : a \in \mathcal{L}\}$. In the special case when $\ell = 1$, the list detector decision regions reduce to the familiar minimum-distance Voronoi cells, i.e. $R_1(a) = \{r : |r - a| < |r - x|, \forall x \in \mathcal{A}, x \neq a\}$ [37]; which are mutually disjoint. In the general case when $\ell > 1$, the list detector decision regions for different symbols can overlap.

Fig.6.1 depicts the 4-QAM list detector decision regions for each element of the alphabet $\mathcal{A}$ when $\ell = 2$. Fig.6.2 depicts the 4-QAM list detector decision regions for each element of the alphabet $\mathcal{A}$ when $\ell = 3$. Assuming all symbols are equally likely and exploiting the symmetry of the 4-QAM alphabet, P_2 can be expressed as:

$$P_2 = Q(\sqrt{2}\alpha/\sigma) = Q(\sqrt{2SNR}). \tag{6.5}$$

Furthermore, P_3 can be expressed as:

$$\begin{aligned} P_3 &= P(\Re\{n\} < -\alpha)P(\Im\{n\} < -\alpha) \\ &= Q^2(\alpha/\sigma) = Q^2(\sqrt{SNR}). \end{aligned} \tag{6.6}$$

In particular we see that $P_3 < P_2$. It is not hard to show that, in general, P_ℓ will be a decreasing function of ℓ. In the extreme case when the list length is maximal ($\ell = q$), the decision list reduces to the entire alphabet, and therefore the list detection error probability will be zero ($P_q = 0$).

In general, an exact expression for the list detection error probability in AWGN with q-ary QAM can be found using:

$$\begin{aligned} P_\ell &= 1 - \frac{1}{q}\sum_{1\le m\le q} P(a_m \text{ on decision list} \mid a_m \text{ transmitted}) \\ &= 1 - \frac{1}{q}\sum_{1\le m\le q} P(a_m + n \in R_\ell(a_m)) \\ &= 1 - \frac{1}{2\pi\sigma^2 q}\sum_{1\le m\le q}\int_{R_\ell(a_m)}\int e^{\frac{-1}{2\sigma^2}|r-a_m|^2} dr. \end{aligned} \tag{6.7}$$

Unfortunately, the intricate shape of the list detection decision regions for most alphabets and most list lengths $\ell > 1$ will make this integral intractable.

6.3.2 Minimum Distance Approximation

Since computing (6.7) is usually intractable, we introduce a *minimum-distance approximation* to upper bound the list detection error probability. Specifically, for each $a \in \mathcal{A}$, we will approximate the list detector decision region $R_\ell(a)$ by a circular disc $D_\ell(a) \subset R_\ell(a)$

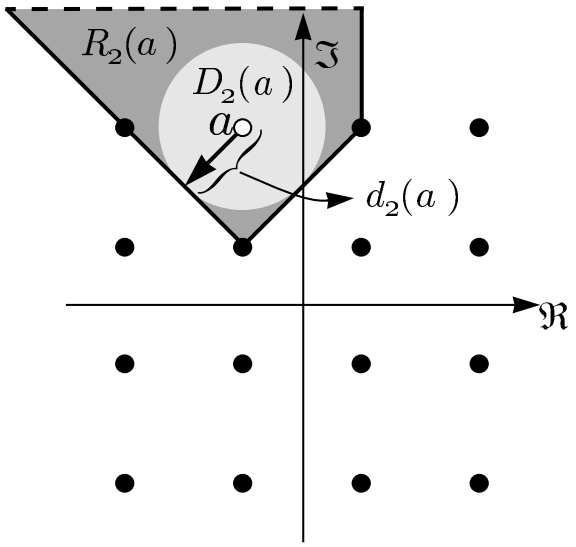

Figure 6.3: The exact list detection decision region $R_2(a)$ for list length $\ell = 2$ and 16-QAM is a semi-infinite polygon. The approximate list detection decision region $D_2(a)$ is the circular disc centered at a with radius $d_2(a)$.

that is centered at a. The radius $d_\ell(a)$ of this disc will be as large as possible, subject to the constraint that the disc be a subset of the list detector decision region, i.e., subject to the constraint $D_\ell(a) \subset R_\ell(a)$. The requirement that $d_\ell(a)$ be as large as possible ensures that our bound is as tight as possible. We may thus interpret $d_\ell(a)$ as the minimum distance between the symbol a and the nearest boundary of its length-ℓ list detector decision region.

Fig.6.3 shows the list detection decision region $R_2(a)$ and the corresponding disc $D_2(a)$ for a 16-QAM constellation when $a = \alpha(1 + 3j)$. Here, $D_2(a)$ is the lightly shaded disc centered at a. Its radius $d_2(a)$ causes the edge of the disc to touch but not cross the boundary of the list detection decision region $R_2(a)$.

Because $D_L(a) \subset R_L(a)$, replacing $R_L(a)$ by $D_L(a)$ in (6.7) results in the following upper bound for the list detection error probability in AWGN:

$$P_L = 1 - \frac{1}{q} \sum_{1 \le m \le q} P(a_m + n \in R_\ell(a_m)) \tag{6.8}$$

$$\le 1 - \frac{1}{q} \sum_{1 \le m \le q} P(a_m + n \in D_\ell(a_m)) \tag{6.9}$$

$$= 1 - \frac{1}{q} \sum_{1 \le m \le q} P(|n|^2 \le {d_\ell}^2(a_m)) \tag{6.10}$$

ℓ	1	2, 3	4	5	6	7, 8	9, 10, 11	12	13	14, 15	16
$d_{\ell,\min}$	α	$\sqrt{2}\alpha$	2α	$\sqrt{5}\alpha$	2.5α	$\sqrt{8}\alpha$	$\sqrt{10}\alpha$	$\sqrt{338/25}\alpha$	$\sqrt{130/9}\alpha$	$\sqrt{18}\alpha$	∞

Table 6.1: Global Minimum Distances for 16-QAM.

$$= \frac{1}{q} \sum_{1 \le m \le q} e^{-{d_\ell}^2(a_m)/2\sigma^2}, \tag{6.11}$$

where the last equality follows from the fact that $|n|^2$ is an exponential random variable with mean $2\sigma^2$. A looser bound results when we replace each $d_\ell(a)$ by the global minimum $d_{\ell,\min} = \min\{d_\ell(a) : a \in \mathcal{A}\}$, yielding:

$$P_\ell \le e^{-{d_\ell}^2(a_m)/2\sigma^2}. \tag{6.12}$$

We will refer to this bound as the *minimum-distance approximation* for the list detection error probability.

The global minimum distances $d_{\ell,\min}$ to the list detection decision region for a 4-QAM constellation for $\ell \in \{1, 2, 3, 4\}$ are $d_{1,\min} = \alpha$, $d_{2,\min} = \sqrt{2}\alpha$, $d_{3,\min} = \sqrt{2}\alpha$ and $d_{4,\min} = \infty$, respectively [96]. From these values we can see that the minimum distance approximation does not always yield a tight upper bound. For example, the minimum distance approximation for P_3 in AWGN with 4-QAM is $P_3 \le e^{SNR}$, which differs by more than 1 dB from the exact result $Q^2(\sqrt{SNR})$ of (6.6). Table 1 tabulates the global minimum distance for each possible list length when the alphabet is 16-QAM.

The list detection decision regions for QAM alphabets larger than 4-QAM are more intricate than those in Fig.6.1. For example, Fig.6.4 shows the list detection decision regions for 16-QAM and $\ell = 3$ for the three unique symbol types: (a) corner, (b) inner, and (c) edge. The associated symbol is indicated by an open circle in the figure. Fig.6.5 shows how these 16-QAM list detection decision regions change when the list length is increased from $\ell = 3$ to $\ell = 7$.

Fig.6.6 compares the actual list detection-error probability to the minimum-distance

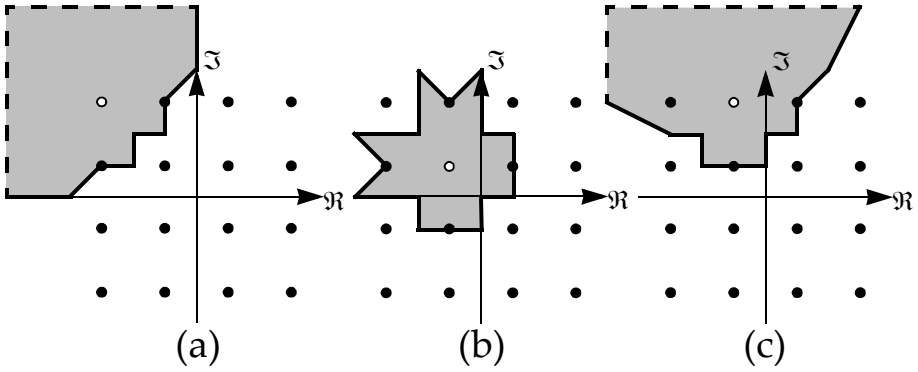

Figure 6.4: List detection decision regions for 16-QAM with $\ell = 3$ for (a) corner point; (b) inner point; and (c) edge point.

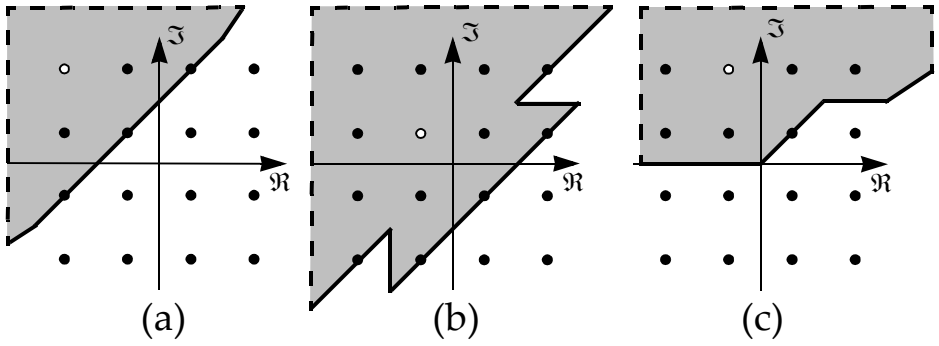

Figure 6.5: List detection decision regions for 16-QAM with $\ell = 3$ for (a) corner point; (b) inner point; and (c) edge point.

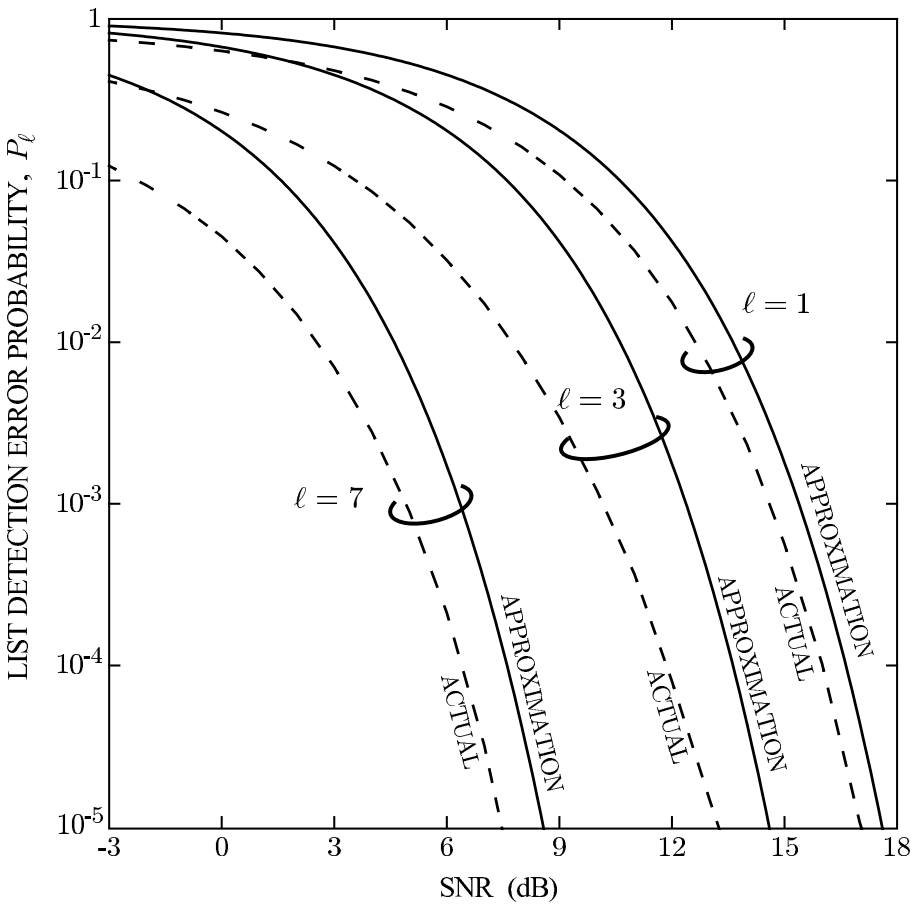

Figure 6.6: Comparing the actual list detection error probability to the minimum-distance approximation for 16-QAM and AWGN, for list lengths $\ell = 1$, $\ell = 3$ and $\ell = 7$.

approximation (6.12) for 16-QAM and AWGN, for $\ell = 1$, $\ell = 3$, and $\ell = 7$. The minimum-distance approximation results in an error of 0.6 dB at 10^{-4} for $\ell = 1$, 1.3 dB for $\ell = 3$, and 1.1 dB when $\ell = 7$.

6.4 Nonuniform Computational Complexity Allocation for OFDM

A conventional OFDM detector implements the same detection algorithm on each subcarrier so that the computation complexity is uniformly distributed across subcarriers [98]. A nonuniform computational complexity allocation (NCCA) OFDM detector implements a different detector on each subcarrier, depending on the subcarrier channel quality, resulting in a nonuniform distribution of computational complexity across subcarriers.

Given this framework we can think of an OFDM detector as having a total complexity budget of B complexity units to be distributed amongst J subcarriers. Let us define:

$$\ell_i = \{1, 2, \ldots q\} \tag{6.13}$$

as the list length assigned to the ith subcarrier detector. The lower bound of 1 and upper bound of q are imposed by the structure of the B-Chase detector, whose list length is limited to the range $\{1, 2, \ldots q\}$. A conventional OFDM detector assigns complexity uniformly, so that the list length is identical ($\ell_i = B/J$) for all subcarriers. We relax this constraint and allow a nonuniform complexity assignment, subject only to the constraints:

$$\begin{gathered} \ell_i \in \{1, 2, \ldots q\} \text{ for } i \in \{1, 2, \ldots J\}, \\ \sum_{1 \le i \le J} \ell_i = B. \end{gathered} \tag{6.14}$$

Note that the values that the complexity budget B might take are more constrained for the uniform case than for the nonuniform case. Specifically, a uniform assignment of $B/J \in \{1, 2, \ldots q\}$ units to each subcarrier would require that the complexity budget satisfy $B \in \{J, 2J, 3J, \ldots qJ\}$. In contrast, a nonuniform yet integer-valued complexity assignment can be made whenever $B \in \{J, J+1, J+2, \ldots qJ\}$. Consequently, a nonuniform complexity assignment is beneficial to a system designer, in contrast to a uniform assignment, because it allows for more freedom in the selection of the overall complexity budget.

There are potentially many ways to select $\{\ell_i\}$ yielding performance superior to that of conventional uniform assignment. We choose to adopt a strategy which allocates more of the complexity budget to the subcarriers which need the most help, in terms of the list detection error probability. Specifically, after initialization, we propose to allocate complexity units one at a time to the subcarrier with the highest list detection error probability.

In Fig.6.7 we present the algorithm `allocate`, which summarizes our proposed strategy for distributing computational resources to the different subcarriers. The inputs for the algorithm are the complexity budget B and the channel matrices $\{\mathbf{H}^{(1)}, \ldots \mathbf{H}^{(J)}\}$, while the outputs are the list lengths $\{\ell_1, \ldots \ell_J\}$ for the different subcarriers.

Input: $B, \{\mathbf{H}^{(1)}, \ldots \mathbf{H}^{(J)}\}$
Output: $\{\ell_1, \ldots \ell_J\}$

Initialize $\ell_1 = \ell_2 = \ldots = \ell_J = 1$
for $i = 1 : J$ **do**
    $k_i = \texttt{BChaseSelect1}(\mathbf{H}^{(i)}, \ell_i)$
    $SNR_i = \|\mathbf{h}_{k_i} - \hat{\mathbf{h}}_{k_i}\|^2 E_a / N_0$
end
for $\texttt{count} = 1 : B - J$ **do**
    $i = \arg\max P_{\ell_i}(SNR_i)$
    $\ell_i = \ell_i + 1$
    $k_i = \texttt{BChaseSelect1}(\mathbf{H}^{(i)}, \ell_i)$
    $SNR_i = \|\mathbf{h}_{k_i} - \hat{\mathbf{h}}_{k_i}\|^2 E_a / N_0$
end

Figure 6.7: An algorithm for allocating complexity with budget B amongst J OFDM subcarriers.

The `allocate` function begins by initializing each list length to one, so that the remaining complexity budget is $B - J$. It then computes the instantaneous SNR values for each tone. The `allocate` algorithm then loops $B - J$ times, each time increasing the list length on the subcarrier with the maximum list detection error probability. The algorithm terminates once the complexity budget has been exhausted. Note that, with our assumption that $B \leq qJ$, the constraints (6.14) will always be satisfied. In practice one might cap each list length at the value needed in order for the corresponding list detection error probability to achieve a desired target error probability, thereby reducing power consumption.

A key parameter within the `allocate` algorithm is the instantaneous SNR at the input to the B-Chase list detector on subcarrier i, denoted SNR_i. For the ZF case, this SNR can be written as $SNR_i = \|\mathbf{h}_{k_i} - \hat{\mathbf{h}}_{k_i}\|^2 E_s / N_0$, where $k_i \in \{1, \ldots N\}$ denotes the index of the first symbol detected by B-Chase(ℓ_i) on subcarrier i, where h_{k_i} denotes the k_ith column of $\mathbf{H}^{(i)}$, and where $\mathbf{h}_{k_i}$ denotes the projection of the k_ith column of $\mathbf{H}^{(i)}$ onto the subspace spanned by the remaining columns of $\mathbf{H}^{(i)}$. Each time a list length changes, the corresponding index k_i may change, in which case the corresponding SNR_i must be updated. The `BChaseSelect1`

function, which takes as inputs a MIMO channel and an allocated list length L_i, produces the index k_i using selection algorithm #1 of [98]. Because all list lengths are initialized to 1, the index k_i found during the first for loop is the same as that found using the V-BLAST ordering [100].

Although the above-described `allocate` algorithm is based on the ZF version of the B-Chase detector, the same algorithm can be used for the MMSE case as well, with only two modifications: First, the inputs to the `allocate` algorithm should be the extended channel matrices $\{\underline{\mathbf{H}}^{(i)}\}$ instead of $\{\mathbf{H}^{(i)}\}$, where [45, 14]:

$$\underline{\mathbf{H}}^{(i)} = \begin{bmatrix} \mathbf{H}^{(i)} \\ \sqrt{\frac{N_0}{E_s}}\mathbf{I} \end{bmatrix}, \tag{6.15}$$

where $\mathbf{I}$ is the $N \times N$ identity matrix. Second, the SNR computation in lines 4 and 11 should be changed to $SNR_i = \|\mathbf{h}_{k_i} - \hat{\mathbf{h}}_{k_i}\|^2 Es/N_0 - 1$, to account for the MMSE bias.

6.5 Numerical Results

We now present numerical results quantifying the performance of the proposed NCCA detector using B-Chase detectors and the `allocate` algorithm over a MIMO-OFDM Rayleigh-fading channel with 4 inputs and 4 outputs. We adopt the typical urban channel model presented in Table 5.1 of 3GPP TR 25.943 [1], which we approximated using a 16-tap model, assuming a sample rate of 5.128 MHz at the D/A and A/D converters. We assume that 64 subcarriers are used, where 48 subcarriers contain data and 16 subcarriers are zero. No bit loading is performed and each subcarrier carries the same QAM alphabet. Finally, in the case of MMSE detection, all branch metrics computed during the tree search are unbiased and the final decisions are based on the ZF minimum distance cost.

Fig.6.8(a) and Fig.6.8(b) illustrate the instantaneous SNR values as a function of frequency for two typical channel realizations. The SNR_i values shown are determined at the end of the `allocate` algorithm. Fig.6.8(e) and Fig.6.8(f) illustrate the corresponding complexity allocations $\{\ell_i\}$ of `allocate`, assuming $B = 144$ and 16-QAM. Observe

that the high-SNR subcarriers are assigned short list lengths, while the low-SNR subcarriers are assigned long list lengths. This aspect is highlighted in Fig.6.8(c) and Fig.6.8(d), which illustrate the inverse values of SNR_i as determined at the end of the `allocate` algorithm. The strong relationship between SNR_i^{-1} and $\{\ell_i\}$ implies that a lower complexity `allocate` algorithm may be possible by simply scaling SNR_i^{-1} and quantizing using a given complexity budget.

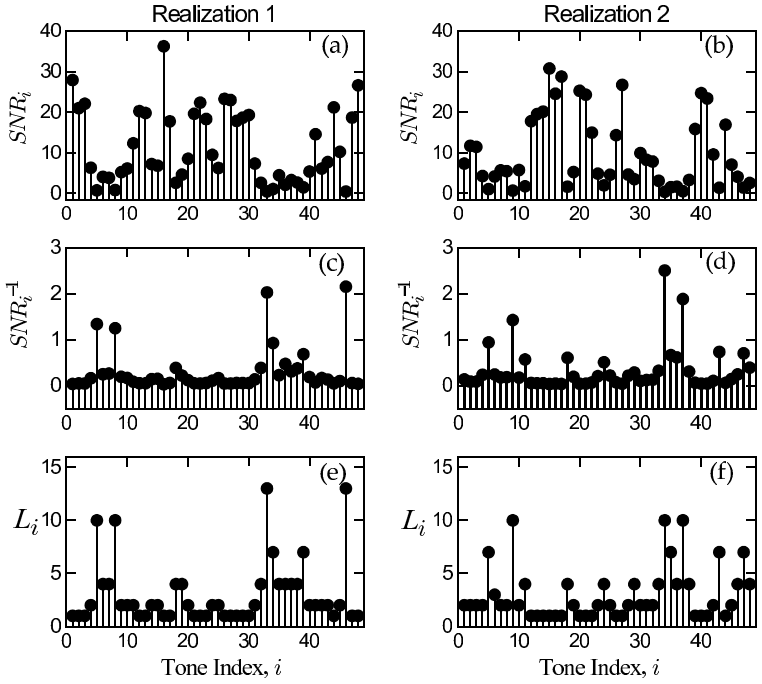

Figure 6.8: The instantaneous SNR values for two typical channels (a)-(b), their inverse values (c)-(d), and the corresponding list-length allocation (e)-(f) of `allocate`, with an average SNR of 8 dB, 16-QAM alphabet, and $B = 144$.

In Fig.6.9(a) we illustrate the average ordered SNR values (and Fig.6.9(b) their inverse values); these results were found by sorting the instantaneous SNR values $\{SNR_i\}$ in increasing order for each of 10^4 independent Rayleigh-fading channel realizations, and then averaging the ordered results. The corresponding mean value of ℓ_i on each subcarrier, assuming a 16-QAM alphabet is also shown for four different budgets in (c) $B = 96$, (d) $B = 192$, (e) $B = 288$, and (f) $B = 384$. Note that as the SNR value for a subcarrier increases, the complexity assigned to that subcarrier decreases.

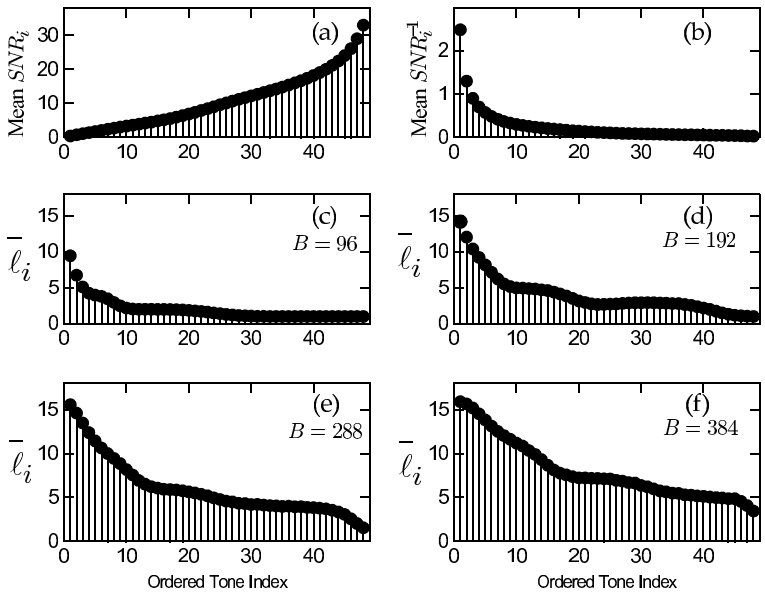

Figure 6.9: The average SNR (a) and inverse SNR (b) after ordering, averaged over 10^4 channel realizations, and the corresponding average list lengths for a 16-QAM alphabet with (c) $B = 96$, (d) $B = 192$, (E) $B = 288$, and (f) $B = 384$.

In Fig.6.10 we compare the performances of several MIMO-OFDM detectors for the same 4-input 4-output Rayleigh-fading channel with 16-QAM inputs. The upper two curves show the performance of conventional OFDM detectors based on ZF and MMSE versions of the B-Chase(2) algorithm, where the computational complexity is distributed uniformly across subcarriers ($\ell_i = 2$ for all i). The group of four curves labeled "NCCA" shows the performance of four variations of the proposed NCCA detector, all of which have the same complexity budget as the conventional detectors ($B = 96$), but distribute computational resources nonuniformly across the subcarriers. All four of the NCCA detectors use B-Chase(ℓ_i) on subcarrier i, where the list lengths are allocated using the `allocate` algorithm. The two black NCCA curves are for ZF and MMSE, assuming that the list detection error probabilities within the `allocate` algorithm were computed from a table of precomputed values designed to emulate the exact list detection error probability. The two grey NCCA curves are for ZF and MMSE, assuming that the list detection error probabilities within the `allocate` algorithm were approximated by the minimum-distance approximation (6.12).

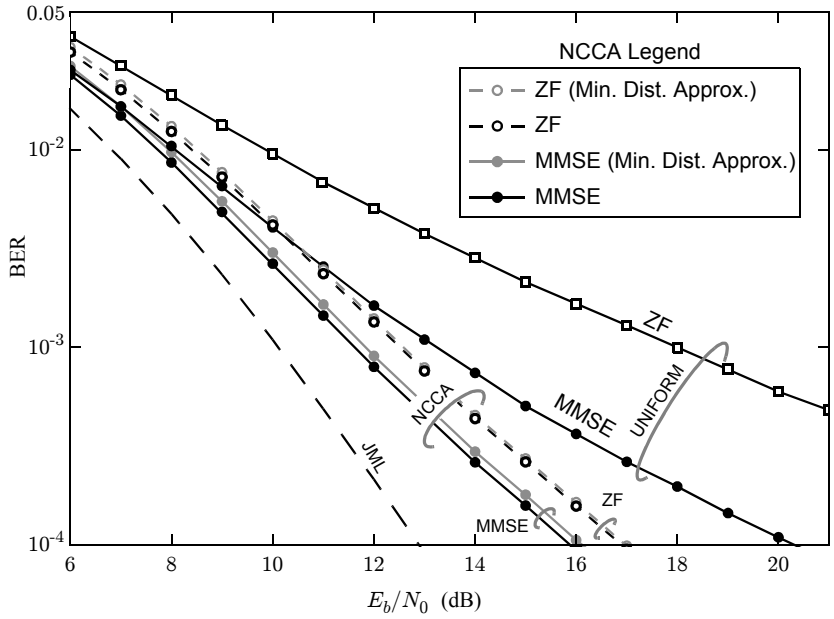

Figure 6.10: Error probability performance for B-Chase over a 4-input 4-output frequency-selective Rayleigh-fading channel with 16-QAM on each OFDM subcarrier. The complexity budget is $B = 96$ for both the uniform (conventional) and nonuniform (proposed NCCA) receivers.

From these results we can see that the NCCA MMSE B-Chase detector outperforms the conventional OFDM MMSE B-Chase detector by 4.4 dB at 10^{-4}, with a gap of 3.0 dB to the joint maximum likelihood (JML) detector [26]. The NCCA ZF B-Chase detector outperforms the conventional OFDM ZF B-Chase detector by 5.5 dB at 10^{-3}, with a gap of 2.4 dB to JML performance. The minimum distance approximation incurs a penalty of 0.2 dB for the MMSE case and 0.1 dB the ZF case at 10^{-4}.

In Fig.6.11 we illustrate the performance-complexity trade-off for several MIMO-OFDM detectors by plotting the required SNR per bit versus the complexity budget B. These results are based on a 4-input 4-output Rayleigh-fading channel with 64-QAM inputs. The two upper curves show the performance of conventional OFDM detectors based on ZF and MMSE versions of the B-Chase(ℓ), for $\ell \in \{1, \ldots 12\}$. The two curves marked by circles show the performance of the proposed NCCA approach. The performance of the

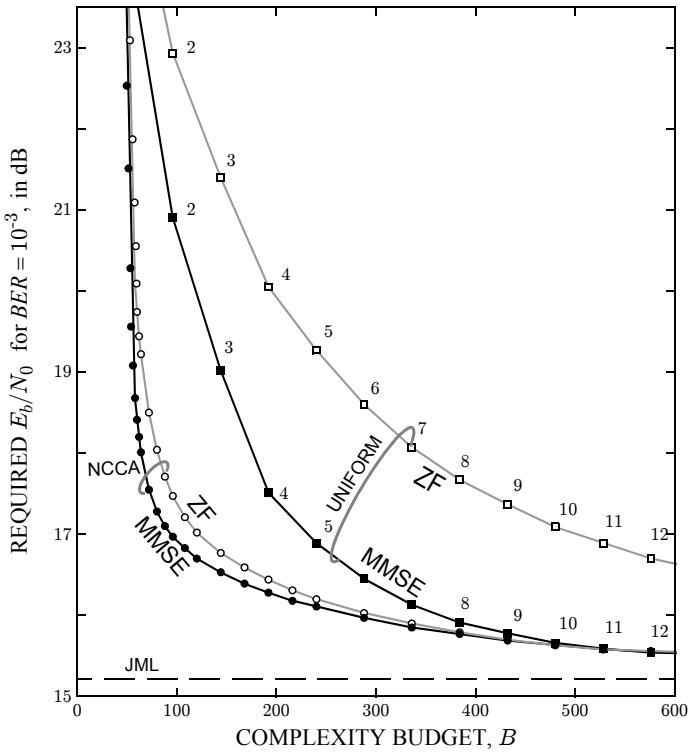

Figure 6.11: Performance-complexity trade-off for B-Chase over a 4-input 4-output frequency-selective Rayleigh fading channel with 64-QAM on each OFDM subcarrier for various complexity budgets.

JML detector is given as reference.

The vertical distance between curves measures the SNR advantage at a fixed complexity. From Fig.6.11 we can see that the NCCA detector outperforms the conventional detector at all complexity budgets. For example, when $B = 96$, the NCCA detector with MMSE B-Chase outperforms the conventional MMSE B-Chase detector by 3.9 dB, with a gap of 1.8 dB to JML performance. When $B = 192$, the performance gain is 1.2 dB, with a gap of 1.1 dB to JML performance. The horizontal distance between the curves measures the complexity advantage at a fixed SNR requirement. From Fig.6.11 we see the NCCA approach delivers significant complexity reduction. For example, the NCCA detector with MMSE B-Chase and $B = 56$ achieves the same performance as the conventional detector (with $B = 144$) with less than 40% of the complexity.

Fig.6.11 shows that the gap between ZF and MMSE is significantly smaller with NCCA than with the conventional detector.

6.6 Conclusion

In this chapter we proposed a distributed complexity allocation strategy for OFDM detection over frequency-selective channels. Typically MIMO-OFDM receivers will implement the same detector at each subcarrier and, when these detectors are fixed complexity detectors, this represents uniform computational complexity allocation. When the same detector is implemented on each subcarrier the overall error-rate performance is dominated by the weakest subcarrier. Improved error detection is achieved by exploiting channel state information to allocate a given complexity budget nonuniformly across the different subcarriers. We proposed to allocate computational complexity nonuniformly at the receiver, an idea we named nonuniform computational complexity allocation. Our NCCA algorithm used a metric known as the list error probability, which is the probability that the list of the ℓ symbols closest to the channel output does not contain the transmitted symbol, to assign computational resources. We computed an exact expression for the list error probability in AWGN, and provided a minimum distance approximation that serves as an upper bound for the list error probability. Numerical results for Rayleigh fading channels show that the proposed nonuniform MIMO-OFDM detector outperformed a conventional detector with the same complexity by 4.4 dB.

CHAPTER 7

CONCLUSIONS

7.1 Summary

In this dissertation, we investigated the use of candidate lists to solve the soft-output MIMO detection problem. We focused on solutions with low and fixed computational complexity. The key findings and conclusions for this dissertation are summarized as follows:

- List detection is an attractive solution to the soft-output MIMO detection problem.
- The process of list detection can be constructed as that of a search through a detection tree.
- The goal of list detectors employing a detection tree can be phrased as finding the ℓ leaf nodes in the tree with minimum cost.
- Near max-log optimal performance can be achieved with only a small subset from the set of all leaf nodes in a detection tree.
- Using the number of branch metric computations performed during the search of the detection tree is a suitable measure of the computational complexity for tree-based list MIMO detectors.
- The number of branch metric computations as a measure of computational complexity leads to bounds on the computational complexity of tree-based list detectors.
- A lower bound on the computational complexity of tree-based list MIMO detection can be derived by considering a *genie tree search detector*: for a given list size $\ell = |\mathcal{L}|$, this detector generates only the subset of leaf nodes from $\mathcal{L}$ which are actually used in the calculation of the LLRs based on the max-log approximation.

- Soft-output detection algorithms with low and fixed computational complexity achieving near max-log optimal performance are feasible.

- Breadth-first algorithms were shown to be a viable approach for achieving a desirable performance-complexity tradeoff when using tree-based detection. In particular, the fixed computational complexity of breadth-first detection was shown to be advantageous to the overall performance-complexity profile, rather than a detriment to the system's performance.

- The use of smart candidate adding techniques that only visit nodes in the detection tree once (i.e. parallel search) were shown to be highly efficient at solving the soft-output MIMO detection problem.

- The ordering of the MIMO channel matrix is an important consideration when designing MIMO detection algorithms. We examined the significant impact of the detection ordering on fixed computational complexity soft-output detection algorithms.

- We presented a smart ordering and candidate algorithm (SOCA) algorithm that achieves a desirable performance-complexity profile in both slow and fast fading scenarios. The SOCA algorithm achieves its desirable properties by using parallel smart candidate adding and careful selection of the preprocessing algorithm.

- A framework for characterizing algorithms that only visit nodes in the detection tree once, possess fixed computational complexity, and are breadth-first was presented. This framework allows for the classification of many prior detection algorithms so that their performance-complexity profiles can be compared. Additionally, the framework enables the construction of new breadth-first detection algorithms through simple parameterization.

- Weakly encoded MIMO systems subject to slow fading conditions require more computational complexity and higher signal-to-noise ratios to achieve the performance of

fast fading MIMO systems employing strong encoding algorithms. The use of space-time codes can increase the diversity gain over MIMO systems employing spatial multiplexing, but at the cost of increasing the effective channel dimensions.

- Soft-output detection of the golden code is an important but computationally difficult task. We proposed a low- and fixed- complexity soft-output detector for the golden code that used linear equalization to simplify the task of finding a list of candidate values for one pair of information symbols, and then – for each pair on the list – it uses decision-feedback equalization to find candidate values for the remaining pair of information symbols.

- Selection of the LLR clipping level for soft-output MIMO detection can have a significant impact of the performance of MIMO receivers. This is particularly true for suboptimal breadth-first detection algorithms.

- An LLR clipping algorithm was presented that determines the LLR clipping level based on the instantaneous SNR and list length employed for suboptimal detection algorithms.

- For the same computational complexity budget and receiver algorithm, MIMO-OFDM receivers that allocate computational complexity resources nonuniformly achieve lower error rates than those that uniformly allocation computational complexity.

- An effective metric for allocating computational complexity in a MIMO-OFDM receiver is the list error probability, i.e. the probability that the list of the ℓ symbols closest to the channel output does not contain the transmitted symbol. We computed an exact expression for the list detection error probability in AWGN, and provided a minimum distance approximation that serves as an upper bound on the list error probability.

- Our proposed nonuniform-computational complexity allocation (NCCA) algorithm, based on assigning the computational complexity to subcarriers with the highest list error probability first was shown to be effective in reducing the overall system error rate performance.

7.2 Future Work and Final Remarks

This dissertation focused on the problem of soft-output multiantenna detection with an emphasis on tree-based solutions that find a set of leaf nodes in the tree. In particular the proposed solutions all possessed low and fixed computational complexity.

Throughout this dissertation our solutions were presented in the context of a single user MIMO system. Extending this work to multi-user and/or cooperative MIMO systems represents a promising avenue of future work. Additionally, investigations into the combination of spatial multiplexing with antenna beamforming appears to be a feasible, but as yet relatively young area of study. Due to the difficulty of the detection problem for such a system, low complexity soft-output detection algorithms are of great importance.

The SOCA algorithm from chapter 3 was shown to be a promising soft-output detection algorithm with low and fixed computational complexity for both slow and fast fading scenarios. Architectural implementations of parallel smart candidate adding approaches, like the SOCA algorithm, are therefore an area of practical significance. This development work would add weight to the claim that the SOCA algorithm lends itself to a parallel architecture with low latency.

In chapter 4 we proposed a soft-output detection algorithm for the golden code. Extending this algorithm to work with space-time codes similar to the golden code would be a worthwhile investigation. Examples of similar space-time codes of interest include the embedded Alamouti space-time codes [83] and the more general embedded orthogonal space-time codes [84].

Chapter 5 proposed an approach for SNR-aware LLR clipping, albeit without proof of

optimality. In [112] the optimality of the approach used in this dissertation was demonstrated for a single-input single-output system employing BPSK modulation with a repetition code transmitting over an AWGN channel. An open problem is extending [112] to higher modulation schemes, more advanced codes, fading channels and/or multiantenna channels. Additionally, our use of the list length ℓ as a parameter for the SNR-aware clipping level is intuitive, but by no means optimal. Consequently, the optimality of SNR-aware LLR clipping remains an open problem.

Chapter 6 proposed to nonuniformly allocation computational complexity resources within a hard-output MIMO-OFDM list detector based on the list error probability. An obvious area where our work on nonuniform computational complexity can be extended is to the case of soft-output MIMO detection, rather than simply hard-output detection. Here the list error probability may not be as effective a metric for assigning computational complexity. Consequently, a second area where the work in chapter 6 can be extended is to consider metrics other than the list error probability to assign computational complexity.

The use of soft-output multiantenna detection has already found its way into real-world systems such as IEEE 802.11n [48] and WiMax [49]. Future areas of implementation include 4G cellular networks and next generation metropolitan area networks. The unrelenting push for increased data rates and reliability, combined with the ever decreasing cost of hardware, ensures that MIMO technology will become a pervasive technology in future communication systems. As a direct consequence of this fact, low computational complexity algorithms solving the soft-output detection problem will remain an important area of study as these solutions move from research to practice. Additionally, as the number of antennas used in MIMO systems grows, as the cost of hardware and demand allows, the exponential nature of the optimal solution implies that the problem of MIMO detection will remain an important challenge.

REFERENCES

[1] 3rd Generation Partnership Project, "3GPP Technical Report 25.943, V. 6.00 (2004-12)," tech. rep., Technical Specification Group Radio Access Network, Deployment Aspects, Dec. 2004.

[2] Alamouti, S. M., "A simple transmit diversity technique for wireless communication," *IEEE Journal on Selected Areas in Communications*, vol. 6, pp. 1451–1458, Oct. 1998.

[3] Anderson, J. and Mohan, S., "Sequential Coding Algorithms: A Survey and Cost Analysis," *IEEE Transactions on Communications*, vol. 32, pp. 169–176, Feb. 1984.

[4] Arora, A., Krunz, M., and Muqattash, A., "Directional medium access protocol (DMAP) with power control for wireless ad hoc networks," in *IEEE 2004 Global Communications Conference, (Globecom 2004)*, Nov. 2004.

[5] Bahai, A. and Saltzberg, B., *Multicarrier Digital Communications: Theory and Applications of OFDM*. Kluwer Academic, New York, 1999.

[6] Barbero, L. G., Ratnarajah, T., and Cowan, C., "A low-complexity soft-mimo detector based on the fixed-complexity sphere decoder," in *IEEE International Conference on acoustics, speech and signal processing (ICASSP '08), to appear*, (Las Vegas, USA), 30.-4. Mar./Apr. 2006.

[7] Barbero, L. G. and Thompson, J. S., "A Fixed-Complexity MIMO Detector Based on the Complex Sphere Decoder," in *IEEE Workshop on Signal Processing Advances for Wireless Communications (SPAWC'06)*, (Cannes, France), July 2006.

[8] Barbero, L. G. and Thompson, J. S., "Extending a Fixed-Complexity Sphere Decoder to Obtain Likelihood Information for Turbo-MIMO Systems," *IEEE Transactions on Vehicular Technology*, pp. 2804–2814, Sept. 2008.

[9] Barbero, L. G. and Thompson, J. S., "Fixing the Complexity of the Sphere Decoder for MIMO Detection," *IEEE Transactions on Wireless Communications*, vol. 7, pp. 2131–2142, June 2008.

[10] Bäro, S., "Turbo Detection for MIMO Systems Using a List Sequential Detector: Improved Soft Output by Path Augmentation," in *Proceedings of the ITG Conference on Source and Channel Coding (SCC'03)*, 2003.

[11] Bäro, S., Hagenauer, J., and Witzke, M., "Iterative detection of MIMO transmission using a list-sequential (LISS) detector," in *IEEE International Conference on Communications (ICC'03)*, vol. 4, pp. 2653 – 2657, Mar. 2003.

[12] Barry, J. R., Lee, E. A., and Messerschmitt, D. G., *Digital Communication*. Boston: Kluwer Academic Publishers, third ed., 2004.

[13] Belfiore, J.-C., Rekaya, G., and Viterbo, E., "The Golden Code: A 2x2 full rate Space-Time Code with Non Vanishing Determinants," *IEEE Transactions on Information Theory*, vol. 51, 2005.

[14] Böhnke, R., Wübben, D., Kühn, V., and Kammeyer, K., "Reduced complexity MMSE detection for BLAST architectures," in *IEEE Global Communications Conference*, vol. 4, pp. 2258–2262, Dec. 2003.

[15] Bittner, S., Zimmermann, E., and Fettweis, G., "Low Complexity Soft Interference Cancellation for MIMO-Systems," in *Proceedings of the IEEE Vehicular Technology Conference (VTC'06)*, (Melbourne, Australia), May 2006.

[16] Bittner, S., Zimmermann, E., Rave, W., and Fettweis, G., "List sequential MIMO detection: Noise bias term and partial path augmentation," in *IEEE International Conference on Communicaitons (ICC'06)*, (Istanbul, Turkey), 11.-15 June 2006.

[17] Bocharova, I. E., Johannesson, R., Kudryashov, B. D., and Loncar, M., "An Improved Bound on the List Error Probability and List Distance Properties," *IEEE Transactions on Information Theory*, vol. 54, pp. 13–32, Jan. 2008.

[18] Burg, A., Borgmann, M., Simon, C., Wenk, M., Zellweger, M., and Fichtner, W., "Performance tradeoffs in the VLSI implementation of the sphere decoding algorithm," in *Fifth IEEE International Conference on 3G Mobile Communication Technologies*, pp. 93–97, Oct. 2004.

[19] Burg, A., Borgmann, M., Wenk, M., Zellweger, M., Fichtner, W., and Bölcskei, H., "VLSI Implementation of MIMO Detection using the Sphere Decoding Algorithm," *IEEE Journal of Solid-State Cicuits*, vol. 40, July 2005.

[20] Butler, M. and Collings, I., "A zero-forcing approximate log-likelihood receiver for MIMO bit-interleaved coded modulation," vol. 8, pp. 105–107, Feb. 2004.

[21] Caire, G., Taricco, G., and Biglieri, E., "Bit-interleaved coded modulation," *IEEE Transactions on Information Theory*, vol. 44, pp. 927 – 946, May 1998.

[22] Chen, S. and Zhang, T., "Low Power Soft-Output Signal Detector Design for Wireless MIMO Communication Systems," in *International Symposium on Low Power Electronics and Design (ISLPED'07)*, Aug. 2007.

[23] Chin, W. H., "QRD Based Tree Search Data Detection for MIMO Communication Systems," in *Proceedings of the IEEE Vehicular Technology Conference*, vol. 3, (Stockholm, Sweden), pp. 1624–1627, May 2005.

[24] Choi, W. J., Cheong, K. W., and Cioffi, J. M., "Iterative Soft Interference Cancellation for Multiple Antenna Systems," in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNCŠ00)*, vol. 1, pp. 304–309, 2000.

[25] Cui, T. and Tellambura, C., "An efficient generalized sphere decoder for rank-deficient MIMO systems," *IEEE Communications Letters*, vol. 9, pp. 423–425, May 2005.

[26] Damen, O., Chkeif, A., and Belfiore, J.-C., "Lattice code decoder for space-time codes," *IEEE Communications Letters*, vol. 4, pp. 161–163, May 2000.

[27] Dayal, P. and Varanasi, M. K., "An Optimal Two Transmit Antenna Space-Time Code And Its Stacked Extensions," *IEEE Transactions on Information Theory*, vol. 51, pp. 4348–4355, Dec. 2005.

[28] de Jong, Y. L. C. and Willink, T. J., "Iterative tree search detection for MIMO wireless systems," *IEEE Transactions on Communications*, vol. 53, pp. 930–935, June 2005.

[29] Dejonghe, A. and Vandendorpe, L., "Turbo equalization for multilevel modulation: An eścient lowcomplexity scheme," in *Proceedings of the IEEE International Conference on Communications (ICC'02)*, vol. 3, (New York City, USA).

[30] Dong, B., Wang, X., and Doucet, A., "A new class of MIMO demodulation algorithms," *IEEE Transactions on Signal Processesing*, vol. 51, pp. 2752–2763, Nov. 2003.

[31] Farhang-Boroujeny, B., Haidong, Z., and Zhenning, S., "Markov chain Monte Carlo algorithms for CDMA and MIMO communication systems," *IEEE Transactions on Signal Processing*, vol. 54, pp. 1896–1909, May.

[32] Fincke, U. and Pohst, M., "Improved methods for calculating vectors of short length in lattice, including a complexity analysis," *Mathematics of Computation*, vol. 44, pp. 463–471, Apr. 1985.

[33] Foschini, G. J., "Layered Space-Time Architecture for Wireless Communication in a Fading Environment When Using Multi-Element Antennas," *Bell Laboratories Technical Jounral*, pp. 41–59, Oct. 1996.

[34] Foschini, G. J., Golden, G. D., Valenzuela, R. A., and Wolniansky, P. W., "Simplified processing for wireless communication at high spectral efficiency," *IEEE Journal on Selected Areas in Communications*, vol. 17, pp. 1841–1852, Nov. 1999.

[35] Garrett, D., Davis, L., ten Brink, S., Hochwald, B., and Knagge, G., "Silicon complexity for maximum likelihood MIMO detection using spherical decoding," *IEEE Journal of Solid-State Circuits*, vol. 39, pp. 1544–1552, Sept. 2004.

[36] Godara, L., "Applications of antenna arrays to mobile communications, Part II: Beam-forming and direction-of-arrival considerations," *Proceedings of the IEEE*, vol. 85, p. 1195Ű1245, Aug. 1997.

[37] Goldsmith, A., *Wireless Communication*. Cambridge University Press, 2005.

[38] Guo, D. and Wang, X., "Blind detection in MIMO systems via sequential Monte Carlo," *IEEE Journal on Selected Areas in Communications*, vol. 21, pp. 453–464, Apr. 2003.

[39] Guo, Z. and Nilsson, P., "Algorithm and implementation of the K-best sphere decoding for MIMO detection," *IEEE Journal on Selected Areas in Communications*, vol. 24, pp. 491–503, Mar. 2006.

[40] Hagenauer, J., "The turbo principle in mobile communications," in *Proceedings of the International Symposium on Information Theory and its Applications (ISITA'02)*, (Xi'an, China), 07.-11.Oct. 2002.

[41] Hagenauer, J. and Kuhn, C., "Turbo Equalization for Channels with High Memory using a List-Sequential (LISS) Equalizer," in *Proceedings of the International Symposium on Turbo Codes and Related Topics (ISTCS̆03)*, (Brest, France), Sept. 2003.

[42] Hagenauer, J. and Kuhn, C., "The List-Sequential (LISS) algorithm and its application," *IEEE Transactions on Communications*, vol. 55, May 2007.

[43] Hagenauer, J., Robertson, P., and Papke, L., "Iterative (turbo) decoding of systematic convolutional codes with the MAP and SOVA algorithms," in *Proc. ITG Conference on Source and Channel Coding (SCC'94)*, pp. 21 – 29, 1994.

[44] Hanzo, L., Woodard, J. P., and Robertson, P., "Turbo Decoding and Detection for Wireless Applications," *Proceedings of the IEEE*, vol. 95, pp. 1178 – 1200, June 2007.

[45] Hassibi, B., "An efficient square-soot algorithm for BLAST," in *IEEE Conference on Acoustics, Speech, and Signal Processing*, vol. 2, pp. 737–740, June 2000.

[46] Hermite, C., "Extraits de lettres à M. Jacobi sur différents objets de la théorie des nombres, (in French)," *J. Reine und Angewandte Math.*, vol. 40, p. 261Ű315, 1850.

[47] Hochwald, B. and ten Brink, S., "Achieving near-capacity on a multiple-antenna channel," *IEEE Transactions on Communications*, vol. 51, pp. 389–399, Mar. 2003.

[48] IEEE 802 Working Group 11, T. G. N., "IEEE P802.11n/D3.00, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Amendment 4: Enhancements for Higher Throughput.," tech. rep., Sept. 2007.

[49] IEEE 802 Working Group 16, T. G. E., "IEEE P802.16Rev2/D2, ŞDRAFT Standard for Local and metropolitan area networks,Ť Part 16: Air Interface for Broadband Wireless Access Systems," tech. rep., Dec. 2007.

[50] Jaldén, J. and Ottersten, B., "Parallel Implementation of a Soft Output Sphere Decoder," in *Conference Record of the 39th Asilomar Conference on Signals, Systems and Computers*, pp. 581–585, Oct. 2005.

[51] Jaldén, J., Barbero, L. G., Ottersten, B., and Thompson, J. S., "Full Diversity Detection in MIMO Systems with a Fixed-Complexity Sphere Decoder," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'07)*, (Honolulu, Hawaii, USA), Apr. 2007.

[52] Jeon, K., Kim, H., and Park, H., "An Efficient QRD-M Algorithm Using Partial Decision Feedback Detection," in *40th Asilomar Conference on Signals, Systems, and Computers*, Oct. 2006.

[53] Kim, B.-S. and Choi, K., "SNR Measurement Free Adaptive K-Best Algorithm for MIMO Systems," in *IEEE Wireless Communications and Networking Conference (WCNC'08)*, pp. 628–633, Apr. 2008.

[54] Kim, K. J., Yue, J., Iltis, R., and Gibson, J., "A QRD-M/Kalman filter-based detection and channel estimation algorithm for MIMO-OFDM systems," *IEEE Transactions on Wireless Communications*, vol. 4, p. 710Ű720, Mar. 2005.

[55] Knuth, D. E., *The Art of Computer Programming. Volume 3: Sorting and Searching*. Addison-Wesley Professional, 2 ed., 1998.

[56] Korkine, A. and Zolotareff, G., "Sur les formes quadratiques (in French)," *Math. Annalen*, vol. 6, p. 366Ű389, 1873.

[57] Lenstra, A. K., Lenstra, H. W., and Lovász, L., "Factoring Polynomials with Rational Coefficients," *Math. Ann.*, vol. 261, pp. 515–534, 1982.

[58] Li, Y. and Luo, Z.-Q., "Parallel detection for V-BLAST system," in *Proc. IEEE International Conference on Communications (ICC'02)*, vol. 1, (New York, NY USA), p. 340Ű344, Apr. 2002.

[59] Love, D. J., Hosur, S., Batra, A., and Jr., R. W. H., "Space-Time Chase Decoding," *IEEE Transactions on Wireless Communications*, vol. 4, pp. 2035–2039, Sept. 2005.

[60] Luo, J., Pattipati, K., P.Willett, and Levchuk, G., "Optimal grouping for a group decision feedback detector in synchronous CDMA communications," *IEEE Transactions on Communications*, vol. 51, pp. 341–346, Mar. 2003.

[61] Marsch, P., "On the complexity of efficient detection algorithms for multiple antenna systems," diploma thesis, TU Dresden, Nov. 2004.

[62] Marsch, P., Zimmermann, E., and Fettweis, G., "Smart Candidate Adding: A new Low-Complexity Approach towards Near-Capacity MIMO Detection," in *13th European Signal Processing Conference (EUSIPCO'05)*, (Antalya, Turkey), 04.-08. Sept. 2005.

[63] Mennenga, B. and Fettweis, G., "Search Sequence Determination for Tree Search based Detection Algorithms," in *IEEE Sarnoff Symposium*, Mar. 2009.

[64] Mennenga, B., Matus, E., and Fettweis, G., "Vectorization of the sphere detection algorithm," in *IEEE International Symposium on Circuits and Systems*, pp. 2806 – 2809, May 2009.

[65] Milliner, D. L. and Barry, J. R., "An Adaptive M-Algorithm for Detection of Multiple-Input Multiple-Output Channels," in *IEEE Signal Processing Advances in Wireless Communications, (SPAWC'07)*, (Helsinki, Finland), 17.-20. June 2007.

[66] Milliner, D. L., Sinnokrot, M. O., and Barry, J. R., "A Soft-Output Detector for the Golden Code," in *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC'09)*, Sept. 2009.

[67] Milliner, D. L., Zimmermann, E., Barry, J. R., and Fettweis, G., "A Fixed-Complexity Smart Candidate Adding Algorithm for Soft-Output MIMO Detection," *IEEE Journal of Selected Topics in Signal Processing*. submitted.

[68] Milliner, D. L., Zimmermann, E., Barry, J. R., and Fettweis, G., "A Framework for Fixed Complexity Single-Stage Breadth-First Detection," in *10th International Symposium on Spread Spectrum Techniques and Applications (ISSSTA'08)*, (Bologna, Italy), 25.-28. Aug. 2008.

[69] Milliner, D. L., Zimmermann, E., Barry, J. R., and Fettweis, G., "Computational Complexity Bounds for List MIMO Detection," in *The 11th International Symposium on Wireless Personal Multimedia Communications (WPMC08)*, (Lapland, Finland), 8.-11. Sept. 2008.

[70] Milliner, D. L., Zimmermann, E., Fettweis, G., and Barry, J. R., "Channel State Information Based LLR Clipping for List MIMO Detection," in *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC'08)*, 15-18. Sept. 2008.

[71] Mohan, S. and Anderson, J., "Computationally Optimal Metric-First Code Tree Search Algorithms," *IEEE Transactions on Communicaitons*, vol. 32, pp. 710–717, June 1984.

[72] Myllylä, M., Antikainen, J., Juntti, M., and Cavallaro, J. R., "The effect of LLR clipping to the complexity of list sphere detector algorithms," in *Forty-First Asilomar Conference on Signals, Systems and Computers*, pp. 1559–1563, 2007.

[73] Nekuii, M. and Davidson, T. N., "List-based soft demodulation of MIMO QPSK via semidefinite relaxation," in *IEEE 8th Workshop on Signal Processing Advances in Wireless Communications*, pp. 1–5, June 2007.

[74] Robertson, P., Villebrun, E., and Hoeher, P., "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," in *IEEE International Conference on Communications (ICC'95)*, vol. 2, (Seattle, USA), pp. 1009–1013, June 1995.

[75] S. Chen, T. Zhang, Y. X., "Breadth-first tree search MIMO signal detector design and VLSI implementation," in *IEEE MILCOM 2005*, vol. 3, pp. 1470–1476, Oct. 2005.

[76] Schnorr, C. P. and Euchner, M., "Lattice basis reduction: Improved practical algorithms and solving subset sum problems," in *Mathematical Programming*, vol. 66, pp. 181–199, Aug. 1994.

[77] Seethaler, D., Artés, H., and Hlawatsch, F., "Dynamic nulling-and-canceling for efficient near-ML decoding of MIMO systems," *IEEE Transactions on Signal Processing*, vol. 54, pp. 4741–4752, Dec. 2006.

[78] Seethaler, D., Matz, G., and Hlawatsch, F., "Efficient Soft Demodulation in MIMO-OFDM Systems with BICM and Constant Modulus Alphabets," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 4, May 2006.

[79] Seethaler, D., Matz, G., and Hlawatsch, F., "Low-complexity MIMO data detection using SeysenŠs lattice reduction algorithm," in *IEEE International Conference on Acoustics, Speech, and Signal Processing, (ICASSP '07)*, Apr. 2007.

[80] Shariat-Yazdi, R. and Kwasniewski, T., "A multi-mode sphere detector architecture for WLAN applications," in *IEEE International SOC Conference*, pp. 155–158, Sept. 2008.

[81] Shen, C., Fitz, M., and Siti, M., "Generalized Soft-Output Layered Orthogonal Lattice Detector for Golden Code," in *IEEE Wireless Communications and Networking Conference, (WCNC'07)*, pp. 525–529, Mar. 2007.

[82] Shin, W., Kim, H., Son, M., and Park, H., "An improved LLR computation for QRM-MLD in coded MIMO systems," in *IEEE Vehicular Technology Conference (VTC2007 Fall)*, (Baltimore, Maryland, USA), Sept. 2007.

[83] Sinnokrot, M., Barry, J. R., and Madisetti, V., "Embedded Alamouti Space-Time Codes for High Rate and Low Decoding Complexity," in *Asilomar Conference on Signals, Systems, and Computers, (Asilomar 2008)*, Oct. 2008.

[84] Sinnokrot, M., Barry, J. R., and Madisetti, V., "Embedded Orthogonal Space-Time Codes for High Rate and Low Decoding Complexity," in *IEEE Global Communications Conference (Globecom 2009)*, Nov. 2009.

[85] Sinnokrot, M. O. and Barry, J. R., "Fast Maximum-Likelihood Decoding of the Golden Code," *IEEE Transactions on Information Theory*, Nov. 2008. submitted.

[86] Steingrimsson, B., Luo, Z., and Wong, K., "Soft quasi-maximum likelihood detection for multipleantenna wireless channels," *IEEE Transactions on Signal Processing*, vol. 51, pp. 2710–2718, Nov. 2003.

[87] Studer, C., Burg, A., and Bölcskei, H., "Soft-output sphere decoding: Algorithms and VLSI implementation," *IEEE Journal on Selected Areas in Communications*, vol. 26, pp. 290–300, Feb. 2008.

[88] Su, K. and Wassel, I. J., "Efficient MIMO Detection by Successive Projection," in *IEEE International Symposium on Information Theory*, Sept. 2005.

[89] Tarokh, V., Seshadri, N., and Calderbank, A. R., "Space-Time Codes for High Data Rate Wireless Communication: Performance Criterion and Code Construction," *IEEE Transactions on Information Theory*, vol. 44, pp. 744–765, Mar. 1998.

[90] ten Brink, S., Kramer, G., and Ashikhmin, A., "Design of low-density parity-check codes for modulation and detection," *IEEE Transactions on Communications*, vol. 52, pp. 670–678, Apr. 2004.

[91] Varanasi, M. K., "Group detection for synchronous Gaussian code-division multiple-access channels," *IEEE Transactions on Information Theory*, vol. 41, p. 1083Ű1096, July 1995.

[92] Viterbo, E. and Boutros, J., "A universal lattice code decoder for fading channels," *IEEE Transactions on Information Theory*, vol. 45, pp. 1639–1642, July 1999.

[93] Wang, R. and Giannakis, G., "Approaching MIMO channel capacity with reduced-complexity soft sphere decoding," in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC'04)*, vol. 3, pp. 1620–1625, 21.-25. Mar. 2004.

[94] Wang, R. and Giannakis, G., "Approaching MIMO channel capacity with Soft Detection Based on Hard Sphere Decoding," *IEEE Transactions on Communications*, vol. 54, pp. 587–590, Apr. 2006.

[95] Waters, D. W., *Signal Detection Strategies and Algorithms for Multiple-Input Multiple-Output Channels*. PhD in Electrical and Computer Engineering, Georgia Institute of Technology, Nov. 2005.

[96] Waters, D. W. and Barry, J. R., "The Chase Family of Detection Algorithms for MIMO Channels," in *International Global Communications Conference (Globecom'04)*, vol. 4, pp. 2635–2639, 29.-3. Nov./Dec. 2004.

[97] Waters, D. W. and Barry, J. R., "The Sorted-QR Chase Detector for Multiple-Input Multiple-Output Channels," in *IEEE Wireless Communications and Networking Conference 2005, (WCNC'05)*, (New Orleans, Louisiana, USA), pp. 538–543, 13.-17.Mar. 2005.

[98] Waters, D. W. and Barry, J. R., "The Chase family of Detection Algorithms for MIMO Channels," *IEEE Transactions on Signal Processing*, vol. 56, pp. 739–747, Feb. 2008.

[99] Weinstein, S. B. and Ebert, P. M., "Data transmission by frequency division multiplexing using the discrete Fourier transform," *IEEE Transactions on Communications*, vol. COM-19, pp. 628–634, Oct. 1971.

[100] Wolniansky, P., Foschini, G., Golden, G., and Valenzuela, R., "V-BLAST: an architecture for realizing very high data rates over the rich-scattering wireless channel," in *URSI ISSSE*, pp. 295–300, Sept. 1998.

[101] Wong, K. W., Tsui, C. Y., Cheng, R. S. K., and Mow, W. H., "A VLSI architecture of a K-best lattice decoding algorithm for MIMO channels," in *IEEE International Symposium on Circuits and Systems (ISCAS'02)*, vol. 3, pp. 273–276, 2002.

[102] Wübben, D., Böhnke, R., Kühn, V., and Kammeyer, K. D., "Efficient algorithm for decoding layered space-time codes," *Electronic Letters*, vol. 37, pp. 1348–1350, Oct. 2001.

[103] Wübben, D., Böhnke, R., Kühn, V., and Kammeyer, K. D., "MMSE extension of V-BLAST based on sorted QR decomposition," in *Proceedings of the IEEE Semiannual Vehicular Technology Conference (VTC2003-Fall)*, (Orlando, USA), Oct. 2003.

[104] Wübben, D., Böhnke, R., Kühn, V., and Kammeyer, K. D., "Near-maximum-likelihood detection of MIMO systems using MMSE-based lattice-reduction," in *IEEE Conference on Communications*, vol. 2, pp. 798–802, June 2004.

[105] Yee, M. S., "Max-log-MAP sphere decoder," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'05)*, vol. 3, pp. 1013–1016, 18.-23. Mar. 2005.

[106] Zheng, J., Bai, B., and Li, Y., "Clipping value estimate for iterative tree search detection," *Journal of Commuications and Networks*, 2009. submitted.

[107] Zheng, L. and Tse, D., "Diversity and Multiplexing: A Fundamental Tradeo in Multiple-Antenna Channels," *IEEE Transactions on Information Theory*, vol. 49, pp. 1073–1096, May 2003.

[108] Zimmermann, E., *Complexity Aspects in Near-Capacity MIMO Detection-Decoding*. PhD thesis, Technische Universität Dresden, Aug. 2007.

[109] Zimmermann, E. and Fettweis, G., "Unbiased MMSE Tree Search MIMO Detection," in *International Symposium on Wireless Personal Multimedia Communications (WPMC'06)*, (San Diego, USA), 17.-20. Sept. 2006.

[110] Zimmermann, E. and Fettweis, G., "Generalized Smart Candidate Adding for Tree Search Based MIMO Detection," in *ITG/IEEE Workshop on Smart Antennas (WSA'07)*, (Vienna, Austria), Feb. 2007.

[111] Zimmermann, E. and Fettweis, G., "On the Efficiency of the Sequential Detector for Solving Soft-Output Detection Problems," *IEEE Communications Letters*, vol. 12, pp. 840–842, Nov. 2008.

[112] Zimmermann, E., Milliner, D. L., Barry, J. R., and Fettweis, G., "Optimal LLR Clipping Levels for Mixed Hard/Soft Output Detection, to appear," in *IEEE 2008 Global Communications Conference, (Globecom 2008)*, (New Orleans), Nov. 2008.

[113] Zimmermann, E., Milliner, D. L., Fettweis, G., and Barry, J. R., "A parallel smart candidate adding algorithm for soft-output MIMO detection," in *Proc. 7th International ITG Conference on Source and Channel Coding (SCC'08)*, (Ulm, Germany), Jan. 2008.