2017

Complexity awar C-RAN scheduling for LDPC codes over BEC

Kyle Gordon Whetzel

Complexity Aware C-RAN Scheduling for LDPC Codes Over BEC

Kyle Whetzel

Problem Report submitted to the
College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements
for the degree of

Master of Science
in
Electrical Engineering

Daryl S. Reynolds, Ph.D.
Brian Woerner, Ph.D.
Matthew C. Valenti, Ph.D., Chair

Lane Department of Computer Science and Electrical Engineering

Morgantown, West Virginia
2017

Keywords: Computational-outage probability, Tanner Graphs, finite-length, error rates

Abstract

Complexity Aware C-RAN Scheduling for
LDPC Codes Over BEC

Kyle Whetzel

Effective transmission of data over a noisy wireless channel is a vital part of today's high speed technology driven society. In a wireless cell network, information is sent from mobile users to base stations. The information being transmitted is protected by error-control codes. In a conventional architecture the signal processing, including error-control decoding, is performed locally at each base station. Recently, a new architecture has emerged called Centralized Radio Access Network (C-RAN), which involves the centralized processing of the signals in a computing cloud. Using a computing cloud allows computational resources to be pooled, which improves utilization and efficiency. When the computational resources are finite and when the computational load varies over time, then there is a chance that the load exceeds the available resources. This situation creates a so-called computational outage, which has characteristics that are similar to outages caused by channel fading or interference.

In this report, the computational complexity is quantified for a common class of error-correcting codes known as low-density parity check (LDPC) codes. To make the analysis tractable, a binary erasure channel is assumed. The concept of density evolution is used to obtain the complexity as a function of the code design parameters and the signal-to-interference-plus-noise ratio (SINR) of the channel. The analysis shows that there is a trade-off in that aggressively signaling at a high data rate causes high computational demands, while conservatively backing off on the rate can dramatically reduce the computational demand. Motivated by this trade-off, a scheduling algorithm is developed that balances the demands for high throughput and low computational outage rates.

Contents

Chapter 1

Introduction

Smart phones and other connected mobile devices have become vital to many people in today's society. With the rise of their popularity has come a demand for more data from wireless access networks. This demand is not done growing. By the year 2020 mobile traffic volume is expected to increase 1000 times compared to what it was just a few years ago[1]. An increase in wireless system's capacity and quality must be achieved in order to meet these demands. This will require a progression to a new 5th generation of mobile communication networks. Several novel technologies have been proposed in order to make this transition to 5G. One of these proposals is a shift to a densification of radio access points. Accompanied with these denser networks is the idea of centralizing portions of wireless systems functions to achieve coordination gains. This proposed technology is coined *C-RAN*, or Centralized Radio Access Networks.

1.1 C-RAN

A cellular base station consists of several components. These components can be grouped as two main entities. One that does the processing in the analog domain, including the power amplification and the RF circuitry. The other processing in the digital domain. To convert between the two analog to digital converters (ADC) and digital to analog converters (DAC) are required. In a traditional cellular network these components are packaged together in a single unit, one per base station. However, as technology has improved the analog components became lighter and cheaper allowing them to be placed close to the antenna at the top of towers. The digital equipment, the so called *base band unit* (BBU), was left at the base of towers in an equipment shack. This separation of RF and BBU has several benefits. First and foremost the antenna feed line, which is a major source of

loss, can be made very short. Also the base band equipment can be placed in a controlled enclosure with temperature control and plentiful power. However, an ADC is required to be maintained close to the RF unit at the top of towers.

Once digitized the information can then be fed though a high speed optical cable to the BBU. This link between the RF component and BBU is known as the *fronthaul* link. Common Public Radio Interface (CPRI) is the protocol used for this link. CPRI transmits a constant bit rate up to 12 Gbps. It allows for a link of up to 40 kilometers with a latency of only 0.1 ms. This long distance allows BBUs for multiple cells to be consolidated in so called *base band hotels*. This lends to cost effectiveness because less infrastructure is needed for delivering power, temperature control, and network links to the BBUs [2].

These base band hotels also allow for joint processing and the sharing of computing resources. This sharing of resources creates a statistical multiplexing gain. The gain comes from exploiting temporal and spatial traffic fluctuations which are inherent to cellular networks . Resources are able to be efficiently allocated according to the needs at a given time.

With these advantages of C-RAN come some challenges. The main one being the time pressures of the system. Mobile networks are hard real-time systems with tight timing and protocol constraints [3]. Data must be quickly transmitted though the fronthaul and processed in real time with a hard deadline. It is possible for the capacity of the fronthaul link to not be sufficient to meet these deadlines. There is also finite computing resources at each base band hotel. It is possible that at a given time the computational load demanded by the network is greater than the processing resources available. If either of these situations arise, either insufficient fronthaul capacity or process rescources, a *computational outage* is said to occur. These computational outages are just as detrimental to wireless systems as outages caused by fading and interference [4].

Of the tasks handled at these centralized BBUs forward error control (FEC) is amongst one of the most computationally intensive [5]. FEC aims to add redundancy to transmitted data such that it can recover from errors occurred during transmission. By adding this redundancy data rate is compromised. The Shannon Capacity is the theoretical limit of data rate for a given amount of noise [6]. A lot of literature has focused on maximizing the code rate to approach the Shannon Capacity. However, signaling at higher data rates requires more computational resources for decoding. This trade-off becomes very important in C-RAN systems. It will become the duty of a C-RAN *scheduler* to select the coding scheme in real time to meet the demands of the system.

1.2 Problem Statement

In this problem report variabilities common to wireless networks, such as locations of mobiles and channel conditions, will be simulated to see their impact on processing load. Scheduling algorithms will be examined and analyzed to see their effect on performance of the system. The focus will be on the transmission from the mobile unit to the base station, known as the *uplink*. Transmissions are assumed to be encoded with LDPC codes and the channel is to be modeled as a binary erasure channel. This set up allows for precise calculation of decoding complexity as a function of parameters of the code and the quality of the channel.

Chapter 2

LDPC Codes

By nature, wireless communication is inherently unreliable. Whether it be interference from other transmitters, thermal noise, or speed relative to the base station, errors are going to occur during propagation through the system. This gives rise for the need of error correcting capabilities in wireless systems. Forward error correction codes add controlled redundancy to to data in order to combat these errors which are inevitable. LDPC codes are a type of error correcting codes first developed in 1963 by Robert Gallager [7]. However due to lack of technology they were mostly ignored until just recently.

2.1 Forward Error Correction

2.1.1 Linear Block Codes

Consider a data transmission system in which digital data is segmented into *messages* u of length k bits. A *block code* maps each message u to a *codeword* $\mathbf{c}$ of length n bits, where $n > k$. The ratio of message bits to the total number of bits in a codeword, $R = k/n$, is called the *code rate*. $\mathcal{C}$ is the set of all code words. $|\mathcal{C}| = 2^k$ is the total number of codewords contained in $\mathcal{C}$. k is also known as the dimension of the code. Block codes are often denoted as (n, k) to show the size of the codewords and the messages.

An example of an error control block code is a repetition code. In a repetition code $k = 1$, or there is only one message bit. The bit is repeated n times. The rate of this code is $R = 1/n$. While errors are easily correctable with this scheme the throughput of the system would be quite low since there is only one information bit per codeword.

On the opposite end of the spectrum is the single parity-check code. The codewords are the message and one additional parity-check bit. The length of a codeword is $n = k + 1$. Thus the code rate is $R = k/(k + 1)$. The throughput with the code is high due to the high code rate, but the error correcting capability is minimal. As these two examples demonstrate, in coding theory a major dilemma is the trade off between the throughput and the amount of errors that can be handled during transmission.

A set $\mathcal{G}$ is considered a *spanning set* if every vector in $\mathcal{C}$ can be found by taking a linear combination of vectors in $\mathcal{G}$. If a spanning set $\mathcal{G}$ has the minimum number of vectors required to span $\mathcal{C}$, $\mathcal{G}$ is known as a minimal spanning set or a *basis*. There can be multiple bases for any code $\mathcal{C}$. The number of vectors in any basis of $\mathcal{C}$ is known as the *dimensionality* of the code. It is often denoted as $dim(\mathcal{C})$. The dimensionality of $\mathcal{C}$, $dim(\mathcal{C}) = |\mathcal{G}| = k$, where k is the length of a codeword.

Let $F = \{0, 1\}$ be a field under modulo-2 addition and multiplication. *Hamming Space* $\mathcal{V}_n = \{0, 1\}^n$ is the set of length n binary vectors, and is a vector space over F. A *binary linear code* $\mathcal{C}$ can be formed by taking all linear combinations of k linearly independent vectors from $\mathcal{V}_n$, and that set of k codewords is a basis for $\mathcal{C}$. In a binary linear code the sum of any two codewords is also a codeword. For example if $c_i, c_j \in \mathcal{C}$, then $c_i + c_j \in \mathcal{C}$. A binary linear code must contain an all 0's codeword. There is a one-to-one correspondence between messages and codewords in a linear code, and the code is able to convey k bits of information. The number of locations two codewords vary is called the *distance* between the two. The greater the minimum distance between any two words in a code leads to more errors being able to be corrected.

2.1.2 Generator Matrices

Any codeword $c \in \mathcal{C}$ can be found by forming a linear combination of basis vectors $g \in \mathcal{G}$. Multiplying a vector by a matrix results in a linear combination of the rows of that matrix. Thus, any codeword c contained in $\mathcal{C}$ can be formed by multiplying a k-dimensional vector $u \in \{0, 1\}^k$ by the *generator matrix* G. The rows of G are the basis vectors. The size of G is k x n

$$G = \begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_{k-1} \end{bmatrix}$$

As an equation, $c = uG$. Where u is any binary k-tuple known as the data word or message.

2.1.3 Dual Codes and Parity Check Matrices

Two vectors are said to be *orthogonal* if their inner product is zero. Let $\mathcal{C}$ be a subspace of $\mathcal{V}_n$. The *dual code*, $\mathcal{C}^{\perp}$, of $\mathcal{C}$ is the set of all vectors w $\in \mathcal{V}_n$ that are orthogonal to every vector $c \in \mathcal{C}$. The dimensionality theorem states that if $dim(\mathcal{C}) = k$ then $dim(\mathcal{C}^{\perp}) = n - k$

A *parity-check matrix*, H, is a rank $(n - k)$ matrix whose rows span the dual code $\mathcal{C}^{\perp}$. The parity check matrix H is usually is full rank, in which case H is a generator matrix for $\mathcal{C}^{\perp}$. While G uniquely specifies how messages u are mapped to codewords c, parity-check matrices are not unique. All that is required for a parity-check matrix H is that it spans the dual code $\mathcal{C}^{\perp}$. This leads to several ways to design a code using the parity-check matrix. There are many uses of the H matrix. ts. Randomly designed LDPC codes gives rise to more advantages. The minimum number of columns of H that sum to 0 is also the minimum distance between code words. In section 2.3 designing, encoding, and decoding using H will be further examined.

2.1.4 Systematic Codes

A linear block code can be designed such that codewords are partitioned as $c = (u, p)$. In this p contains $n - k$ parity bits. If codewords and messages are related in this way the code is said to be in *systematic form*. For a generator matrix G the standard systematic form is: $G = [I_k|P]$ where I_k is a k x k identity matrix.

If a code is in the systematic form one possible parity check matrix H can be found through transformation of the generator matrix G. Using matrix manipulation the parity check matrix can be found as $H = [P^T|I_{n-k}]$. As stated, this is just one possible way of designing a parity check matrix. A method to directly form the parity check matrix will be examined in section 2.3.

2.2 The Binary Erasure Channel

The Binary Erasure Channel (BEC) is a communications system model used commonly in coding theory. In the BEC model the input is a binary bit, data 0 or data 1. There are three possible outputs in the BEC. The receiver either correctly receives the bit that was transmitted (zero or one), or a message stating that the bit was not received, or "erased". These errors that occur during transmission are known as an erasure e. A bit is erased during transmission with a probability ϵ.

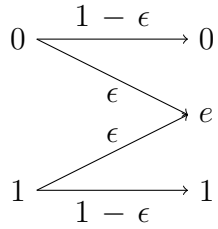

Figure 2.1: The Binary Erasure Channel

Therefore a bit is received correctly with the probability $1-\epsilon$. It is possible to reliably communicate over the BEC with a code of rate $R = 1-\epsilon$, where ϵ is the erasure probability of the channel. Figure 2.1 shows a visualization of the binary erasure channel.

2.3 Encoding and Decoding using Parity-Check Matrices

2.3.1 Single Parity-Check Codes

The single parity-check code operates by ensuring the number of 1s in any received codeword is always even. Consider a rate $R = 5/6$ parity check code.

$$\mathbf{c} = [\underbrace{1 \quad 0 \quad 1 \quad 0 \quad 1}_{\mathbf{u}} \quad \underbrace{1}_{\text{parity bit}} \quad]$$

In this example the single parity bit is set to one since the number of ones in the message u is odd. The selection of the parity bit can also be thought of as an exclusive or (XOR) operation. With this type of code a single erasure can be correct at any position within the code. The shortcoming is that only one erasure can be corrected.

2.3.2 Encoding Product Codes

A product code is a type of error controlling code that provides the ability to correct multiple erasures. With a product code data is placed into a k x k rectangular array. Each row of the of the array is encoded with a single parity-check code. Each column is also encoded with a single parity-check code. The result of this is a code of rate $R = k^2/(k+1)^2$. The following is an example

with $k = 2$.

$c_0 = u_0$	$c_1 = u_1$	$c_2 = c_0 \oplus c_1$	=	1 0 1
$c_3 = u_2$	$c_4 = u_3$	$c_5 = c_3 \oplus c_4$		1 1 0
$c_6 = c_0 \oplus c_3$	$c_7 = c_1 \oplus c_4$	$c_8 = c_2 \oplus c_5$		0 1 1

The $\oplus$ designates an XOR operation. With this size product code it can be seen that there are 4 message bits and 9 total code bits, thus the rate is $R = 4/9$.

2.3.3 Decoding Product Codes

Decoding a product is performed by iteratively decoding the single parity-check code on each row and column. The following is an example of decoding a product code.

e e 1		e e 1		1 e 1		1 0 1	
1 e 0	$\Rightarrow$	1 1 0	$\Rightarrow$	1 1 0	$\Rightarrow$	1 1 0	
0 e e		0 e e		0 e 1		0 1 1	
Received word		Row decoding		Column decoding		Row decoding	

In this example 5 erasures are successfully corrected. This particular code is capable of correcting all patterns of 3 erasures and some, but not all, patterns of more than 3. Following through the example it can be seen that at least one erasure is successfully correct with each iteration.

2.3.4 Product Code Parity-Check Matrix

The example product code in section 2.3.2 is characterized by the set of 5 linearly-independent equation. One equation for each of the parity bits.

$$
\begin{aligned}
c_2 = c_0 \oplus c_1 \quad &\Rightarrow \quad c_0 \oplus c_1 \oplus c_2 = 0 \\
c_5 = c_3 \oplus c_4 \quad &\Rightarrow \quad c_3 \oplus c_4 \oplus c_5 = 0 \\
c_6 = c_0 \oplus c_3 \quad &\Rightarrow \quad c_0 \oplus c_3 \oplus c_6 = 0 \\
c_7 = c_1 \oplus c_4 \quad &\Rightarrow \quad c_1 \oplus c_4 \oplus c_7 = 0 \\
c_8 = c_2 \oplus c_5 \quad &\Rightarrow \quad c_2 \oplus c_5 \oplus c_8 = 0
\end{aligned}
$$

It generally takes $(n - k)$ linearly independent equations to specify a linear error control code. The system of these equations can be expressed in matrix form as $cH^T = 0$, in which H is a parity-check matrix. The following is the parity check matrix for this particular product code.

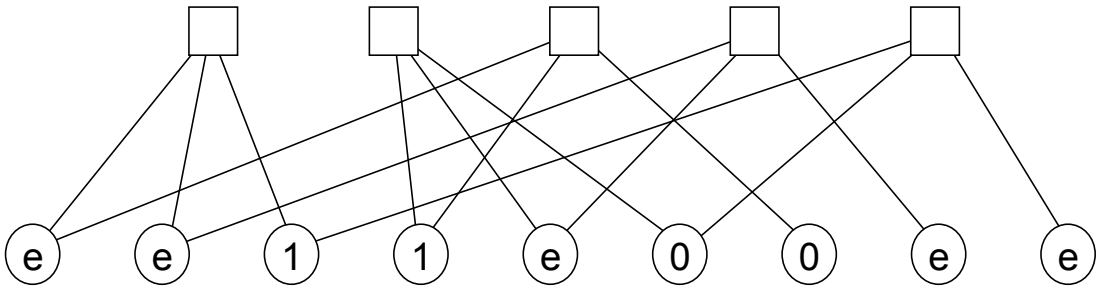

Figure 2.2: Example Tanner Graph

$$\begin{array}{rcl} c_0 \oplus c_1 \oplus c_2 & = & 0 \\ c_3 \oplus c_4 \oplus c_5 & = & 0 \\ c_0 \oplus c_3 \oplus c_6 & = & 0 \\ c_1 \oplus c_4 \oplus c_7 & = & 0 \\ c_2 \oplus c_5 \oplus c_8 & = & 0 \end{array} \quad \Leftrightarrow \quad H = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

System of equations Parity-check matrix

Each equation corresponds to a row of H. Each code bit used in an equation corresponds to the column location of 1's for that particular equation.

2.3.5 Tanner Graphs

Tanner Graphs are a way to represent parity-check matrices. Tanner Graphs are a bipartite graph, or bigraph. A bipartite graph is one that has two independent sets of vertexes. The edges never connect vertexes in the same set, they must connect a vertex in one set to a vertex in the other. In Tanner Graphs the two sets are called the check nodes and the variable nodes. The check nodes represent the $n - k$ parity-check equations. The variable nodes represent the n code bits. If an entry in the parity-check matrix $H_{i,j} = 1$ then the i^{th} check node is connected to the j^{th} variable node. Figure 2.2 shows the Tanner Graph for the parity check matrix example examined in section 2.3.4.

The check nodes are connected to the variable nodes as specified by the parity-check matrix and system of equations for the code. For example the first check node, the box on the top left, is connected to the first three variable nodes as in shown in H. Decoding can be performed by using the Tanner Graph. First the variable nodes are loaded with the observed code bits. In figure 2.2 an example received message containing erasures is shown. Each check node examines its incident

edges to see if it is connected to a single variable node containing an erasure. If this is the case those check nodes can then correct that erasure using the logic of a single parity-check code. Looking again at the example Tanner graph, the first check node is connected to two erasures, therefor no decoding can be performed at this time. Looking at the second check node, the variable nodes connected to it contain one single erasure. Since the other variable nodes contain a single 1 the erasure can be decoded to also be a 1 to cause even parity. Each successful decoding of an erasure creates the possibility that a check node that was unable to perform decoding in a previous iteration is now able to. This process of checking incident edges at each check node is then iterated until all erasures are corrected or there are no more possible corrections to be made. A set of erased variable nodes that prevent complete correction, regardless of the decoding of other variables, is known as a stopping set.

2.4 Parameters of LDPC Codes

The *degree* of a check node is the number of incident edges connected to it. This number is equal to the the number of ones, or the *Hamming Weight*, of the corresponding row in the H matrix. The complexity of decoding using a parity-check matrix is dependent on the degree of the check nodes. Therefor it is desired to have H matrices with low row weight. The length of the code is also important when designing a code with a party-check matrix. In order to achieve high code rates that approach capacity a long code is needed.

A Low Density Parity-Check (LDPC) code is characterized by a sparse parity-check matrix. A sparse matrix is a large matrix containing very few ones. Randomly designed sparse matrices lend to both advantages of having a long code and having low row weights. The minimum number of columns of H that sum to 0 is also the minimum distance between code words. Because the matrix is sparse it will take many rows to sum to zero. Also row and column weights relate to the complexity of the code, and these remained fixed as the code rate increases. This leads to decoding complexity being linearly increased with the length of the code, opposed to exponential growth of codes that don't have a sparse H matrix.

2.4.1 Regular and Irregular LDPC Codes

If the rows of a LDPC parity-check matrix have constant weight, or constant check-node degree d_c, the code is said to be *check regular*. If the Hamming weight of the columns of a LDPC parity-

check matrix are also constant, that is the variable-node degree d_v is constant, the code is said to be *regular*. A naming convention for regular codes is in the form of (d_v, d_c) regular code. For example a (3,4) regular code contains variable nodes all of degree 3 and variable nodes all of degree 4.

Although regular codes are simple and perform moderately well, they are not capable of achieving capacity. Irregular LDPC codes are those without constant degree distribution, and when properly designed are capable of achieving capacity. When designing irregular LDPC codes the variable node distribution is not constant. The check node degree is however held constant, or close to it. The design consists of choosing the proper degree distribution. The edge perspective ρ_i is the fraction of *edges* touching degree i check nodes, and λ_i is the fraction of *edges* touching degree i variable nodes.

The degree distributions can be described in polynomial form. $\rho(x) = \sum_i \rho_i x^{i-1}$ is the distribution for check nodes, and $\lambda(x) = \sum_i \lambda_i x^{i-1}$ for variable nodes. For the degree distributions to be valid they must satisfy the following:

$$0 \leq \lambda_i \leq 1 \quad \text{and} \quad 0 \leq \rho_i \leq 1$$
$$\sum \lambda_i = 1 \quad \text{and} \quad \sum \rho_i = 1$$

The first equations show that any fraction of degree i nodes obviously must be less than or equal to one. The second equations state that the sum of all degree coefficients must sum to unity. From the book chapter [8] the design rate of the code can be found in terms of the polynomial form of the degree distributions.

$$R = 1 - \frac{\int_0^1 \rho(x)dx}{\int_0^1 \lambda(x)dx} \tag{2.1}$$

2.4.2 Density Evolution

Density Evolution tracks the probability of erasures through iterations of decoding. For a LDPC code, either regular or irregular code, it can be found that the probability of a variable-node remaining erased after the ℓ^{th} iteration is

$$\epsilon_\ell = \epsilon_0 \lambda \left(1 - \rho \left(1 - \epsilon_{\ell-1}\right)\right) \tag{2.2}$$

Where ϵ_0 is the initial erasure probability before any error correcting has been performed. A code is said to *converge* if the erasure probability is reduced to or below a certain threshold after a sufficiently large number of iterations. Therefor a decoder could receive a stopping set and still

converge. Let this convergence threshold cutoff be denoted as ϵ_{thresh}. The criteria for convergence is that the erasure probability is constantly decreasing. In terms of variables, convergence for an initial channel erasure probability ϵ_0 is defined as $\epsilon_\ell < \epsilon_{\ell-1}$ for all ℓ in which $\epsilon_\ell > \epsilon_{thresh}$. By combining the inequality for convergence with equation 2.2 yields

$$\underbrace{\epsilon_0 \lambda\left(1-\rho\left(1-\epsilon_{\ell-1}\right)\right)}_{\epsilon_\ell} \quad < \quad \epsilon_{\ell-1} \tag{2.3}$$

A change of variable, $\epsilon_{\ell-1} \rightarrow x$, is performed and the inequality 2.3 is then defined as a function to get the following equation.

$$f(\epsilon_0, x) \quad = \quad \epsilon_0 \lambda\left(1-\rho\left(1-x\right)\right) \tag{2.4}$$

Decoder convergence is possible if and only if $f(\epsilon_0, x) < x$ for all $\epsilon_{thresh} \leq x \leq \epsilon_0$. It is useful to find the largest ϵ_0 for which the code converges. This initial erasure probability threshold, will be denoted as ϵ^*. The equation 2.4 can be rearranged to solve for ϵ_0 in order to create a function useful for finding this threshold.

$$\epsilon(x) \quad = \quad \frac{x}{\lambda\left(1-\rho\left(1-x\right)\right)} \tag{2.5}$$

To find ϵ^* the largest value of ϵ_0 such that $\epsilon_0 < \epsilon(x)$ for all x must be determined. The function $\epsilon(x)$ is a convex function, ϵ^* is its minimum value. The following is a formal way to express this threshold.

$$\epsilon^* \quad = \quad \min\{\epsilon(x) : \epsilon(x) > x\} \tag{2.6}$$

This convergence threshold is the largest channel erasure probability that can be corrected by a large *typical* LDPC code with degree polynomials $\rho(x)$, $\lambda(x)$. Figure 2.4.2 shows density evolution for a code when the ϵ_0 is just above and just below ϵ^*

2.5 Complexity Definition

The complexity of the decoder is synonymous to the amount of processing resources required to successfully decode a received message. Let K be the work required per codeword. This amount is equal to the number of decoder iterations multiplied by the number of edges in a Tanner graph. Assuming a check-regular LDPC code the number of edges is equal to the check-node degree multiplied by the number of check nodes. This can be written in equation form as $\mathrm{K} = \ell * d_c * (n-k)$.

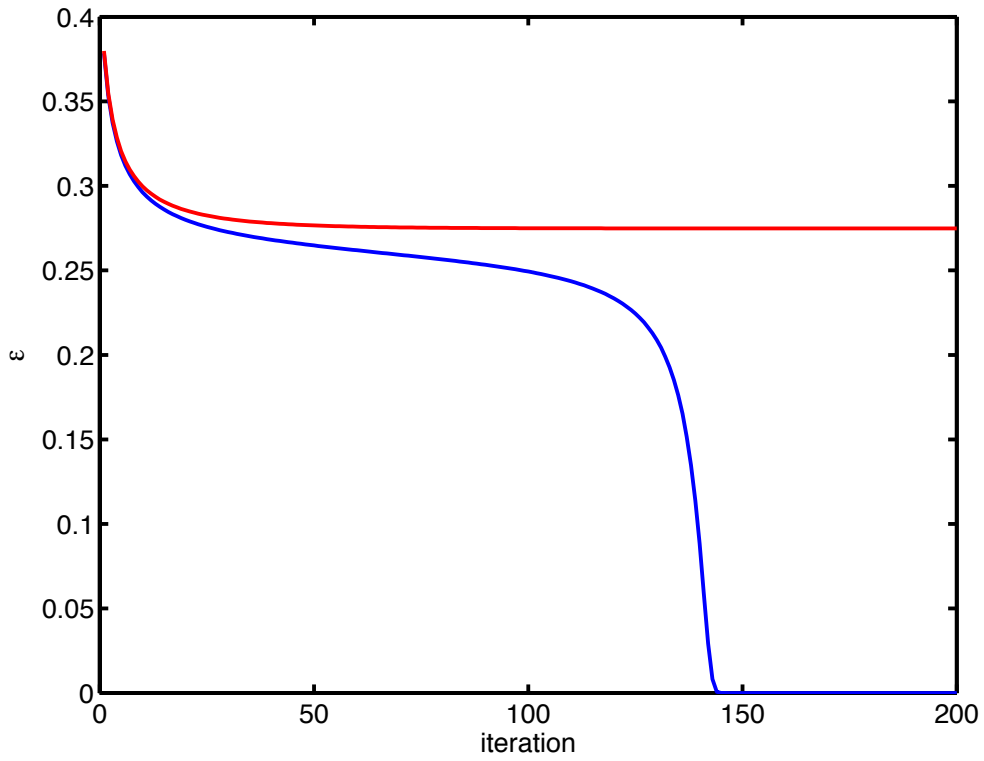

Figure 2.3: Density Evolution of a (3,6) regular code. The threshold $\epsilon^* = 0.4294$ for this code. One plot is for $\epsilon_0 = 0.43$ the other $\epsilon_0 = 0.429$

To get a better understanding of complexity versus throughput of information it would be beneficial to know the required work per data bit. Let C be this complexity per data bit. To find C, K will be normalized by dividing by the number of data bits, k. This yields the equation $\mathrm{C} = \frac{\ell d_c(n-k)}{k}$. This equation works for a code of a specific length parameters. To generalize this equation denote the variables n and k in terms of the design rate of the code R. Knowing that $R = k/n$ the complexity per code bit can be written as follows.

$$\mathrm{C} = \frac{\ell d_c(1-R)}{R} \tag{2.7}$$

2.6 Rate Optimized Codes

The first step in simulating a C-RAN system is to select a pool of codes for the scheduler to choose from. For this simulation a group of rate optimized LDPC codes will be designed. This optimization will be based off the equations in section 2.4. The key for this optimization will be to maximize the equation 2.6. Designing codes that use maximizing this ϵ^* as the optimization parameter develops codes that can handle the largest initial erasure probability ϵ_0. For ease of computation the codes will be constrained to be check regular codes. This leads to only having to design for the variable node degree, $\lambda(x)$, that maximizes ϵ^*. Many optimization tools available only solve to minimize a value. To work with this, maximizing a value is equivalent to minimizing the negative of that value. This leads to the optimization at hand being denoted as finding the $\lambda(x)$ that minimizes: $\min_{\lambda(x)}\{-\epsilon^*\}$. For the code to be valid it must meet the following constraints.

$$0 \leq \lambda_i \leq 1 \qquad \text{for } 1 \leq i \leq d_{max} \tag{2.8}$$

$$\sum_{i=1}^{d_{max}} \lambda_i = 1 \tag{2.9}$$

$$\int_0^1 \lambda(x)dx = \frac{\int_0^1 \rho(x)dx}{1-R} = \frac{\int_0^1 x^{d_c-1}dx}{1-R} \rightarrow \sum_{i=1}^{d_{max}} \frac{\lambda_i}{i} = \frac{1}{(d_c)(1-R)} \tag{2.10}$$

Where d_{max} is the largest allowable variable node degree. To improve performance it is desirable to not allow any degree 1 variable nodes. This can be done by modifying the inequalities in 2.8 such that $\lambda_1 = 0$ and $0 \leq \lambda_i \leq 1 \quad \text{for } 2 \leq i \leq d_{max}$.

R	1/5	1/4	1/3	2/5	1/2	3/5	2/3	3/4
ϵ^*	0.7275	0.7079	0.6573	0.5890	0.4777	0.3673	0.2740	0.1669

Table 2.1: Initial Erasure Probability Thresholds

2.7 Code Design and Decoding Complexities

For the simulation the constant check node degree is set to $d_c = 7$. The maximum allowable variable node degree is set to $d_v = 200$. Eight codes of different design rates varying from $R = 1/5$ to $R = 3/4$ were developed. Let (R) be the set of these eight codes available in the system. Table 2.1 shows the resulting initial erasure probability thresholds, ϵ^*, for the codes designed through optimization. This table shows that the largest initial erasure probability, ϵ_0, that can be handled by the pool of codes in the system is 0.7275. Any situation where the initial erasure probability is greater will result in failure to decode the message.

Using equation 2.7, complexity can be calculated for various values of ϵ_0. This equation requires knowledge of the number of iterations in density evolution. Density evolution, described in section 2.4.2, was simulated for the 8 codes in the pool. The convergence threshold was set to $\epsilon_{thresh} = 10^{-3}$ and the max number of iterations was set to 10^3. The simulation calculated complexity for the 8 codes with initial erasure probability ranging from 0 to just above the threshold ϵ^* for each code. The results of this simulation can be seen in figure 2.4. As the initial erasure probability ϵ_0 approaches the the maximum ϵ_0 for each code, ϵ^*, the computational complexity spikes up. This shows the desirability of complexity aware scheduling systems. If operating near the initial erasure probability threshold, the computation load can be lowered dramatically by switching to a lower coding rate.

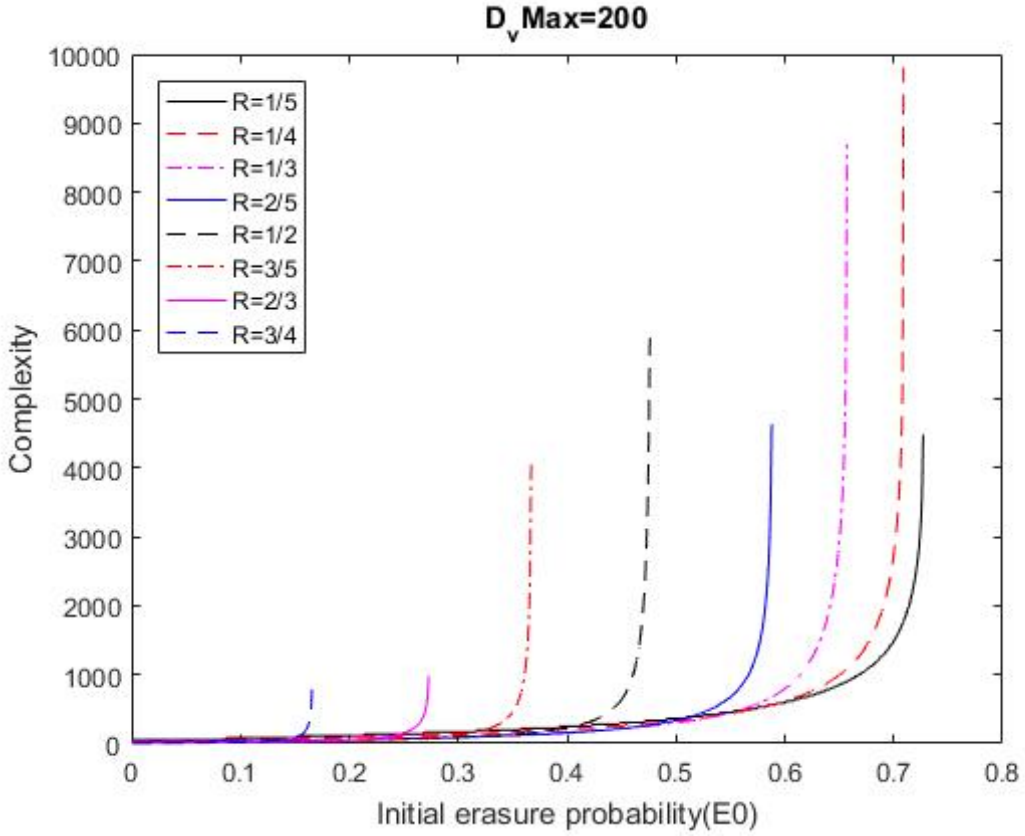

Figure 2.4: Complexity Required For Convergence. On the x-axis is the initial erasure probability ϵ_0 and on the y-axis the required decoder complexity required for each code at that ϵ_0

Chapter 3

Complexity Aware Scheduling for C-RAN

With a pool of codes created this chapter will create a C-RAN scheduling system. Four scheduler algorithms, including complexity aware and unaware schemes are discussed. The performance of these algorithms are then tested by creating a network simulation and graphical results are produced.

3.1 Mapping System Parameters to Complexity

3.1.1 Cellular Network Setup

In the simulation the base stations of the cell network are placed in a hexagonal grid pattern. The hex pattern is common in wireless networking as it maximizes the distance between neighboring base stations, reducing interference. Mobile units (MU) are randomly dropped into the system such that there is only one mobile per base station cell. The amount of mobile units that are in the system is varied from 60 percent of cells having MU, to 100 percent of the cells having a MU. The number of base stations is fixed at 110, as is their location. Figure 3.1 shows the simulated system at 100 percent of the base stations having a MU, or 100 percent system utilization. The x's represent the the 110 base stations. The o's represent the mobile units. Their location is random and will vary from trial to trial, maintaining the restriction that there is only one per cell. The +'s on the base station in the center of the system represent the base stations in the computing cluster. For this particular trial shown the computing cluster size (cs) is 7, which will be the default size.

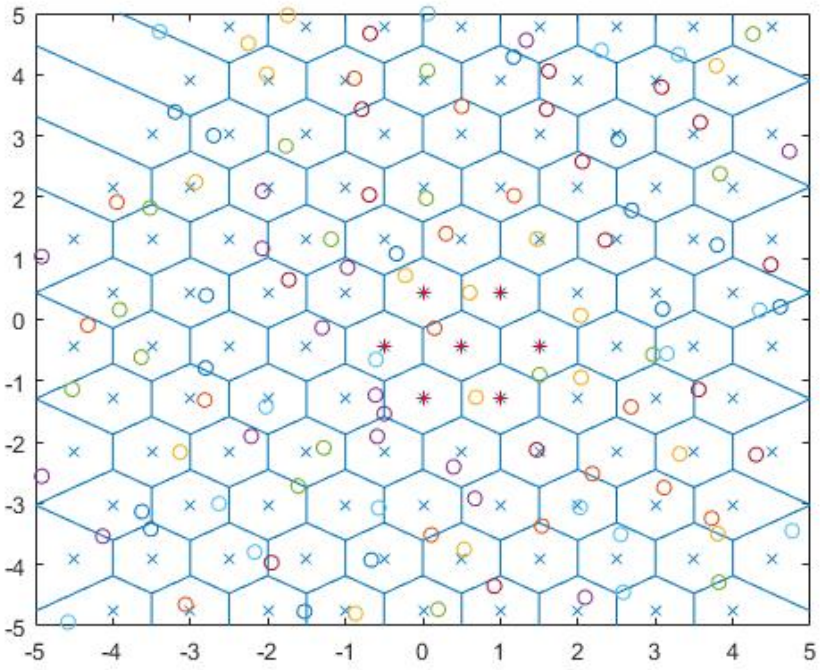

Figure 3.1: Cell Network at 100 percent Utilization

3.1.2 Simulating SINR and Complexity

When the cells are placed the Signal to Noise plus Interference (SINR) level will be calculated for each base station in the computing cluster. Equation 3.1 is the SINR at basestation Y_j

$$\gamma_j = \frac{g_{i,j}|Y_j - X_j|^{\alpha(s-1)}}{\Gamma^{-1} + \sum_{i \neq j} g_{i,j}|Y_j - X_i|^{-\alpha}|Y_i - X_i|^{s\alpha}} \tag{3.1}$$

where $g_{i,j}$ is the fading power gain from mobile unit X_i to base station Y_j. For the simulation we assume no fading, for ease of computation, so $g_{i,j}$ is always 1. The rest of the equation is defined by $|Y_j - X_i|$ being the distance from a base station j to a mobile unit i, α is the path-loss exponent, s is the compensation factor for fractional power control, Γ the SIR at unit distance is set to 20dB, and the summation is over all interferers. The power control factor is set to 0.1 as that is the value reported to maximize the throughput in [9]. The path-loss exponent's default value is assumed to be 3, but will be varied in one of the trials.

Once SINR is calculated for each base station in the computing cluster an algorithm for mapping SINR to initial erasure probability ϵ_0 is required. For this the complementary cumulative distribution function (CCDF) of the SINRs obtained is used. The CCDF makes use of the SINRs

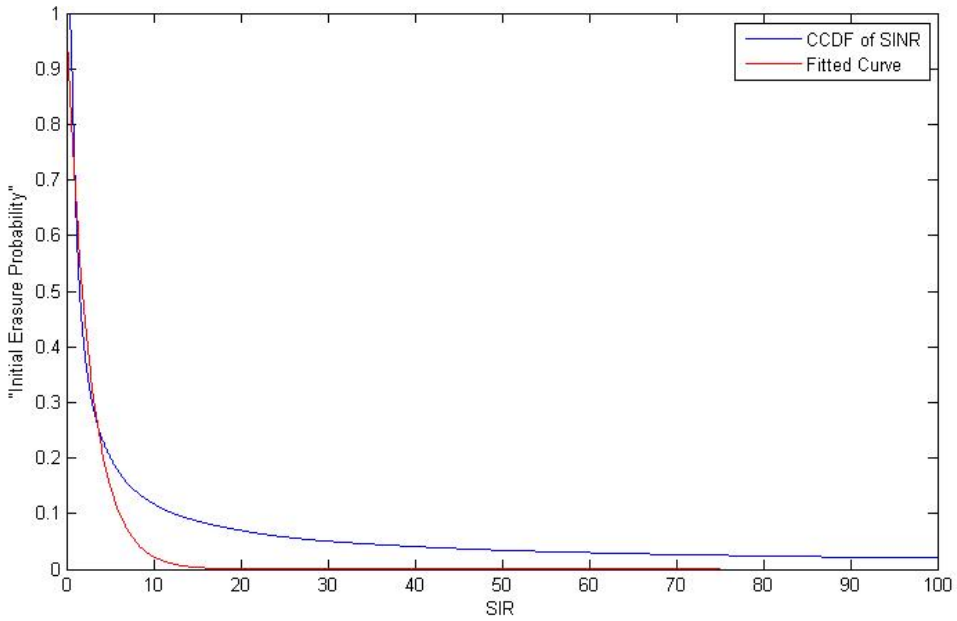

Figure 3.2: CCDF of SINR and created Exponential fitted Curve

obtained with all of the default simulation values. By definition, the CDF of a random variable X evaluated at x is the probability that X is less than x. That is $F_X(x) = P(X < x)$. Since this is a probability, as is ϵ_0, the value ranges from 0 to 1. Because high SINR values need to be mapped to low initial erasure probability ϵ_0 values, the CCDF is used. The complementary cumulative distribution function CCDF=1-CDF. Using MATLAB's curve fitting tools an exponential function is fitted to the CCDF. This exponential curve is used to map all SINR values to a corresponding initial erasure probability. The generated CCDF and resulting exponential mapping function are shown in figure 3.2.

3.2 Scheduling Algorithms

Four different scheduling algorithms are used for simulation. These include complexity unaware and complexity aware methods. Let C_{server} be the total computational complexity resources available to a computing cluster. Let C_κ be the computational complexity required at a base station for user κ within the computing cluster. The rate, $r_\kappa \in \mathcal{R}$, is the code rate assigned to the user κ. Throughput, T, will be defined as the average of all assigned code rates r_κ contained in the computing cluster.

3.2.1 Max Rate Selection

The first and simplest scheduling algorithm used in the system is maximum code rate selection (MRS). This method is not complexity aware and no code switching is performed by the scheduler. Each user, κ, is assigned the maximum code rate available based on ϵ_0 at the base station and ϵ^* of the codes in the pool. If the ϵ_0 for a certain user is greater than the largest ϵ^* available, the rate for that user, r_κ, is set to 0. Complexity, C_κ, is then calculated for each user within the computing cluster. If $\sum_{\kappa=1}^{cs} \mathrm{C}_\kappa > \mathrm{C}_{server}$ a computational outage occurs. The rate is set to zero for all users resulting in no throughput.

3.2.2 Easiest Job First

Easiest job first (EJF) is a complexity aware scheduling algorithm. It initializes by performing the MRS code selection scheme. If $\sum_{\kappa=1}^{cs} \mathrm{C}_\kappa > \mathrm{C}_{server}$ the scheduler selects the user κ that has the lowest complexity value, or the easiest job. If this minimum complexity $\mathrm{C}_{ej} < \mathrm{C}_{server}$ decoding is performed for that user and C_{ej} is then added to the sum of complexities already decoded, C_{sum}. C_{sum} is set to 0 before any decoding has occurred. This process then repeats by selecting the easiest job of those remaining to be decoded as long as $\mathrm{C}_{sum} < \mathrm{C}_{server}$.

3.2.3 Local Limit

The local limit algorithm is a simple complexity aware code switching algorithm. It begins by selecting the highest possible code rate for each user as in the MRS algorithm. It sets a limit, C_{loc}, on the computational resources allocated to one user. If any user κ demands a decoder complexity greater than C_{loc}, r_κ is bumped down to a lower rate code if one exists. Complexity is then recalculated. This process is then reiterated until $\sum_{\kappa=1}^{cs} \mathrm{C}_\kappa < \mathrm{C}_{server}$, or there are no lower rate codes to recede to. If the case is the latter, and no more codes are available, a computational outage has occurred.

3.2.4 Scheduling with Complexity Cut-Off

The complexity cut-off scheduling (SCC) method is a more sophisticated code switching algorithm. Again, it initializes by setting rates to the maximum possible as in the MRS scheme. It then checks if $\sum_{\kappa=1}^{cs} \mathrm{C}_\kappa > \mathrm{C}_{server}$. If that is the case user κ^* is chosen such that C_{κ^*} is the maximum C_κ. The code rate for user κ^*, r_{κ^*}, is then decreased to the next lower coding rate if one exists. If

Parameter	Default
Path-Loss Exponent	$\alpha = 3$
Channel Utilization	$u = 100\%$
Computing Cluster Size	cs = 7
Simulation Trials	$N_{trials} = 10^5$
SIR at Unit Distance	$\Gamma = 20dB$
Fading	None
Fractional Power Control Factor	$s = 0.1$
Number of Base Stations	$N_{bs} = 110$
Convergence Threshold	$\epsilon_{thresh} = 10^{-3}$
Max Decoding Iterations	$N_{iterations} = 10^3$
Number of Codes in Pool	Size($\mathcal{R}$)=8
Max Correctable ϵ_0	$\epsilon^* = 0.7275$

Table 3.1: Default Simulation System Parameters

a lower rate code does not exist for κ^* the user κ with the next highest C is then chosen. $\sum_{\kappa=1}^{cs} \mathrm{C}_\kappa$ is then recalculated with the new coding rate. If this sum is less than C_{server} decoding commences, else the process is recursed such that an updated κ^* is selected. The recursion proceeds until the complexity constraint is met or no lower rate codes are available. It is again a computational outage if the scenario of no lower rate codes arises.

3.3 Simulation Parameters

Using Monte Carlo techniques 3 different simulations are performed with varying independent variables each time. The number of Monte Carlo trials is set to 10^5. The random variable is location of the mobile units. The same mobile placement is used for all 3 simulations. Table 3.1 shows the default values for all simulations. These values are fixed unless they are the independent variable for the particular simulation.

The first simulation casts the path-loss exponent α as the independent variable. It will be varied such that $2 \leq \alpha \leq 5$. The Path-loss exponent for free space is $\alpha = 2$. A path-loss exponent around $\alpha = 5$ is nearing the end of feasible values of α, and would be found in a non-ideal suburban environment. The second simulation will vary the channel utilization of the system. The channel usage will be varried such that $0.6 \leq u \leq 1$. The last simulation modifies the computing cluster size such that $1 \leq cs \leq 10$. Throughput, as defined in section 3.2, is the performance metric for all three simulations.

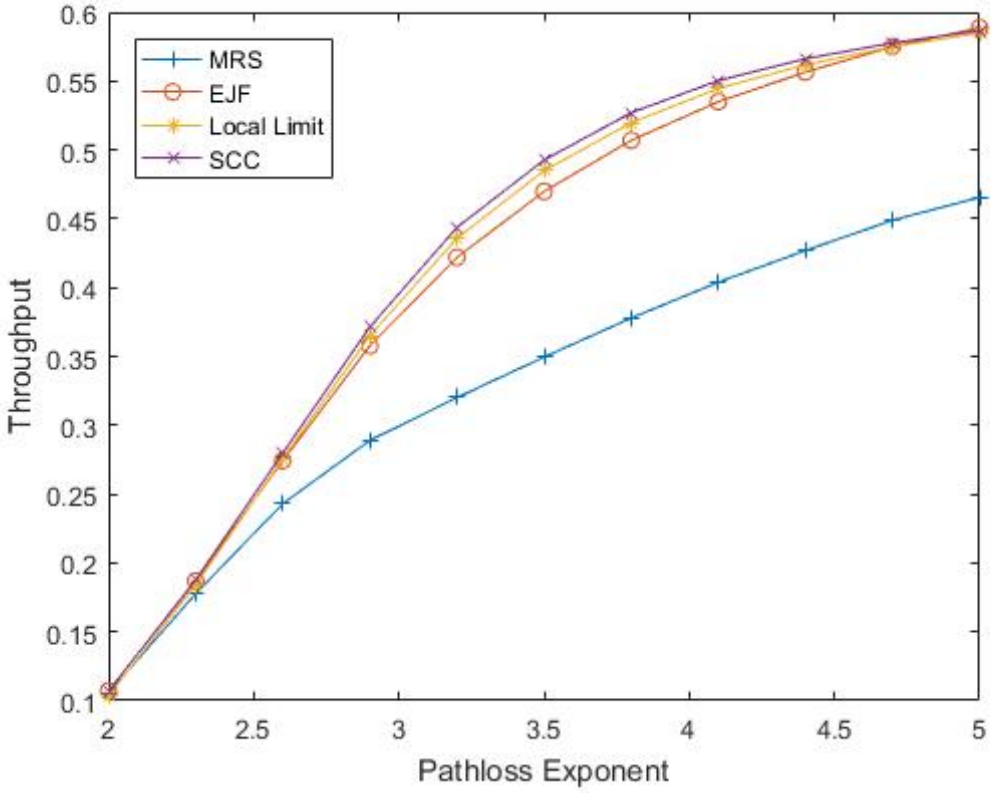

Figure 3.3: Throughput versus Path-loss exponent α. α is varied from a value for free space,2 , to 5 which is at the edge of naturally occuring values. The computing cluster size is held constant at 7 and the channel utilization constant at 100%.

3.4 Simulation Results

Figure 3.3 shows the effect on varying the pathloss exponent α on throughput for the different scheduling algorithms. When $\alpha = 2$ the throughput for all is really low due to the high interference created by interferers transmitting trough free space.

Figure 3.4 displays the results when varying the channel utilization. When randomly placing the mobile units there is no guarantee a user will be placed in every cell of the computing cluster. The throughput is lower at lower channel utilization's because no data is being transmitted if there isn't a user in that cell.

Figure 3.5 is the results of varying the computing cluster size. The first data point here is when cluster size is set to one. This is the same as a traditional cellular network where no processing resources are shared. From the plot you can see that increasing cluster size increases the throughput for all complexity aware algorithms.

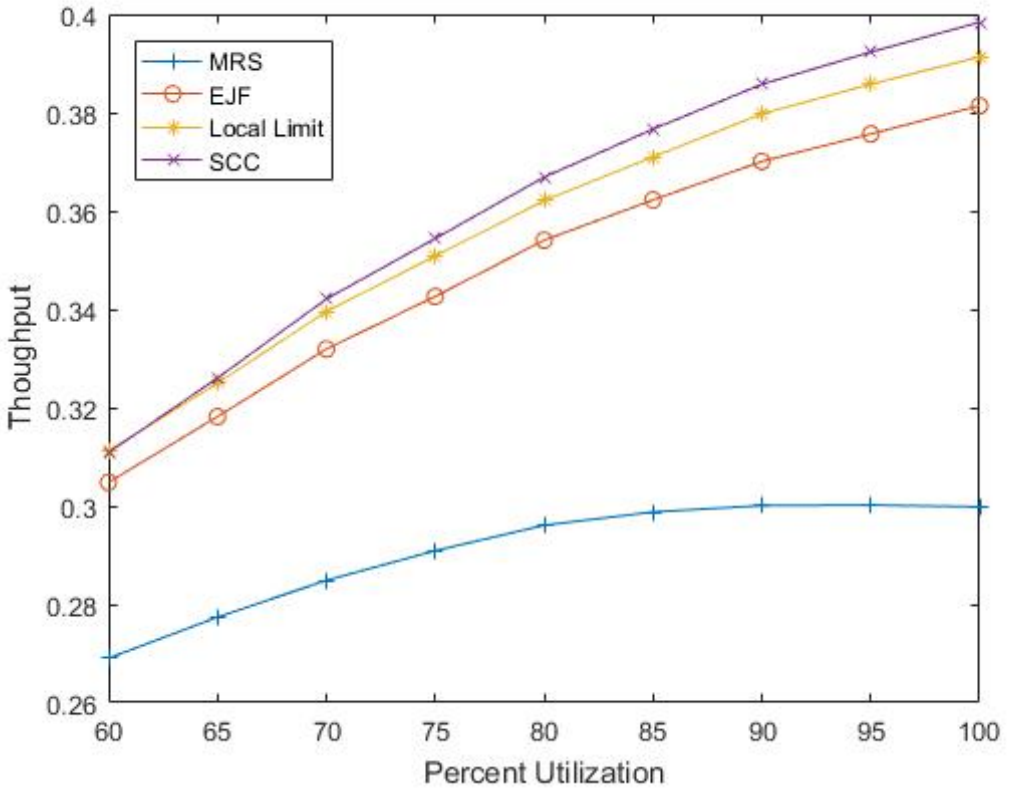

Figure 3.4: Throughput versus Channel Utilization. If there is no user in a computing cluster cell the throughput for that cell is set to zero. Pathloss exponent α is held constant at 3 and the computing cluster size at 7.

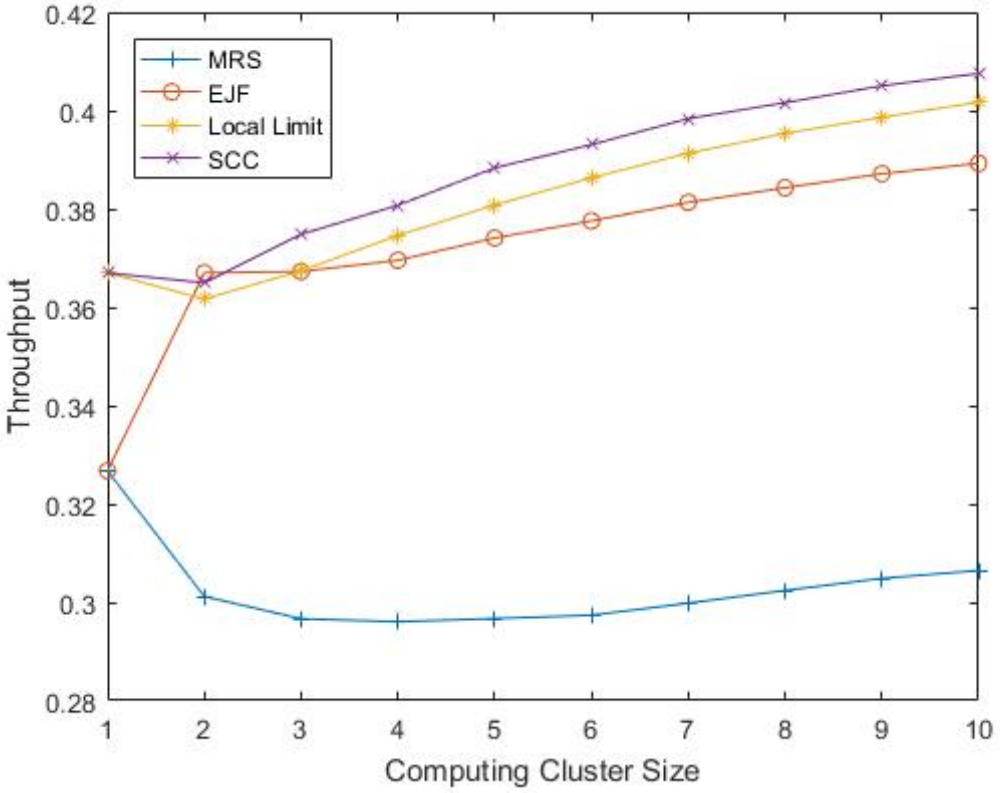

Figure 3.5: Throughput versus the Computing Cluster Size. The pathloss exponent is constant at $\alpha = 3$ and the channel utilization is held constant at 100%.

Looking at all the simulations it can be seen that the complexity aware schemes always outperform the complexity unaware MRS. In terms of throughput the more sophisticated SCC does as good or better for all values of the variables being varied.

Chapter 4

Conclusions and Future Work

This chapter will summarize the contents of this problem report. Possibilities for future work to expand upon the findings of this report will also be discussed.

4.1 Concluding Remarks

Advancements in hardware for RF technology, high speed networking, and general computing has made it feasible to process baseband signal of wireless network in a central processing pool. This network paradigm shift, known as C-RAN, creates many benefits. These include the ability to share computing resources among multiple cells and centralizing the management of system infrastructure. These benefits come along with some challenges. The primary being the load imposed on the front haul connection, and the risk that the demand on computing resources exceeds its capacity. When either the fronthaul capacity isn't sufficient, or the processing center doesn't have sufficient computation resources a outage occurs. From a provider's perspective this outage is no different from a more traditional impairment such as fading or interference.

This problem report provides connections between the varying conditions of a wireless network and the load imposed on the central processing pool. The focus was on the cellular up-link, the connection from the mobile user to the base stations. In order to perform mathematically precise and tractable analysis, transmissions are assumed to be encoded with LDPC codes sent over a binary erasure channel. These assumptions allow for the per-bit complexity to be found as a function erasure probability and code parameters such as design rate and the degree distribution.

Due to variability of user location and channel conditions, processing load can vary significantly over time. These variances cause a profound impact on the choice of scheduling techniques. For

instance a complexity unaware, rate optimizing, method might result in many computational outages whereas a complexity aware method does not. A key contribution of this report is to observe the impact of scheduling techniques on computational load, and in doing so have analyzed several effective strategies for computationally aware scheduling.

4.2 Future Work

The following is a number of examples of things to further the work done in this report and gain a better understanding of complexity aware C-RAN systems. In lieu of the binary erasure channel use the AWGN version of density evolution, as this would get rid of the need for curve fitting of SINR to initial erasure probability. Another idea to further the findings here would be to consider, and even create, other scheduler algorithms.

Another direction to go would be to differ the approach of creating the LDPC codes. For the optimization calculations add a constraint based on complexity of the code. Having a pool of complexity optimized codes drastically change the results of C-RAN similation.

References

[1] M. Olsson, C. Cavdar, P. Frenger, S. Tombaz, D. Sabella, and R. Jantti, "5GrEEn: Towards Green 5G mobile networks," in *2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Oct 2013, pp. 212–216.

[2] A. Checko, H. L. Christiansen, Y. Yan, L. Scolari, G. Kardaras, M. S. Berger, and L. Dittmann, "Cloud RAN for Mobile Networks-A Technology Overview," *IEEE Communications Surveys Tutorials*, vol. 17, no. 1, pp. 405–426, Firstquarter 2015.

[3] M. C. Valenti, S. Talarico, and P. Rost, "The Role of Computational Outage in Dense Cloud-Based Centralized Radio Access Networks," in *IEEE Global Conference on Communications*, Austin (TX), USA, December 2014.

[4] P. Rost, S. Talarico, and M. C. Valenti, "The Complexity Rate Tradeoff of Centralized Radio Access Networks," *IEEE Transactions on Wireless Communications*, vol. 14, no. 11, pp. 6164–6176, Nov 2015.

[5] P. Rost, A. Maeder, M. C. Valenti, and S. Talarico, "Computationally Aware Sum-Rate Optimal Scheduling for Centralized Radio Access Networks," in *IEEE Global Communications Conference (GLOBECOM)*, Dec 2015.

[6] C. E. Shannon, "A Mathematical Theory of Communication," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 5, no. 1, pp. 3–55, Jan. 2001.

[7] Robert G. Gallager, "Low-density parity-check codes," 1963.

[8] Christian Schlegel, *Trellis and Turbo Coding*, IEEE Press Wiley-Interscience, Piscataway, NJ Hoboken, NJ, 2004.

[9] M. Coupechoux and J. M. Kelif, "How to set the Fractional Power Control Compensation Factor in LTE," in *34th IEEE Sarnoff Symposium*, May 2011.