Theory & Implementation of LDPC Codes for Wireless Communication

Eugene Min, Austin Rothschild, Kyle Huang

March 15, 2025

1 Introduction

Error correcting codes play a profound role in wireless communications, ensuring the accuracy, reliability, and integrity of data transmitted over noisy channels. With advancements of digital hardware capability in the past two decades, Low-Density Parity-Check (LDPC) codes have been a subject of renewed interest since their initial discovery in the 1960s. Approaching theoretical capacity limits with feasible computational requirements, the adoption of LDPC codes has led to significant performance gains in modern communication systems. LDPC codes are now a fundamental error correction scheme, finding widespread success in applications such as WiFi, optical fiber links, communication satellites, and cellular systems.

In Section 2.1 we explore the history and applications of LDPC codes. In Section 2.2, we cover the encoding of LDPC codes. In section 2.3, we discuss how LDPC codes can be decoded using belief propagation algorithms. In Section 3, we utilize commercial channel modelers to simulate and explore the performance of LDPC codes with 5G cellular communication systems used as a motivating case study for the success of LDPC codes at scale. Additionally, we compare LDPC codes to other schemes used in wireless communications.

2 LDPC Coding Theory

2.1 History and Applications

Low-density parity-check (LDPC) codes were first introduced by Robert G. Gallager in his Ph.D. dissertation at MIT in 1962 [1]. LDPC codes are a class of linear error-correcting codes characterized by their sparse parity-check matrices, enabling efficient iterative decoding techniques capable of approaching Shannon's theoretical capacity limits for Gaussian channels [2], [3].

Despite their linear decoding complexity, the computational demands of LDPC codes were initially too high for practical implementation, resulting in limited attention for several decades. Interest in LDPC codes resurged following the discovery of Turbo codes in 1993 [4], which demonstrated that iterative decoding techniques could achieve near-Shannon-limit performance with manageable complexity on modern hardware. This marked a significant departure from classical approaches, which focused primarily on structured codes with large minimum distances. At this time, classical code designs were also facing substantial challenges in identifying codes that simultaneously offered large minimum distances and high coding rates. In contrast, LDPC codes leverage iterative belief propagation algorithms to

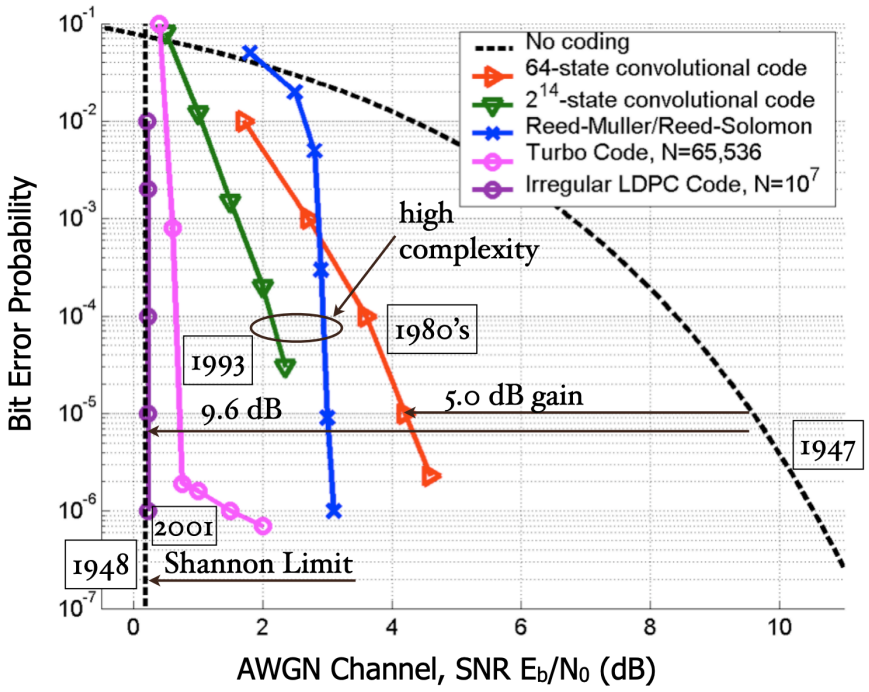


Figure 1: Performance of different large block-length codes over AWGN channels, Rate $R = 1/2$ [5].

progressively refine decoding decisions, significantly improving performance without relying heavily on code structure or minimum distance constraints.

Convolutional and Turbo codes dominated wireless systems before the widespread adoption of LDPC codes in the 2000s. However, in 2001, MacKay's implementation [3] was able to achieve the Shannon limit, which is the theoretical minimum $\frac{E_b}{N_0}$ possible to achieve no errors and capacity [6]. LDPC codes have since become integral to numerous modern communication standards, including deep-space communication links, Digital Video Broadcast Satellite (DVB-S2), WiFi, and current and future wireless standards like 5G and 6G.

2.2 Encoding

In this section, we will cover the theoretical basis behind the encoding scheme for LDPC codes. A LDPC code is a linear block code with a sparse parity check matrix. The parity-check matrix $\mathbf{H}$ is considered sparse if the fraction of nonzero entries is small compared to the total number of elements in $\mathbf{H}$. The sparsity of $\mathbf{H}$ allows the code to be efficiently decoded.

In particular, a (n, m, p)-regular LDPC code has block length n, where each column of its parity check $\mathbf{H}$ contains a small number of m 1's (column weight) and each row of $\mathbf{H}$ contains a small number of p 1's (row weight). On the other hand, an irregular LDPC code is a code for which the number of ones in each row and column, respectively, is not equal. In practice, irregular LDPC codes are used much more often than regular ones as they provide better code performance. [1]

The parity check matrices can be constructed with a sub-matrix containing one ones per column and p ones per row and concatenating copies of the sub-matrix with randomly permuted columns [1]. In practice there is some built in structure to the parity check matrix to allow reuse and control short-cycles.

[1] When designing irregular LDPC codes, one must consider that they are prone to cycles in the Tanner graph, as the nodes connected with many parity check nodes can overlap with other check nodes, which can reduce the effectiveness of the decoder particularly with the iterative decoding algorithms [7]

This is accomplished by designing a protograph and subsequently lifting this into a full-size LDPC code. This approach allows reuse of a common base parity check matrix without having to redesign a new one for each desired code rate, allowing for efficient rate matching. 5G radios adopt this method, which is briefly mentioned in Section 3.

For an example, consider this irregular LDPC code and its corresponding parity check matrix $\mathbf{H}$:

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{pmatrix} \tag{1}$$

From our message m, in order to generate a codeword $\mathbf{x}$, we need to multiply the message by the generator matrix $\mathbf{G}$ to get $\mathbf{x} = m\mathbf{G}$. The generator matrix can be derived by performing row reduction and making $\mathbf{H} = [\mathbf{I_{n-k}}|\mathbf{P}]$. Then, $\mathbf{G} = [\mathbf{P^T}|\mathbf{I_k}]$.

For Matrix 1, we can reduce $\mathbf{H}$ to

$$\mathbf{H} = [\mathbf{I_{n-k}}|\mathbf{P}] = \left(\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{array}\right)$$

Then,

$$\mathbf{P} = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}.$$

Thus,

$$\mathbf{G} = [\mathbf{P^T}|\mathbf{I_k}] = \left(\begin{array}{cccc|cccc} 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{array}\right).$$

Deriving $\mathbf{G}$ from $\mathbf{H}$ through Gaussian elimination is computationally expensive, especially for large block lengths. In practice, such as in WiFi and 5G NR standards, the parity check matrix $\mathbf{H}$ is often defined and fixed. This allows us to precalculate and store $\mathbf{G}$ to allow efficient encoding.

The LDPC encoding is relatively straightforward, but decoding is a harder problem. However, the low density structure of the parity check matrix $\mathbf{H}$ allows for efficient low-complexity decoding, which we will discuss in Section 2.3.

Tanner Graph Representation

In addition to the matrix form of the parity check matrix, we can also represent it with a bipartite graph introduced by Tanner in 2003 [8], which will be the foundation for the development of our decoding algorithms in Section 2.3. This graph holds the constraints imposed by our parity check matrix and can be constructed as such. One set of nodes corresponds to the columns and we call them variable nodes (the message bits), denoted v_i where i corresponds to the column. The other set corresponds rows and we call them check nodes (parity checks), denoted c_j where j corresponds to the row. An edge connects check node c_j to variable node v_i if $H_{i,j} = 1$. These connections represent the parity check equations, since summing all the variable nodes connected to the check nodes are an equivalent function.

The parity check matrix **H** introduced in Matrix 1 can by expressed as a Tanner graph in the following way:

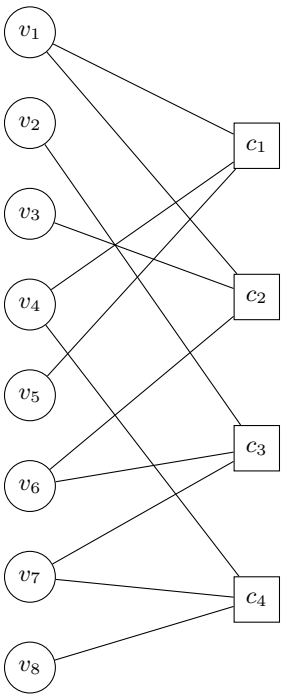


Figure 2: Tanner graph corresponding to Matrix

We interpret each row as a check node, and each column as a variable node. In total, there are 8 variable nodes (v_1, v_2, v_3, v_4, v_5, v_6 v_7 v_8) and 4 check nodes (c_1, c_2, c_3 c_4). The codewords $\mathbf{x}$ satisfying $\mathbf{Hx} = \mathbf{0}$ must make each row (parity check) sum to 0 mod 2:

$$\begin{aligned} c_1: &\quad v_1 + v_4 + v_5 \equiv 0 \pmod 2, \\ c_2: &\quad v_1 + v_3 + v_6 \equiv 0 \pmod 2. \\ c_3: &\quad v_2 + v_6 + v_7 \equiv 0 \pmod 2. \\ c_4: &\quad v_4 + v_7 + v_8 \equiv 0 \pmod 2. \end{aligned}$$

2.3 Decoding

LDPC codes are typically decoded using iterative message-passing algorithms on the associated Tanner graph, described in section 2.2. In this section, we first demonstrate how Tanner graphs enable iterative decoding using an example over the BEC. Then, we review two popular iterative approaches—the

sum-product (or belief propagation) algorithm and the min-sum algorithm—focusing on their operation over the additive white Gaussian noise (AWGN) channel.

2.3.1 Tanner Graphs Can be Used for Decoding

Now, we highlight how errors can be corrected via iterative decoding on a (Tanner) graph representation of an LDPC code. We begin by demonstrating how the Tanner graph structure allows us to detect and correct bit flips iteratively , in a simple setting over the binary erasure channel (BEC).

Problem Statement

We consider the transmission of a codeword $\mathbf{x} \in \{0,1\}^n$ through a Binary Erasure Channel (BEC) with erasure probability p. That is, each bit x_i is independently erased with probability p, resulting in the received vector $\mathbf{y} \in \{0,1,?\}^n$. Our goal is to recover $\mathbf{x}$ from $\mathbf{y}$ using the known parity-check constraints defined by the code's parity-check matrix $\mathbf{H}$. In this example, we will use the matrix from Figure 2 from before.

Algorithm

The decoding process follows this iterative algorithm:

1. Each variable node sends its value y_i to the check nodes they are connected to.
2. **Check Node Update:** Each check node computes the parity of the known bits connected to it:

$$P_j = \sum_{i|H_{ji}=1, y_i \neq ?} y_i \mod 2.$$

3. **Variable Node Update:** If a check node is connected to exactly one erased variable node, that node can be determined as $y_i = P_j$. The resolved bit y_i is communicated to the variable node from the check node.
4. If all erased bits are resolved, return the decoded codeword $\mathbf{x}$. If an iteration completes without resolving any new bits, return failure since no more symbols can be resolved. Otherwise return to Step 1.

Example:

Suppose we transmit the following codeword:

$$\mathbf{x} = (1,1,1,1,0,0,1,0).$$

However, due to erasures in the channel, we receive:

$$\mathbf{y} = (1,?,1,?,0,?,?,0),$$

where "?" represents an erased bit.

Decoding Process

Iteration 1:

1. $\mathbf{y}$ sends all its values to the corresponding check nodes
2. - Check Node 1: We compute $P_1 = 1 \oplus 0 = 1$ and note we are connected to one erased variable node, v_4.

- Check Node 2: We compute $P_2 = 1 \oplus 1 = 0$ and note we are connected to one erased variable node, v_6.
- Check Node 3: We compute $P_3 =?$ and note we are connected to three erased variable nodes, v_2, v_6 and v_7.
- Check Node 4: We compute $P_4 = 0$ and note we are connected to two erased variable nodes, v_4 and v_7.

3. Since c_1 and c_2 are connected to one erased variable node, we send P_1 to v_4 and P_2 to v_6
4. Then, $v_4 = P_1$ and $v_6 = P_2$.
5. After Iteration 1, the partially recovered vector is:

$$\mathbf{y} = (1, ?, 1, 1, 0, 0, ?, 0).$$

Since we resolved new bits and there are still unresolved bits, we must run another iteration.

Iteration 2:

1. $\mathbf{y}$ sends all its values to the corresponding check nodes
2. - Check Node 1: We compute $P_1 = 1 \oplus 1 \oplus 0 = 0$ and note we are not connected to any erased variable nodes.
   - Check Node 2: We compute $P_2 = 1 \oplus 1 \oplus 0 = 0$ and note we are not connected to any erased variable nodes.
   - Check Node 3: We compute $P_3 = 0$ and note we are connected to two erased variable nodes, v_2 and v_7.
   - Check Node 4: We compute $P_4 = 1 \oplus 0 = 1$ and note we are connected to one erased variable node, v_7.
3. Since c_4 is connected to one erased variable node, we send P_4 to v_7
4. Then, $v_7 = P_4$.
5. After Iteration 2, the partially recovered vector is:

$$\mathbf{y} = (1, ?, 1, 1, 0, 0, 1, 0).$$

Since we resolved new bits and there are still unresolved bits, we must run another iteration.

Iteration 3:

1. $\mathbf{y}$ sends all its values to the corresponding check nodes
2. - Check Node 1: We compute $P_1 = 1 \oplus 1 \oplus 0 = 0$ and note we are not connected to any erased variable nodes.
   - Check Node 2: We compute $P_2 = 1 \oplus 1 \oplus 0 = 0$ and note we are not connected to any erased variable nodes.
   - Check Node 3: We compute $P_3 = 0 \oplus 1 = 1$ and note we are connected to one erased variable node, v_2.

- Check Node 4: We compute $P_4 = 1 \oplus 1 \oplus 0 = 0$ and note we are not connected to any erased variable nodes.

3. Since c_3 is connected to one erased variable node, we send P_3 to v_2
4. Then, $v_2 = P_3$.
5. After Iteration 3, the partially recovered vector is:

$$\mathbf{y} = (1, 1, 1, 1, 0, 0, 1, 0).$$

Since there are no more erased bits, we return $\mathbf{y}$ and one can see that we recovered the sent codeword $\mathbf{x}$.

Although this is a simplified demonstration, it illustrates how we can iteratively pass messages between nodes in a Tanner graphs to gain information about our transmitted message and decode it. This structure is referred to as an **iterative message passing decoding scheme**. Intuitively, the messages can be viewed as serving these functions:

- **Variable-to-check messages** convey the "belief" of each variable node about its bit values.
- **Check-to-variable messages** combine all incoming messages to a parity-check node and inform variable nodes about the consistency with the parity.

After introducing this basic idea for the BEC, we will turn our attention to the weidely adopted noise model for communication systems, the AWGN channel, where log-likelihood ratios (LLRs) become central to message passing. We then present two prominent iterative decoding algorithms—the sum-product algorithm and the min-sum approximation.

2.3.2 Sum Product and Min Sum Algorithms

AWGN Channel Model and LLRs

Consider transmitting an n-bit LDPC codeword $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ over an AWGN channel using BPSK modulation, which is depicted in Figure 3. In BPSK, each bit $x_i \in \{0, 1\}$ is mapped to a symbol $s_i \in \{+1, -1\}$ via $s_i = (-1)^{x_i}$. We assume that each bit x_i occurs with equal prior probability $p = 1/2$. These symbols pass through an AWGN channel with noise variance σ^2, yielding received symbols $y_i = s_i + n_i$.

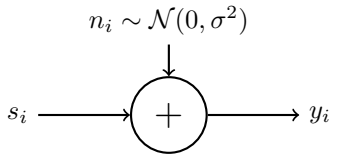


Figure 3: AWGN Channel Model

To facilitate soft decoding, we compute the *log-likelihood ratio* (LLR) for each bit:

$$L_i = \ln\left(\frac{P(x_i = 0 \mid y_i)}{P(x_i = 1 \mid y_i)}\right).$$

When the LLR is positive, this means it was more likely $x_i = 0$ given y_i and when the LLR is negative, it was more likely $x_i = 1$ given y_i. For an AWGN channel with BPSK, it is common to approximate

$$L_i \approx \frac{2}{\sigma^2}\, y_i,$$

assuming equally likely 0 and 1 bits a priori.

Sum-Product Algorithm

The sum-product algorithm [9] is used for LDPC decoding under the AWGN channel and its messages are derived from the LLRs of our received messages.

Let $m_{v\to c}^{(t)}$ denote the message from variable node v to check node c at iteration t, and $m_{c\to v}^{(t)}$ the message from check node c to variable node v at iteration t. Denote by $\mathcal{N}(v)$ the set of check nodes connected to variable node v, and by $\mathcal{N}(c)$ the set of variable nodes connected to check node c. The algorithm follows:

1. Initialize the L_v at each variable node where L_v is the channel LLR for variable node v.
2. **Variable node update:**
$$m_{v\to c}^{(t)} = L_v + \sum_{c'\in\mathcal{N}(v)\setminus c} m_{c'\to v}^{(t-1)},$$
3. **Check node update:**
$$m_{c\to v}^{(t)} = 2\tanh^{-1}\left(\prod_{v'\in\mathcal{N}(c)\setminus v}\tanh\Big(\frac{m_{v'\to c}^{(t)}}{2}\Big)\right).$$
4. After a fixed number of iterations or upon convergence, a hard decision is made for each variable node:
$$\hat{x}_v = \begin{cases} 0, & \text{if } L_v + \sum_{c'\in\mathcal{N}(v)} m_{c'\to v}^{(t)} \geq 0, \\ 1, & \text{otherwise.} \end{cases}$$
5. After a hard decision, if the parity does not pass, return to Step 2 for another iteration of message passing.

The name sum-proudct algorithm arises from the fact that at the variable nodes, we sum the messages from each check node and then subsequently take a product of the messages from the variable nodes at the check nodes.

2.3.3 The Min-Sum Algorithm

While the sum-product algorithm is near-optimal in performance, it can be computationally expensive to implement due to the tanh and $\tanh^{-1}$ operations. In the min-sum algorithm [10], the check node update replaces the tanh combination with a simpler rule:

$$m_{c\to v}^{(t)} \approx \left(\prod_{v'\in\mathcal{N}(c)\setminus v}\operatorname{sign}\big(m_{v'\to c}^{(t)}\big)\right)\cdot\min_{v'\in\mathcal{N}(c)\setminus v}\big|m_{v'\to c}^{(t)}\big|.$$

Variable node updates remain the same as in sum-product. This approximates the minimum magnitude and sign behavior of the sum-product formula, hence the name min-sum algorithm.

For many LDPC codes, the min-sum algorithm performs close to the sum-product algorithm and often requires fewer hardware resources or less decoding time. However, it can introduce a small performance penalty, typically mitigated by scaling or offset adjustments in practical implementations [11]. The choice between sum-product and min-sum (and their variations) often depends on implementation complexity constraints and the desired error-rate performance.

3 Case Study: LDPC in 5G

In this section we highlight an application where LDPC codes have found widespread success: 5G communications. Focusing mainly on the decoding side, we highlight the important features of an LDPC code that make them desirable from a communications perspective, and use physical layer simulations to explore LDPC coded system performance over an AWGN channel using libraries available from NVIDIA Sionna [12]. The code repository for these channel simulations can be found on our GitHub [13]. From these simulations, it is clear what regimes LDPC codes excel in and the advantage they have over other codes for high-data rate communication systems.

With each new cellular generation standard, there is an increased push for higher throughput and capacity while maintaining reliability, low latency, and feasible hardware implementations. Of primary interest for coding schemes used in communication systems are implementation complexity (which includes silicon area and power overhead) and the coding gain, which is measured as the reduction in the power required to achieve a certain bit-error-rate (BER) [14]. LDPC codes have been adopted at scale in 5G cellular systems as they offer some of the best known performance of channel codes along these evaluation axes. Prior generations (3G and 4G LTE) were making use of Turbo and Convolutional codes, however LDPC codes allow for higher rates and greater hardware parallelism in modern semiconductor processes, thus providing a significant performance gap over legacy codes and improved real-time performance.

3.1 Increasing Block Length Size

A desireable feature of LDPC codes is their capacity approaching behavior for large block lengths, making them ideal for the 5G data channel where they support up to $k = 8448$ information bits per codeword for various code rates [15]. The capacity C of a band-limited AWGN channel is given by the well-known Shannon-Hartley theorem:

$$\frac{C}{W} = \log_2(1 + \frac{C}{W} \cdot \frac{E_b}{N_0}) \quad [\text{bit/s/Hz}] \tag{2}$$

where W is the channel bandwidth, E_b is the energy per bit, and N_0 is the noise power spectral density (the noise variance). This is often reformulated as the minimum energy per bit required for error-free transmission:

$$\left(\frac{E_b}{N_0}\right)_{min} = \frac{2^{C/W} - 1}{C/W} = \frac{2^{R \cdot m} - 1}{R \cdot m} \tag{3}$$

where C/W is the spectral efficiency, m is the number of bits per symbol (which is modulation dependent), and R is the code rate. It has been shown that LDPC codes approach capacity to within 0.0045 dB for an $R = 1/2$ code [2],[16]. We simulate the behavior of the BER vs. E_b/N_0 for increasing code sizes below, where we can see the capacity approaching behavior of LDPC codes for increasing block size in Fig. 4. Practically, the block-size and decoding technique (min-sum) lead systems to operate within 1-2 dB of the Shannon limit, which is 0 dB for a $R = 1/2$ code using QPSK ($m = 2$) modulation.

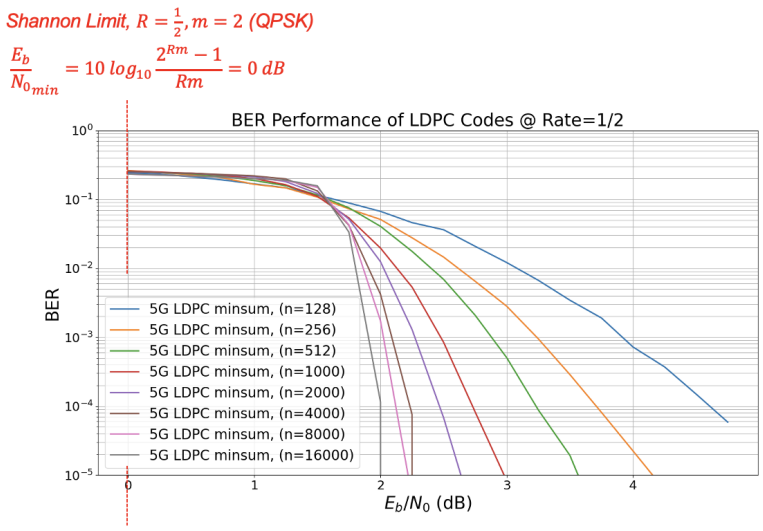


Figure 4: BER performance of LDPC codes using min-sum decoding (20 iterations) as a function of block length, using QPSK Modulation on an AWGN channel.

3.2 Comparison to Legacy Schemes

Legacy cellular systems rely on sequential coding techniques, namely Turbo and Convolutional codes. In 4G systems, Turbo codes are used in the data channel, while 3G and 2G systems use convolutional codes. LDPC codes are advantageous over legacy schemes in a multitude of ways, allowing for superior BER performance at lower E_b/N_0 (coding gain) and throughput. In Fig. 5, we compare the BER performance of LDPC codes using a 20 iteration min-sum decoder with Turbo codes (using BCJR decoding) and Convolutional codes (using Viterbi decoding).

It can be observed that both LDPC codes and Turbo codes exhibit superior BER performance and power efficiency compared to Convolutional codes. Turbo codes tend to outperform LDPC codes at small to moderate block lengths due to their stronger minimum distance properties at these lengths. However, as the block length increases, LDPC codes surpass Turbo codes in BER performance, approaching the Shannon limit more closely.

A key observation is the sharp threshold behavior of LDPC codes, where BER rapidly declines once a critical E_b/N_0 is reached, forming a steeper waterfall than the other two codes. This phenomenon is explained by density evolution, which analyzes the iterative message-passing process on the Tanner graph. As the block length increases, density evolution ensures that the log-likelihood ratio (LLR) messages become progressively more reliable, leading to near-perfect decoding beyond a certain noise threshold. This threshold decreases with increasing block length, making LDPC codes asymptotically capacity-achieving [17].

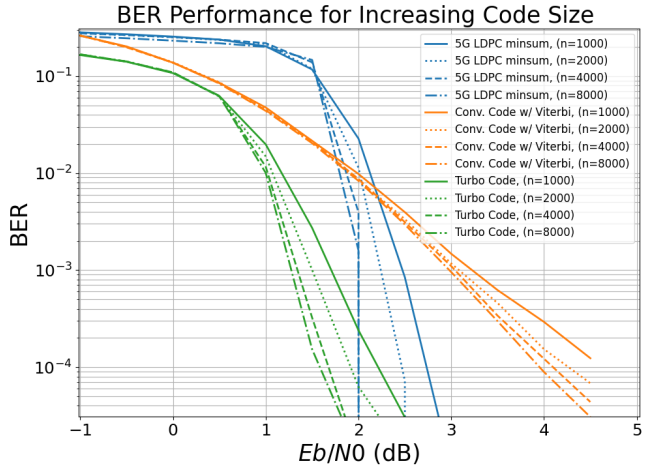


Figure 5: BER performance of LDPC, Convolutional, and Turbo codes as a function of block length. QPSK Modulation on an AWGN channel.

Given the similarity of LDPC codes and Turbo codes over realistic block length sizes, one still may wonder if the performance gains of LDPC codes over Turbo codes require a new error correction scheme in modern wireless systems when only considering this evaluation criterion. This leads us to the next key-performance-indicator, which is decoding throughput.

3.3 Complexity & Throughput

In order to align with the increasing throughput and lower latency requirements for each new cellular generation, the error correcting schemes should perform well in this regard and not bottleneck communication speeds. As depicted in Fig. 6, we simulate the decoder throughput to get an empirical proxy for the computational complexity given the codes and block lengths considered in Section 3.2. This is carried out by measuring the time taken to decode a set of transmitted bits k for given block lengths at a fixed E_b/N_0 value. This time is then averaged over 20 repetitions to obtain the simulated throughput at each block length.

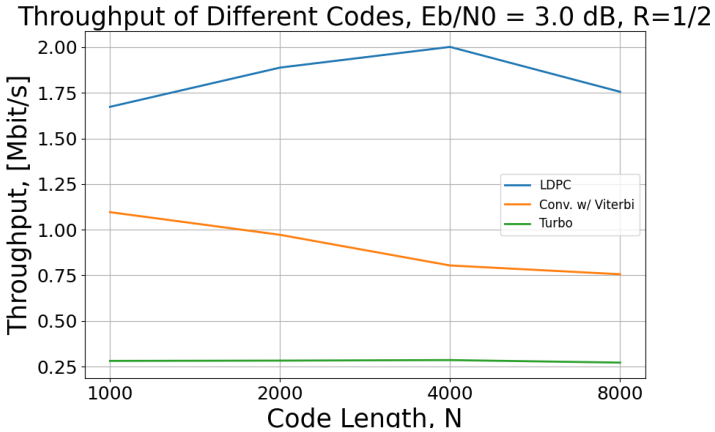


Figure 6: Decoding throughput for $R = 1/2$ codes as a function of block length for fixed $E_b/N_0 = 3$ dB on an AWGN channel, QPSK modulation.

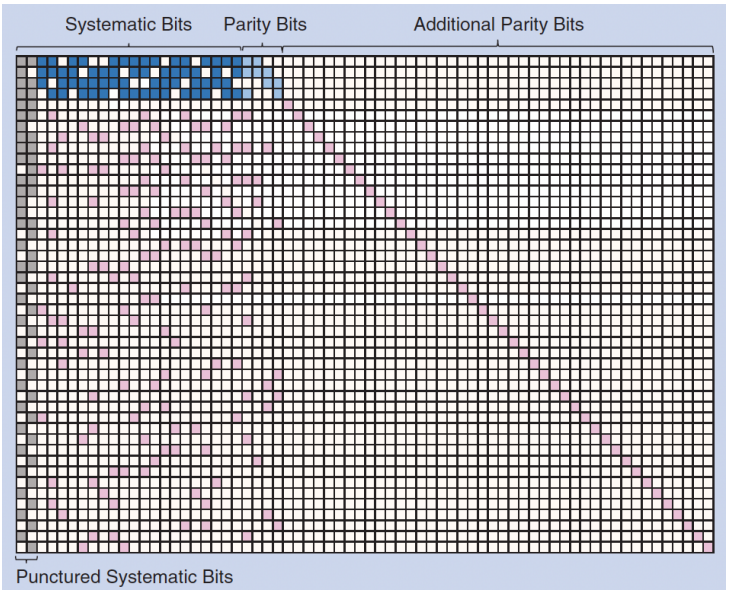


Figure 7: A 5G new-radio LDPC base matrix (BG1) [18]. Each square corresponds to 1 element in the base matrix or a ZxZ sub-matrix of the parity check matrix. The base matrix enables the generation of codewords at various rates by selectively incorporating parity bits or utilizing a subset of the full parity-check matrix.

We observe that LDPC decoding has a significant throughput advantage over Convolutional codes, and especially Turbo codes. LDPC codes can do this because decoding techniques like min-sum decoding carry out a large number of small **independent** check node computations which can be effectively parallelized in hardware with linear complexity in the block length, as opposed to the legacy sequential techniques operating on a Trellis which have higher computational complexity depending exponentially on the number of states in the graph. Additionally, the LDPC check node computations do not have to be strictly time synchronized, allowing for easier hardware implementation.

3.4 LDPC Rate Adaptation

In practical wireless systems, the channel is dynamic and highly variable with space and time. Therefore, the SNR (and thus E_b/N_0) can experience significant fluctuations. As such, coding schemes should be able to adapt the rates in order to get the most out of a dynamic channel. At low SNRs, low-rate codes should be used to focus on transmission reliability and maintain an established link. At high SNRs, the code rate can be reliably increased in order to focus on increasing throughput. This is referred to as *rate matching*. LDPC codes are highly adaptable for rate matching due to their structured parity-check matrices (Fig. 7), which allow flexible bit selection, puncturing, and repetition without significantly degrading performance.

It is worth mentioning some aspects of 5G LDPC encoding and how this approach enables rate-matching. Firstly, there are two LDPC **base graphs** - BG1 and BG2. Each base graph describes a structured protograph through which different block size and code rates can be synthesized, allowing for a wide range of code lengths and rates through *lifting factors* and rate matching. BG1 is typically used for larger block sizes and code rates, while BG2 for smaller block sizes and code rates. Because LDPC codes rely on structured base graphs, they do not require a separate parity-check matrix for each rate. Instead, rate adaptation is achieved through parity bit puncturing or repetition, allowing

dynamic code rate adjustments based on real-time channel state information [14]. Thus, LDPC codes can effectively adapt its rate according to dynamic channel conditions without having to significantly redefine its underlying parity check structure.

Link quality-of-service is managed through the MAC (Medium Access Control) layer through Incremental Redundancy Hybrid Automatic Repeat Requests (IR-HARQ), whereby retransmissions provide additional parity bits, progressively enhancing the decoder's ability to correct errors without resending the same information [19]. This approach is impractical for Turbo and Convolutional codes since their Trellis-based decoding requires systematic and parity bits to be used together, making incremental redundancy less efficient. Unlike LDPC codes, which can dynamically adjust redundancy using a single structured parity-check matrix, Turbo and Convolutional codes rely on fixed trellis structures, requiring predefined puncturing patterns that limit flexibility in IR-HARQ.

As shown in Fig. 8, we simulate the BER and BLER (block-error rate) for different rate LDPC codes using a fixed number of message bits over an AWGN channel. In this scenario, the modulation scheme is adapted to allocate more or less bits-per-symbol depending on the SNR, along with the code rate.

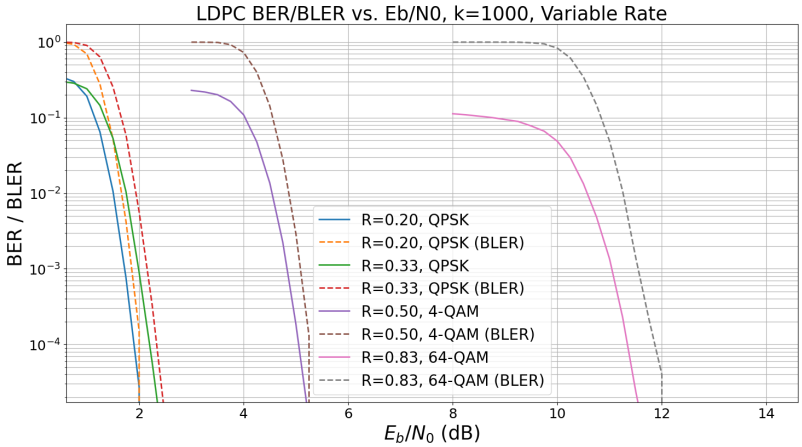


Figure 8: BER performance of LDPC codes for different rates and modulation schemes.

As observed from the waterfall curves, high rate and low order modulation schemes have lower required E_b/N_0 to achieve a target BER, as contrasted from the higher rate and higher order schemes. Plots like this are used to set operating conditions for the code rates and modulation schemes that devices use in real-time[2]. LDPC schemes are highly flexible in this regard, and have exceptionally good high rate performance (low BER floors) as the sparisty of $\mathbf{H}$ increases.

4 Conclusions & Future Directions

From its origins in Gallager's 1962 dissertation, LDPC codes have emerged as a cornerstone of wireless communication systems due to their near-capacity performance, low complexity schemes and its flexibility in construction. Through our simulations, we demonstrated these traits and why they work remarkably well in communication systems, especially for 5G systems that have long blocklengths. However, there remains many areas of exploration to improve LDPC codes.

[2]This is known as an MCS (modulation and coding scheme) index, and is often specified as a look-up table in a modem's memory to determine the optimal information rates depending on the current estimated channel state information [15].

We introduced two iterative message passing decoding algorithms, however, there are many more algorithmic complexity reductions that can be made to achieve different points on the Pareto front of the tradeoff curve between complexity and performance ([20],[21], [22]).

Additionally, we highlighted some key considerations to keep in mind from a system design perspective. Specifically, we considered the performance of 5G LDPC codes in comparison to legacy cellular schemes when prioritizing different design goals: E_b/N_0 efficiency, throughput/data-rate, and adaptability to channel conditions. LDPC codes excel in all of these domains.

Moreover, people have begun to explore the role that AI can play in the decoding of LDPC codes. There has been a multitude of publications regarding this ([23], [24],[25]). In addition, Sionna has these capabilities to explore AI assisted LDPC decoding, so running these simulations could be an interesting space of self exploration and extension on the work already done here.

With all this in mind, LDPC code constructions have to be made with the characteristics of the system in mind. LDPC codes with the right constructions have become quintessential in modern wireless communications and will be crucial to the next revolution of wireless communications, including 6G and beyond.

References

[1] R. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, 1962.

[2] T. Richardson and R. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 599–618, 2001.

[3] D. MacKay and R. Neal, "Near shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 32, pp. 1645–1646, 1996.

[4] T. Richardson and R. Urbanke, "The renaissance of gallager's low-density parity-check codes," *IEEE Communications Magazine*, vol. 41, no. 8, pp. 126–131, 2003.

[5] B. Kurkoski, "Low-density parity-check (ldpc) codes lecture notes."

[6] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[7] T. Richardson and R. Urbanke, *Modern Coding Theory*. USA: Cambridge University Press, 2008.

[8] R. Tanner, "A recursive approach to low complexity codes," *IEEE Transactions on Information Theory*, vol. 27, no. 5, pp. 533–547, 1981.

[9] D. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Transactions on Information Theory*, vol. 45, no. 2, pp. 399–431, 1999.

[10] N. Wiberg, H.-A. Loeliger, and R. Kotter, "Codes and iterative decoding on general graphs," in *Proceedings of 1995 IEEE International Symposium on Information Theory*, 1995, pp. 468–.

[11] J. Chen, A. Dholakia, E. Eleftheriou, M. Fossorier, and X.-Y. Hu, "Reduced-complexity decoding of ldpc codes," *IEEE Transactions on Communications*, vol. 53, no. 8, pp. 1288–1299, 2005.

[12] N. Samuel, C. Häger, S. Cammerer, A. Elkelesh, H. D. Pfister, and J. Hoydis, "Sionna: An open-source library for next-generation physical layer research," *arXiv preprint arXiv:2203.11854*, 2022.

[13] E. M. Austin Rothschild, "Ldpc codes - implementation and simulations," 2024. [Online]. Available: https://github.com/austinrosh/ldpc-codes

[14] T. Richardson and S. Kudekar, "Design of low-density parity check codes for 5g new radio," *IEEE Communications Magazine*, vol. 56, no. 3, pp. 28–34, 2018.

[15] 3rd Generation Partnership Project (3GPP), "3GPP TS 38.212: Multiplexing and Channel Coding for NR," 3rd Generation Partnership Project (3GPP), Tech. Rep. V17.7.0, March 2024. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/38_series/38.212/

[16] S.-Y. Chung, G. Forney, T. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 db of the shannon limit," *IEEE Communications Letters*, vol. 5, no. 2, pp. 58–60, 2001.

[17] D. J. C. MacKay, *Information Theory, Inference & Learning Algorithms*. USA: Cambridge University Press, 2002.

[18] D. Hui, S. Sandberg, Y. Blankenship, M. Andersson, and L. Grosjean, "Channel coding in 5g new radio: A tutorial overview and performance comparison with 4g lte," *IEEE Vehicular Technology Magazine*, vol. 13, no. 4, pp. 60–69, 2018.

[19] J. H. Bae, A. Abotabl, H.-P. Lin, K.-B. Song, and J. Lee, "An overview of channel coding for 5g nr cellular communications," *APSIPA Transactions on Signal and Information Processing*, vol. 8, p. e17, 2019.

[20] I. B. Djordjevic, L. Xu, and T. Wang, "On the reduced-complexity of ldpc decoders for beyond 400 gb/s serial optical transmission," in *Asia Communications and Photonics Conference and Exhibition*, 2010, pp. 566–567.

[21] H. Li, J. Guo, C. Guo, and D. Wang, "A low-complexity min-sum decoding algorithm for ldpc codes," in *2017 IEEE 17th International Conference on Communication Technology (ICCT)*, 2017, pp. 102–105.

[22] M. Stark, L. Wang, G. Bauch, and R. D. Wesel, "Decoding rate-compatible 5g-ldpc codes with coarse quantization using the information bottleneck method," *IEEE Open Journal of the Communications Society*, vol. 1, pp. 646–660, 2020.

[23] C. Zhang, Y.-L. Ueng, C. Studer, and A. Burg, "Artificial intelligence for 5g and beyond 5g: Implementations, algorithms, and optimizations," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 10, no. 2, pp. 149–163, 2020.

[24] L. Wang, C. Terrill, D. Divsalar, and R. D. Wesel, "Ldpc decoding with degree-specific neural message weights and rcq decoding," *IEEE Transactions on Communications*, vol. 72, no. 4, pp. 1912–1924, 2024.

[25] J. Gao, B. Zhang, B. Wang, and Y. Liu, "A ldpc decoding algorithm based on convolutional neural network," in *2022 9th International Conference on Dependable Systems and Their Applications (DSA)*, 2022, pp. 1065–1066.

Theory & Implementation of LDPC Codes for Modern Communications

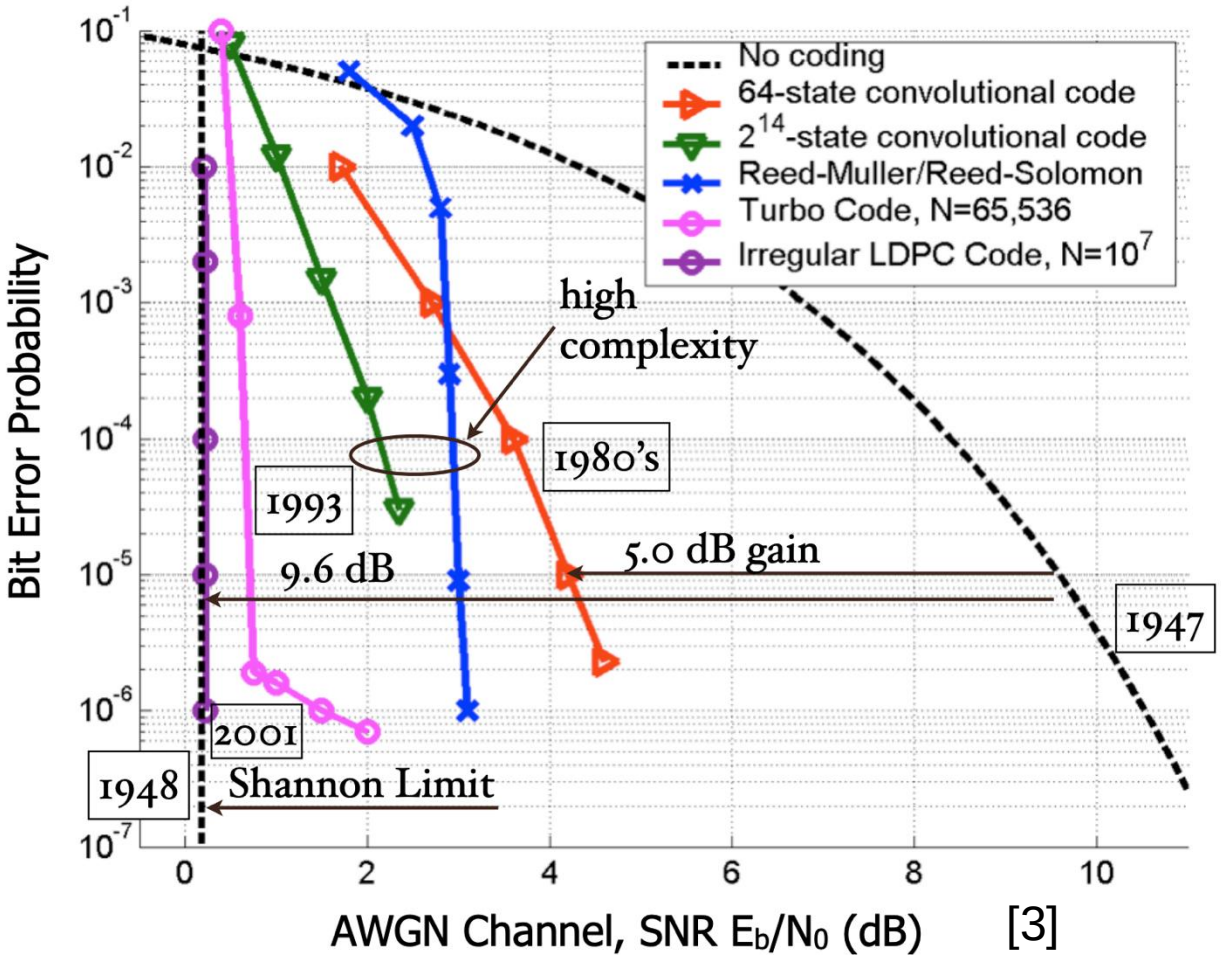
Austin Rothschild, Eugene Min, Kyle Huang

March 12, 2025

Stanford | ENGINEERING
Electrical Engineering

LDPC Codes: History/Overview

- LDPC: **L**ow **D**ensity **P**arity **C**heck
- First discovered by Robert Gallager in the early 1960s [1]
  - Early ML decoders were impractically complex to implement
- 1990s – Iterative decoding revolution started with Turbo codes, LDPC codes now practical [2]
- 2020s – LDPC codes are one of the main ECCs
  - Wi-Fi (since 802.11n)
  - 5G
  - Satellite Communications
  - High-speed fibers

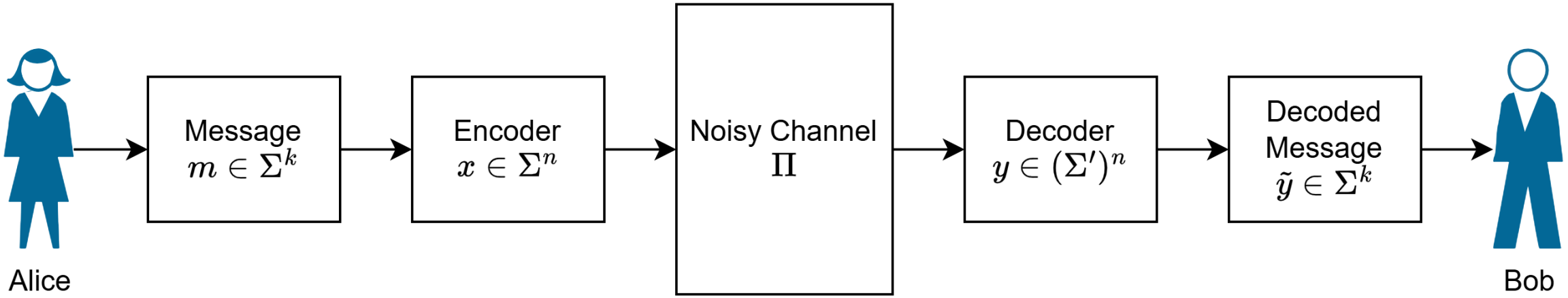


[3]

Stanford | ENGINEERING
Electrical Engineering

Communication Channel

- Message **m**, codeword **x**
- Message is encoded and transmitted through a noisy channel



Encoding

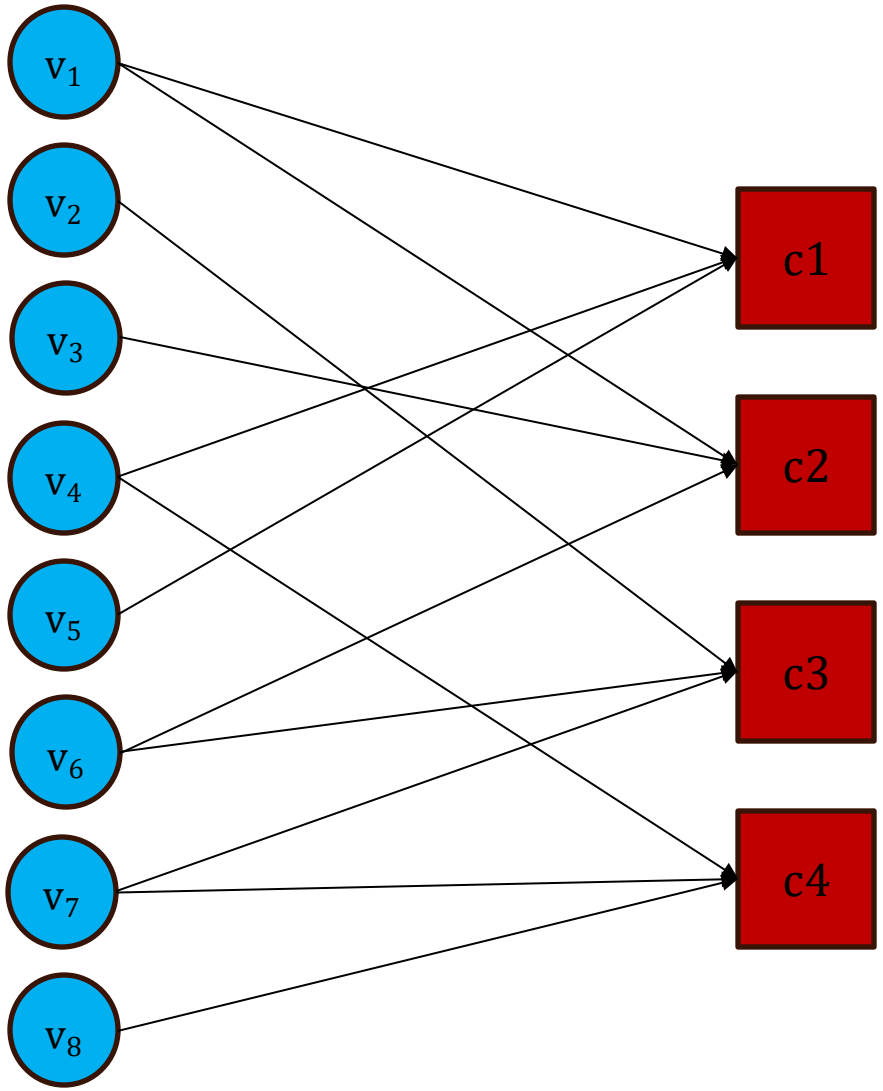
- LDPC codes are **linear block codes** that are defined by their sparse Channel matrix $\mathbf{H}$
- Key: **Sparsity**
  - Few 1's compared to total elements
  - Necessary for efficient decoding
- Generator matrix $\mathbf{G}$ is derived from $\mathbf{H}$
- Codeword $\mathbf{c}$ is generated from $\mathbf{c} = \mathbf{mG}$
- An 8x4 example:

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{pmatrix} \qquad \mathbf{G} = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Stanford | ENGINEERING
Electrical Engineering

Tanner Graphs are used to decode LDPC codes

Parity Check as a Tanner Graph



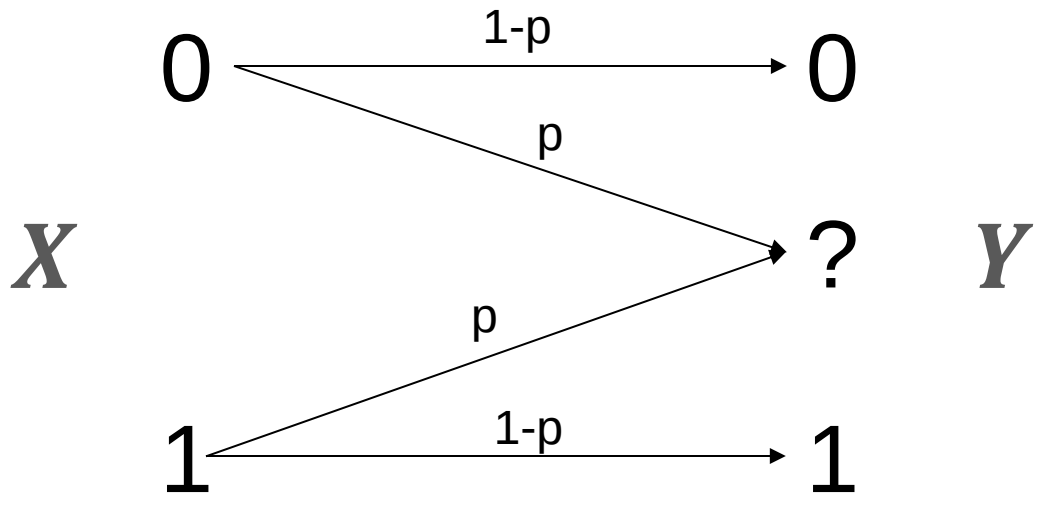
- Tanner graphs are a bipartite graph that contains information about constraints [4]
- In our case, we can use the parity check matrix as our constraints since any codeword $\mathbf{x}$ must satisfy $\mathbf{Hx} = 0$
- We interpret each row as a check node, and each column as a variable node

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Stanford | ENGINEERING
Electrical Engineering

BEC Example

Problem: Consider the transmission of a codeword $x \in \{0,1\}^n$ through a Binary Erasure Channel (BEC) with erasure probability p. Our goal is to recover x from y using the known parity-check constraints defined by the code's parity-check matrix $\mathbf{H}$.

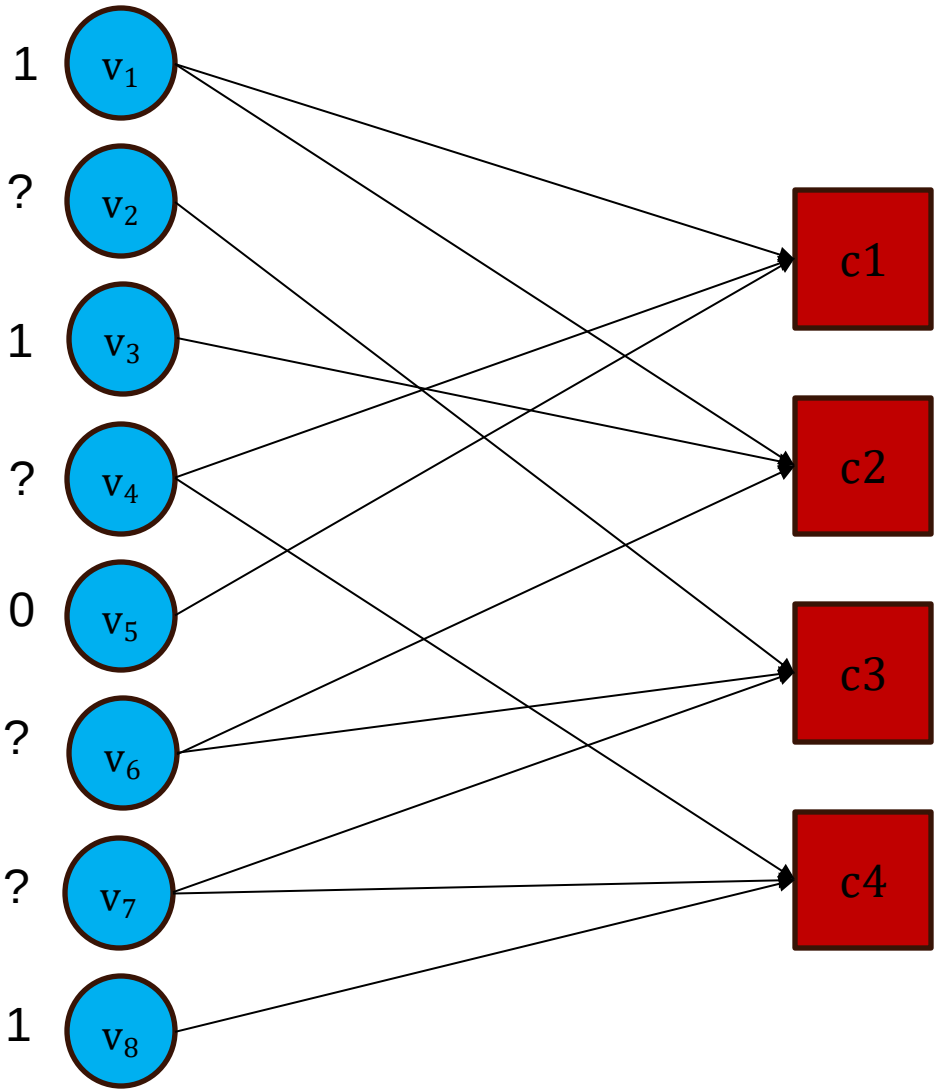


BEC Example

Algorithm:

1. Variable nodes send their value to the check nodes
2. Check nodes compute their parity from received values excluding ‘?’
3. If a check node is connected to one variable node with a ‘?’, send the parity back to the variable node
4. Variable nodes set to ‘?’, assign themselves to the message sent by the check node
5. If y has no more erased bits, return y. If no variables nodes changed their values, return failure. Otherwise return to step 1.

BEC Example

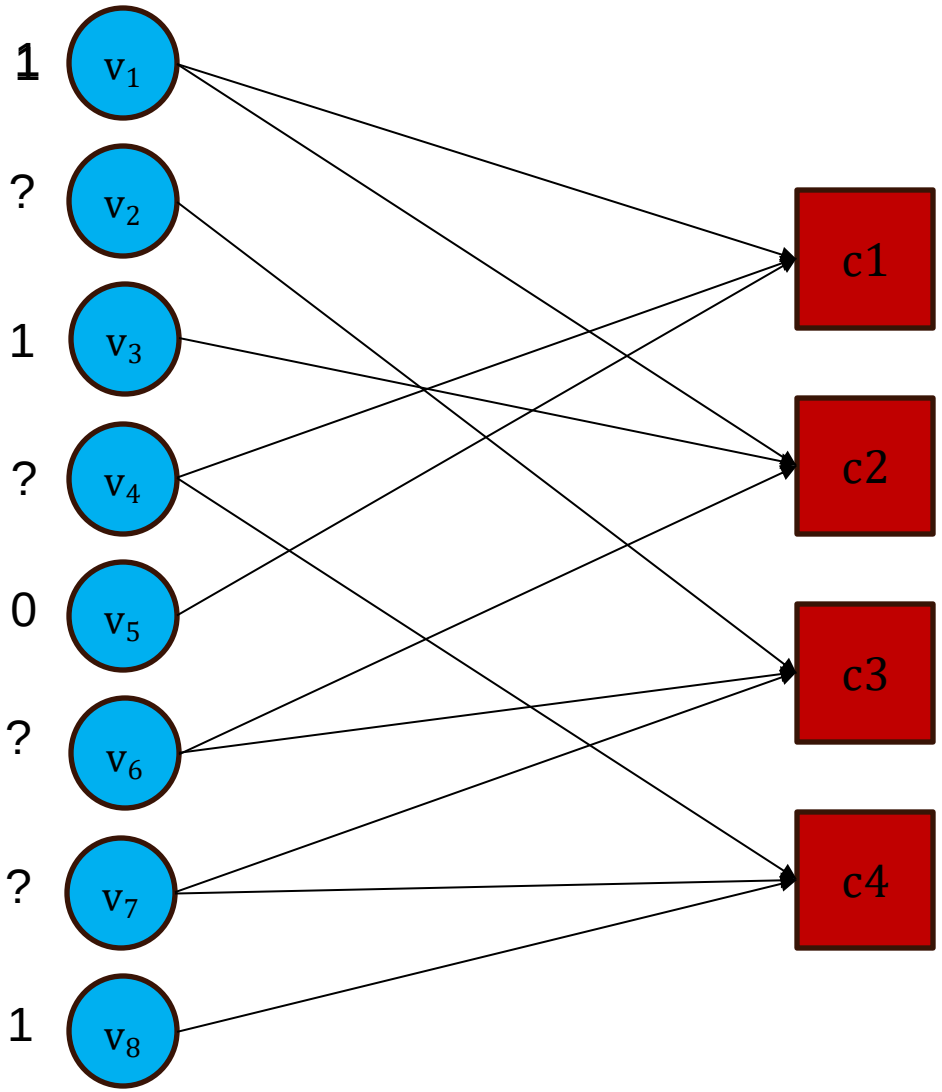


Suppose we transmit $\mathbf{x} = (1, 0, 1, 1, 0, 0, 0, 1)$ and receive $\mathbf{y} = (1, ?, 1, ?, 0, ?, ?, 1)$ with the parity check matrix from before

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Stanford | ENGINEERING
Electrical Engineering

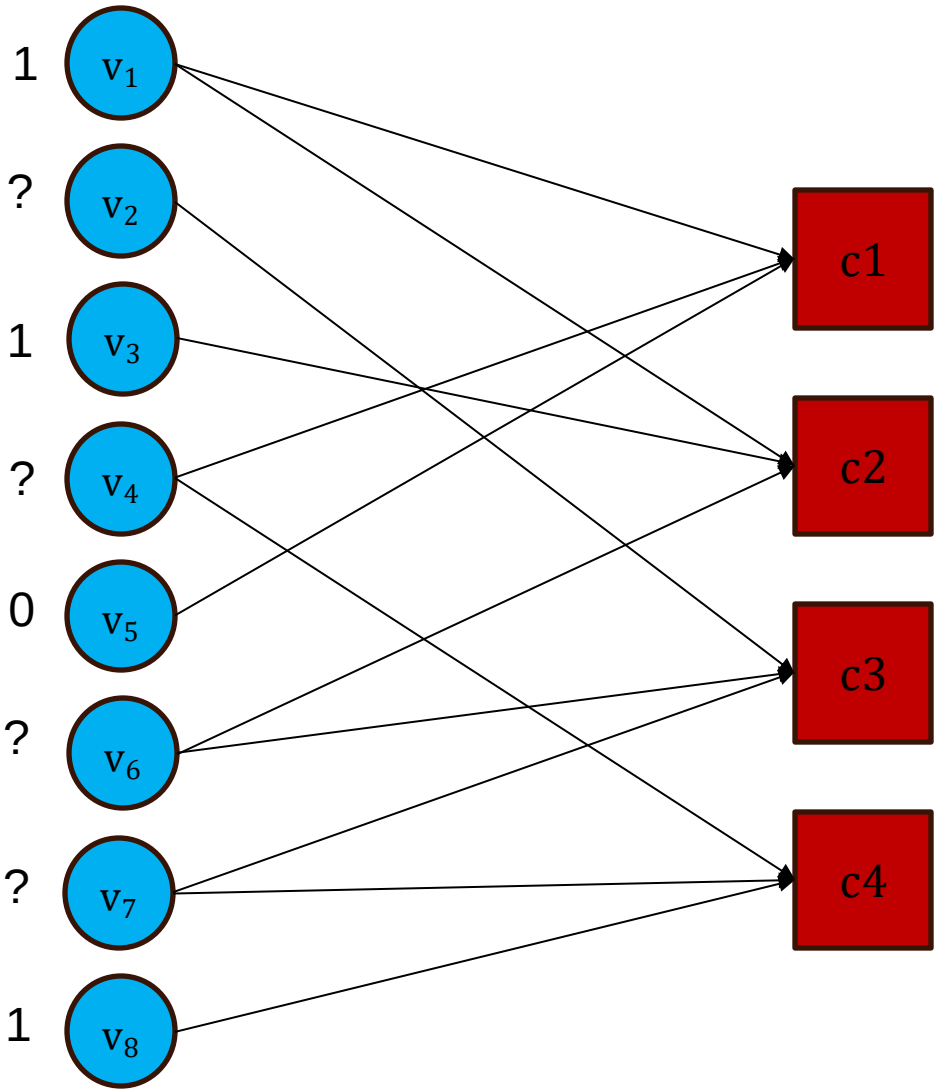
BEC Example



Transmitted $\mathbf{x} = (1, 0, 1, 1, 0, 0, 0, 1)$

Step 1: Variable nodes send their value to the check nodes

BEC Example



Transmitted $\mathbf{x} = (1, 0, 1, 1, 0, 0, 0, 1)$

Step 2: Check nodes compute their parity from received values excluding '?'

$P_1 = 1 + 0 = 1$ (connected to one '?' at v_4)

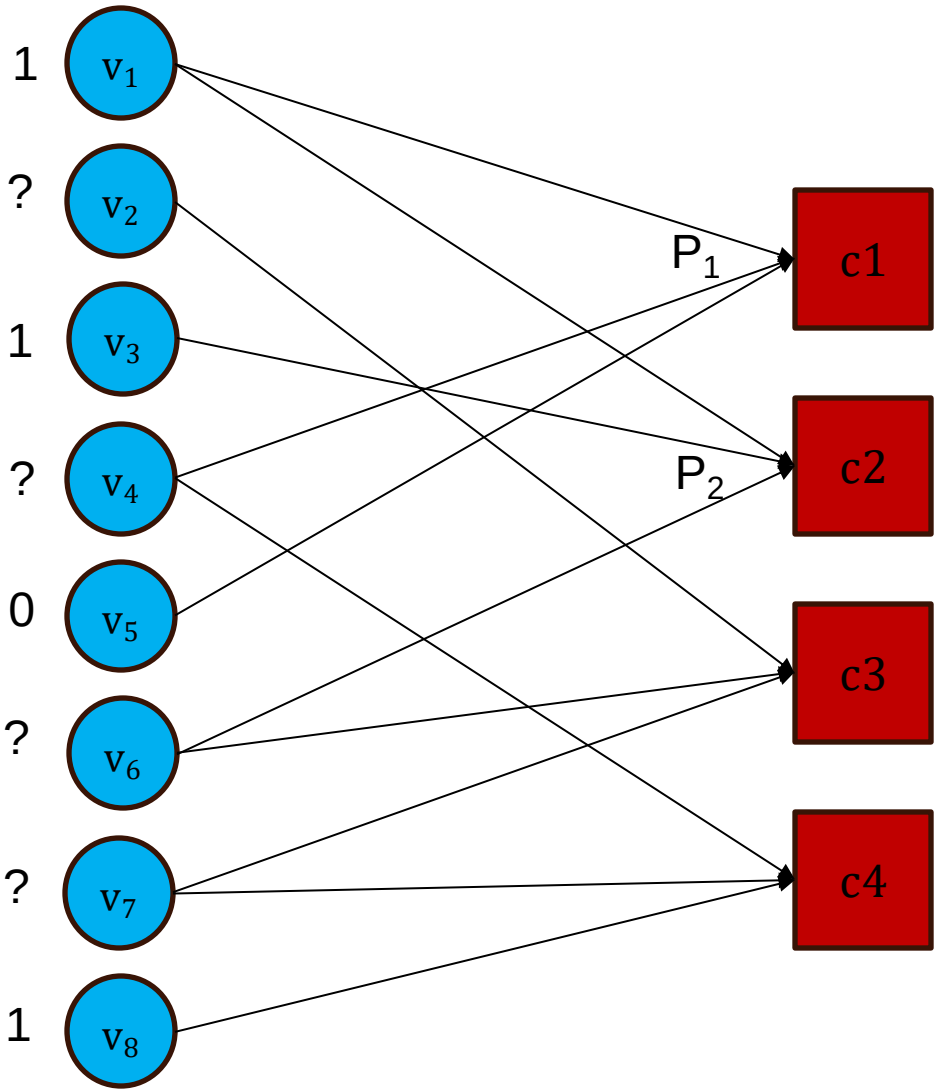
$P_2 = 1 + 1 = 0$ (connected to one '?' at v_6)

$P_3 =$? (connected to three '?'s)

$P_4 = 1$ (connected to two '?'s)

Stanford | ENGINEERING
Electrical Engineering

BEC Example



Transmitted $\mathbf{x} = (1, 0, 1, 1, 0, 0, 0, 1)$

Step 3: If a check node is connected to one variable node with a '?', send the parity back to the variable node

$P_1 = 1$ (connected to **one** '?' at v_4)

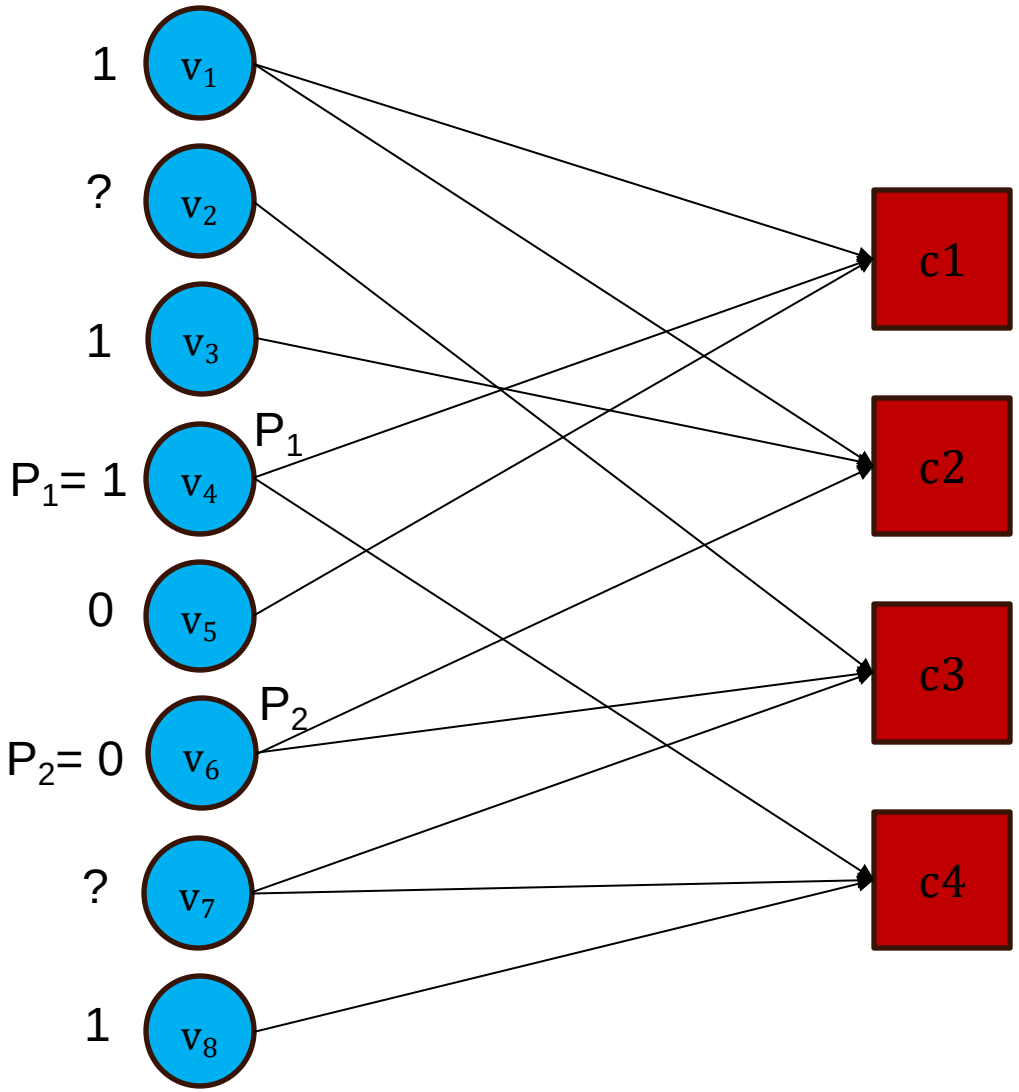
$P_2 = 0$ (connected to **one** '?' at v_6)

$P_3 = ?$ (connected to three '?'s)

$P_4 = 1$ (connected to two '?'s)

Stanford | ENGINEERING
Electrical Engineering

BEC Example



Transmitted $\mathbf{x} = (1, 0, 1, 1, 0, 0, 0, 1)$

Step 4: Variable nodes set to '?', assign themselves to the message sent by the check node

Now our message $y = (1, ?, 1, 1, 0, ?, ?, 1)$

If y has no more erased bits, return y. If no variables nodes changed their values, return failure. Otherwise return to step 1. So, we must run through more iterations. Eventually, one can show you reach the transmitted x.

Important observation:

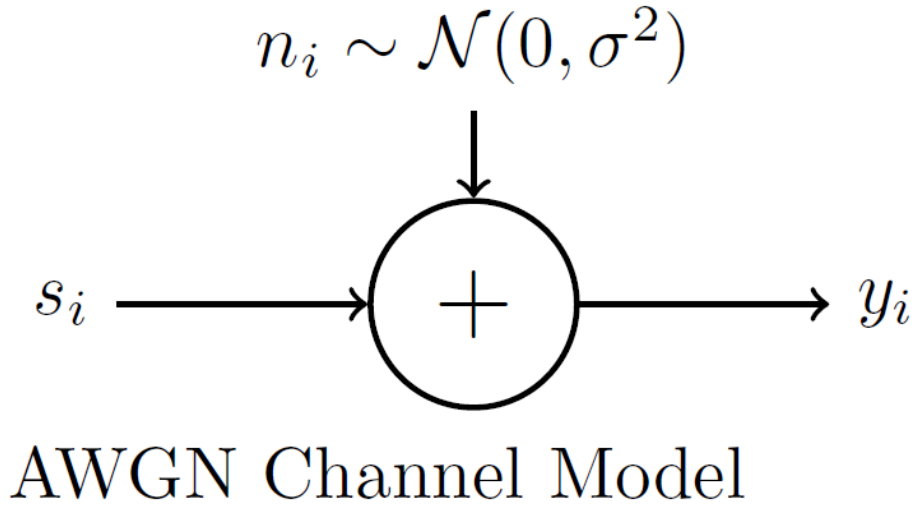
Passing messages allows us to gain information about the original message.

AWGN channel model and LLRs

Consider transmitting an n-bit LDPC codeword $x = (x_1, x_2, \ldots\ldots, x_n)$ over an AWGN channel using BPSK modulation

To facilitate soft (probabilistic) decoding, we compute the log-likelihood ratio (LLR) for each: $\mathrm{L_i} = \ln \frac{P(x_i=0|y_i)}{P(x_i=1|y_i)}$

For an AWGN channel with BPSK, it is common to approximate $L_i \approx \frac{2}{\sigma^2} y_i$

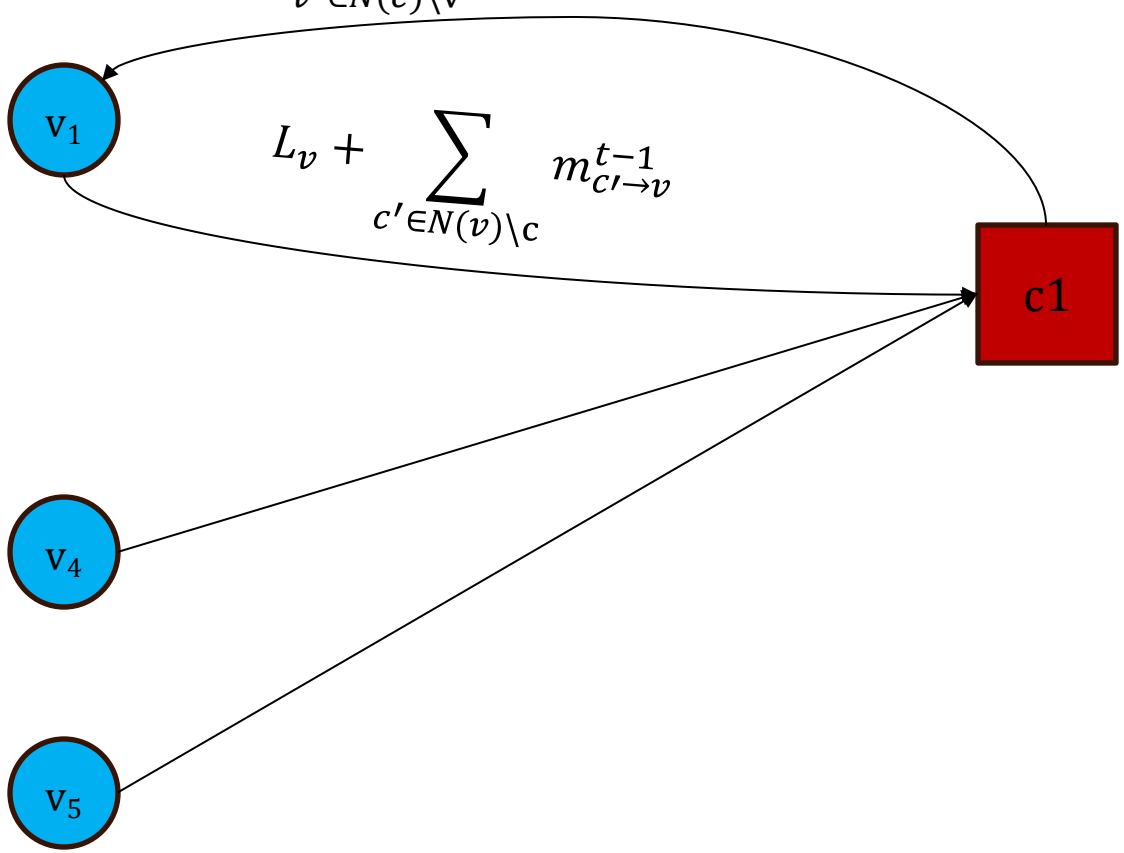


AWGN Channel Model

Sum-Product Algorithm [5]

Can be approximated as:

$$2tanh^{-1}\Big(\prod_{v'\in N(c)\setminus \mathrm{v}} \tanh\Big(\frac{m^{t}_{v'\to c}}{2}\Big)\Big) \approx \Big(\prod_{v'\in N(c)\setminus \mathrm{v}} sign\big(\mathrm{m}^{\mathrm{t}}_{v'\to c}\big)\Big) \cdot \min_{v'\in N(c)\setminus \mathrm{v}} \big|m^{t}_{v'\to c}\big|$$ (min-sum algo) [6]



- BEC Example but now, the messages utilize the LLR
- We sum and then we take a product, hence Sum-Product

LDPC Codes in 5G

What's Desired in Practice

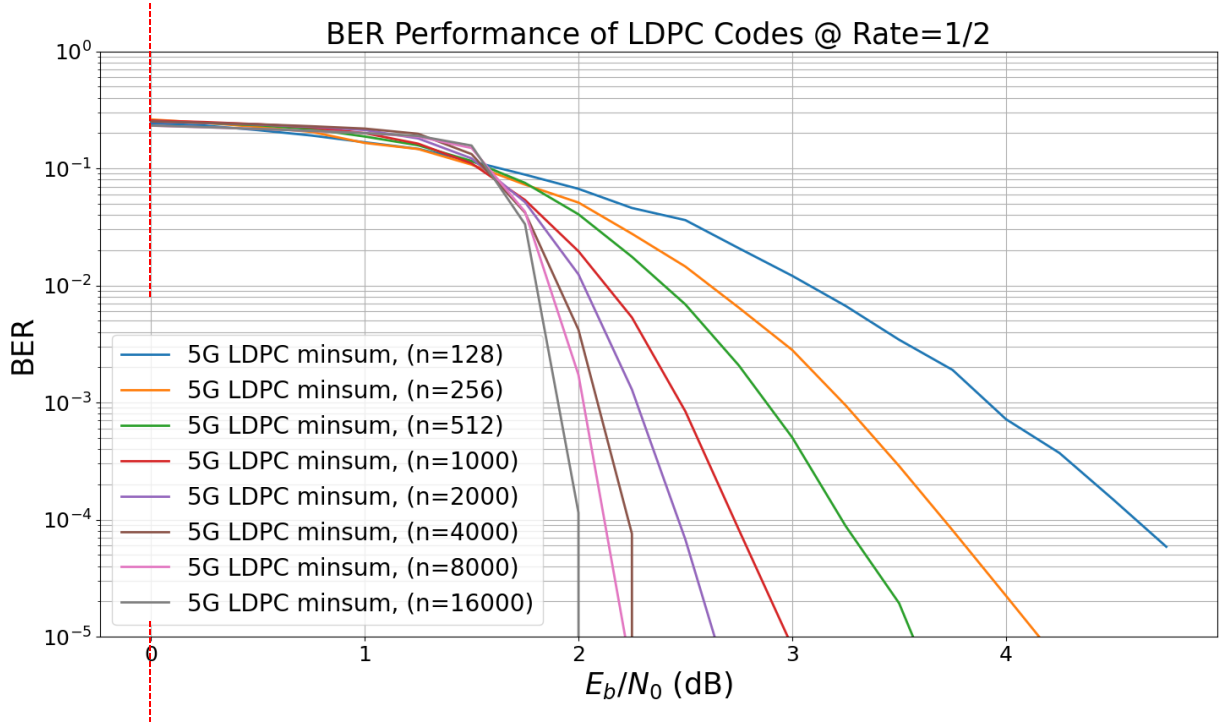
- In wireless systems, codes are subject to power/performance/area (PPA) constraints
- Want codes that are:
  - Reliable
  - Low complexity
  - Low latency/delay (parallelism)
  - Flexibility with rate and block length
    - Wireless channel is time-varying – to make the best use of capacity, code rate should adapt to channel conditions
    - Want to adapt code rate to either maximize throughput or operate most reliably
  - LDPC has many properties that allow for good performance along all of these axis which is why they are used in 5G (data channels) [7]

Increasing Code Lengths

- As N increases,
  - LDPC codes approach Shannon limit
  - Can get within 0.0045 dB [10]
- 5G uses LDPC for large block lengths (data channel)
  - Polar codes are also used, but only in the control channel where K < 1024 bits

Shannon Limit, $R = \frac{1}{2}, m = 2$ *(QPSK)*

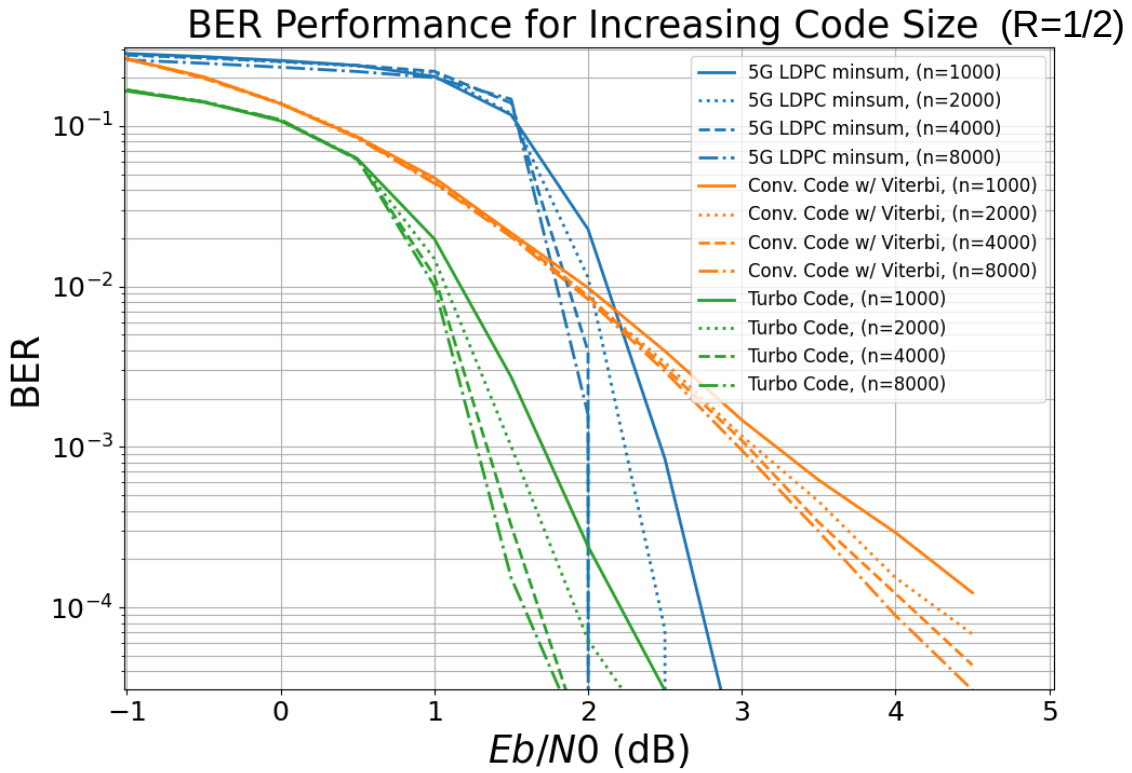
$$\frac{E_b}{N_{0_{min}}} = 10\, log_{10} \frac{2^{Rm} - 1}{Rm} = 0\ dB$$



Comparison to Legacy Schemes: BER

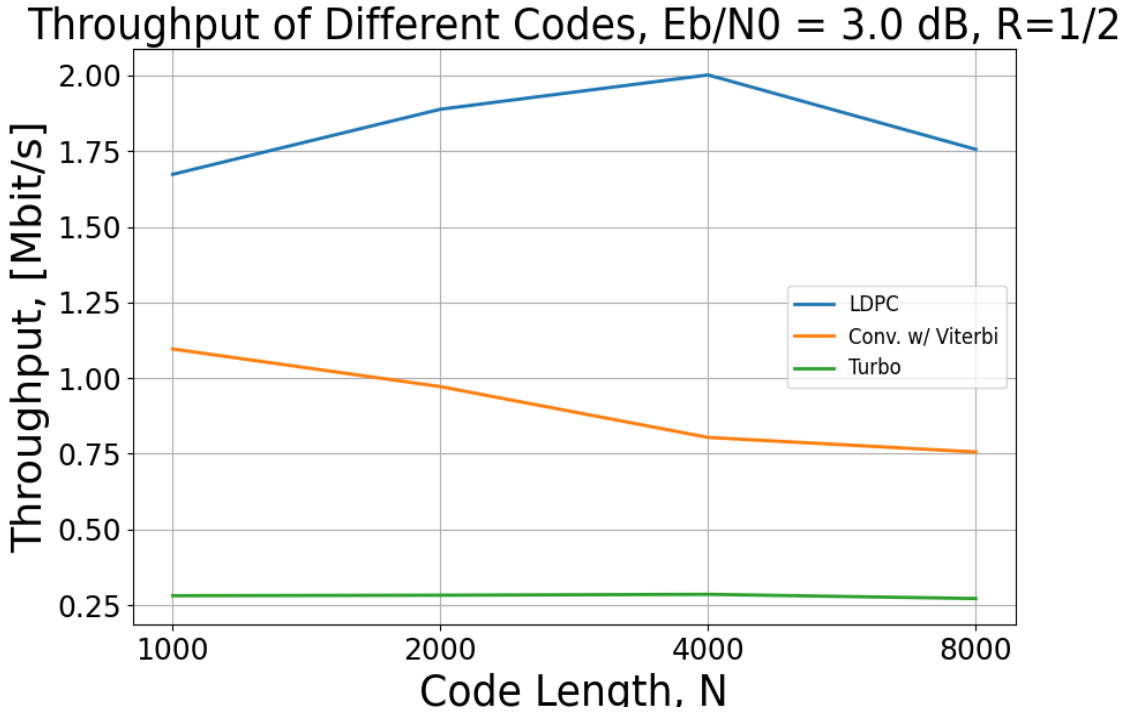
- 5G: LDPC minsum
- 4G: Turbo/Convolutional
- 2G/3G: Convolutional

- LDPC and Turbo codes have similar BER performance, so why move to LDPC codes for 5G?



Comparison to Legacy Schemes: Complexity

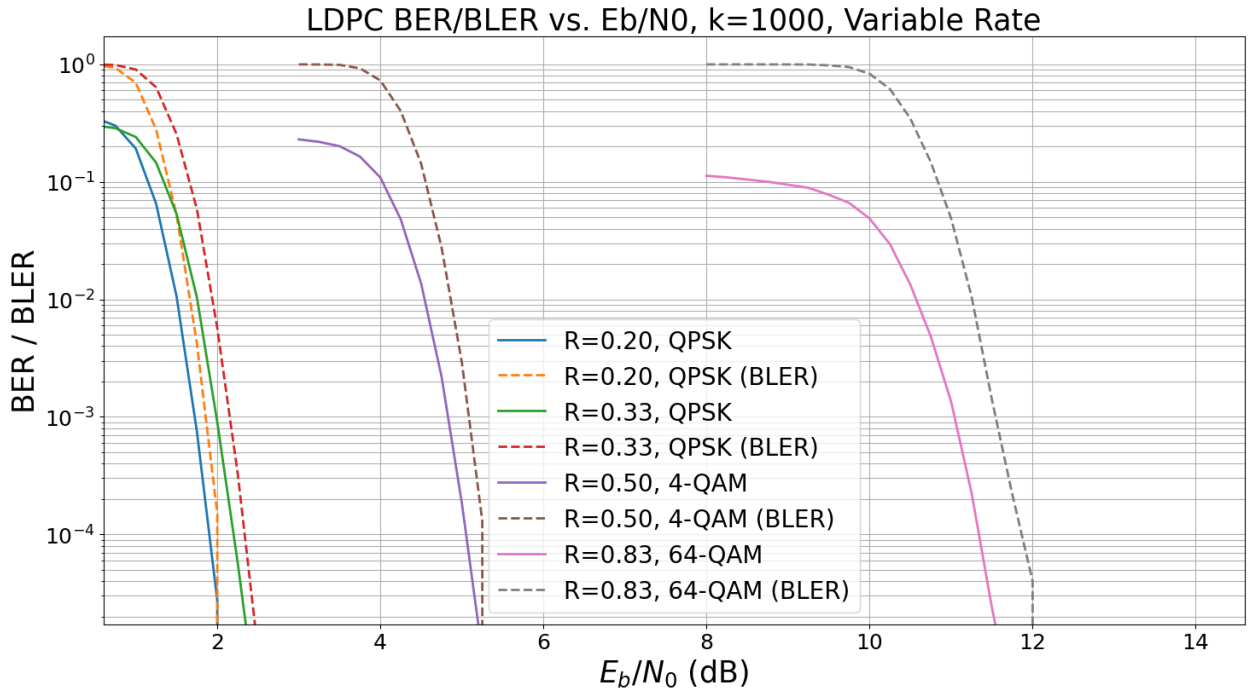
Code Type	Decoding Algorithm	Complexity	
LDPC (5G+)	Min-sum	$O(N * d_v)$ [7]	d_v = check node degree
Turbo (3G/4G)	BCJR	$O(L * N * 2^M)$ [8]	L = # of iterations M = # constraint
Convolutional (2G/3G)	Viterbi (ML)	$O(N * 2^M)$ [9]	M = # constraint



LDPC codes enable highest throughput due to lowest complexity and hardware parallelism

LDPC Rate Flexibility

- Channel conditions (E_B/N_0) dyanmic
  - High SNR, high rate
  - Low SNR, low rate
- LDPC codes have high-rate flexibility
  - $\boldsymbol{H}$ can be adapted to different rates by puncturing (removing check nodes) or extending (adding check nodes)



Stanford | ENGINEERING
Electrical Engineering

References

[1] R. Gallager, “Low-density parity-check codes,” IRE Transactions on Information Theory, vol. 8,no. 1, pp. 21–28, 1962.

[2] Richardson and R. Urbanke, “The renaissance of gallager’s low-density parity-check codes,” IEEE Communications Magazine, vol. 41, no. 8, pp. 126–131, 2003. URL: https://ieeexplore.ieee.org/document/1222728

[3] B. Kurkoski, “Low-density parity-check (ldpc) codes lecture notes.”

[4] R. Tanner, "A recursive approach to low complexity codes," in IEEE Transactions on Information Theory, vol. 27, no. 5, pp. 533-547, September 1981, doi: 10.1109/TIT.1981.1056404. URL: https://ieeexplore.ieee.org/document/1056404

[5] D. MacKay, “Good error-correcting codes based on very sparse matrices,” IEEE Transactions on Information Theory, vol. 45, no. 2, pp. 399–431, 1999

[6] N. Wiberg, H.-A. Loeliger, and R. Kotter, “Codes and iterative decoding on general graphs,” in Proceedings of 1995 IEEE International Symposium on Information Theory, 1995, pp. 468–.

[7] T. Richardson and S. Kudekar, "Design of Low-Density Parity Check Codes for 5G New Radio," in IEEE Communications Magazine, vol. 56, no. 3, pp. 28-34, March 2018, doi: 10.1109/MCOM.2018.1700839. URL: https://ieeexplore.ieee.org/document/8316763

[8] D. Fertonani, A. Barbieri and G. Colavolpe, "Reduced-Complexity BCJR Algorithm for Turbo Equalization," in IEEE Transactions on Communications, vol. 55, no. 12, pp. 2279-2287, Dec. 2007, doi: 10.1109/TCOMM.2007.910638. URL: https://ieeexplore.ieee.org/abstract/document/4395266

[9] J. Feldman, I. Abou-Faycal and M. Frigo, "A fast maximum-likelihood decoder for convolutional codes," Proceedings IEEE 56th Vehicular Technology Conference, Vancouver, BC, Canada, 2002, pp. 371-375 vol.1, doi: 10.1109/VETECF.2002.1040367. URL: https://ieeexplore.ieee.org/document/1040367

[10] Sae-Young Chung, G. D. Forney, T. J. Richardson and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit," in IEEE Communications Letters, vol. 5, no. 2, pp. 58-60, Feb 2001, doi: 10.1109/4234.905935. URL: https://ieeexplore.ieee.org/document/905935

Stanford | ENGINEERING
Electrical Engineering