Received 8 February 2024; revised 30 March 2024; accepted 7 April 2024. Date of publication 16 April 2024; date of current version 7 May 2024.

Digital Object Identifier 10.1109/OJCOMS.2024.3390000

Channel Coding Toward 6G: Technical Overview and Outlook

MOHAMMAD ROWSHAN (Member, IEEE), MIN QIU (Member, IEEE), YIXUAN XIE (Member, IEEE), XINYI GU (Graduate Student Member, IEEE), AND JINHONG YUAN (Fellow, IEEE)

(Invited Paper)

School of Electrical Engineering and Telecommunications, University of New South Wales, Sydney, NSW 2052, Australia

CORRESPONDING AUTHOR: M. ROWSHAN (e-mail: m.rowshan@unsw.edu.au)

This work was supported in part by the Australian Research Council (ARC) Discovery Project under Grant DP220103596, and in part by the ARC Linkage Project under Grant LP200301482.

ABSTRACT Channel coding plays a pivotal role in ensuring reliable communication over wireless channels. With the growing need for ultra-reliable communication in emerging wireless use cases, the significance of channel coding has amplified. Furthermore, minimizing decoding latency is crucial for critical-mission applications, while optimizing energy efficiency is paramount for mobile and the Internet of Things (IoT) communications. As the fifth generation (5G) of mobile communications is currently in operation and 5G-advanced is on the horizon, the objective of this paper is to assess prominent channel coding schemes in the context of recent advancements and the anticipated requirements for the sixth generation (6G). In this paper, after considering the potential impact of channel coding on key performance indicators (KPIs) of wireless networks, we review the evolution of mobile communication standards and the organizations involved in the standardization, from the first generation (1G) to the current 5G, highlighting the technologies integral to achieving targeted KPIs such as reliability, data rate, latency, energy efficiency, spectral efficiency, connection density, and traffic capacity. Following this, we delve into the anticipated requirements for potential use cases in 6G. The subsequent sections of the paper focus on a comprehensive review of three primary coding schemes utilized in past generations and their recent advancements: low-density parity-check (LDPC) codes, turbo codes (including convolutional codes), and polar codes (alongside Reed-Muller codes). Additionally, we examine alternative coding schemes like Fountain codes (also known as rate-less codes), sparse regression codes, among others. Our evaluation includes a comparative analysis of error correction performance and the performance of hardware implementation for these coding schemes, providing insights into their potential and suitability for the upcoming 6G era. Lastly, we will briefly explore considerations such as higher-order modulations and waveform design, examining their contributions to enhancing key performance indicators in conjunction with channel coding schemes.

INDEX TERMS Channel coding, error control coding, error correction codes, wireless, mobile communications, 5th generation, 5G, 6th generation, 6G, encoding, decoding, channel polarization, polar codes, PAD codes, monomial codes, CRC, low-density parity-check codes, LDPC codes, convolutional codes, turbo codes, spatially coupled codes, fountain codes, spinal codes, raptor codes, Luby transform codes, LT codes, lattice codes, non-binary codes, sparse regression codes, SPARC, machine learning, neural codes, neural decoding, successive cancellation decoding, beleif propagation decoing, Message Passing Decoding, BCJR decoding, iterative decoding, Viterbi decoding, Fano decoding, automorphism ensemble decoding, min-sum algorithm, bit-flipping, PPV bound, normal approximation, coded modulation, bit-interleaving, puncturing, shortening, repetition, rate-compatible codes, application layer channel coding, physical layer, waveform, non-terrestrial networks, free space optical links, modulation, block error rate, BLER, frame error rate, FER, reliability, latency, complexity, enhanced mobile broadband, eMBB, machine-type communications, MTC, ultra-reliable low-latency, URLLC, key performance indicator, KPI, hardware architecture, waveform design.

I. INTRODUCTION

Since the early 1980s, mobile communications have undergone a generational change almost every decade. With the emergence of new applications as well as the rapid technology advancements in hardware and computing power, the time difference between mobile communication generations is decreasing. Although 5G of mobile communications systems is already a commercial reality, there has been ongoing research in designing beyond 5G (B5G), and the sixth generation (6G) communication systems.

The emergence of new use cases drives the increasing demands on high data rates, high reliability, and low latency. To fulfill these visionary requirements, the next generation of wireless systems would require the allocation of new frequency bands and the development of new communication architectures. The physical layer techniques will play an important role in the realization of the vision for 6G. Among them, channel coding, modulation, and signal waveforms are essential for the next generation of air interface design. In this paper, our focus is on the channel coding aspect of 6G networks.

Channel coding is essential in all communication systems to ensure reliable and efficient communications. It involves adding redundancy to the transmitted data in a controlled manner, allowing the receiver to detect and correct errors that occur during transmission and reception. Wireless channels are noisy and unreliable. Therefore, data can be corrupted during transmission due to factors such as noise, interference, and channel fading. When errors in transmitted data cannot be corrected, retransmissions are required to obtain the correct data. Consequently, retransmissions can introduce a long delay, leading to poor end-user experience. Essentially, the role of channel coding is to ensure that the data received are the same as the data sent. Channel coding plays a critical role in improving the key performance indicators (KPIs) of a cellular network, including:

- Reliability: Channel coding is crucial to improve communication reliability by providing error detection and correction capabilities. It ensures that the data are reliably received in the presence of noise, interference, or fading conditions.
- Throughput: Channel coding affects throughput by influencing the error rate and reliability of data transmission. Efficient channel coding helps mitigate errors, reduce retransmissions, and improve overall throughput.
- Latency: Although channel coding adds processing delay due to encoding and decoding operations, efficient coding schemes contribute to minimizing the number of retransmissions. This reduction in retransmissions can help mitigate the overall latency.
- Coverage: Channel coding plays a critical role in extending coverage by enhancing the ability of the network to transmit data reliably over longer distances and in challenging radio environments.
- Spectral Efficiency: Efficient channel coding improves spectral efficiency by minimizing the impact of errors on the effective use of available frequency spectrum.
- Energy Efficiency: Channel coding can contribute to the reduction in power requirements at both the base station and user devices, as well as retransmissions, consequently, resulting in an extension of battery life.

TABLE 1. Coding schemes from 1G to 5G.

Gen.	Channel Coding
2G	Cyclic Codes (FIRE/CRC), (Punctured) Convolutional Codes
3G	Convolutional Codes, Turbo Codes
4G	Tail Biting Convolutional Codes, Turbo Codes
5G	Polar Codes, LDPC Codes

As cellular networks have evolved from the second generation (2G) to the current 5G era, the path of channel coding has undergone significant advancements to address the challenges posed by varying channel conditions, increasing data rates, and the quest for ultra-reliable and low-latency communications. The coding schemes used in the 2G to 5G wireless communication standards are listed in TABLE 1.

In this paper, we review the technologies involved in various generations of cellular networks to improve KIPS, with a particular focus on the channel coding schemes employed in each generation. We also review the envisioned requirements and the KPIs for 6G. Then, we turn our focus to channel coding schemes used in previous generations and cover new advances. We also consider other coding schemes developed recently. Then, we draw a conclusion. The list of contents is as follows.

II. THE 3RD GENERATION PARTNERSHIP PROJECT (3GPP)

The 3rd generation partnership project (3GPP) is an umbrella term for a consortium of national (from Japan, USA, China, India and South Korea) or regional (from Europe) telecommunication standards organizations and other organizations to develop protocols for mobile telecommunications. As the name implies, this project was initially established to develop specifications for the 3rd generation (3G) of the cellular network based on 2G in December 1998 [1]. As the 3GPP standardization work is contribution-driven, companies can participate through their membership in organizational partners. The 3GPP work is divided into three streams performed by three technical specification groups (TSGs), each of which consists of multiple working groups (WGs) [1]: Radio Access Networks (RAN), Services and Systems Aspects (SA), and Core Network and Terminals (CT). Among them, TSG RAN is responsible for the technical specifications of radio layer 1 (that is, physical layer, including multiplexing, channel coding, and error detection, the subject of this paper), layer 2, layer 3, etc.

3GPP standards are organized as *Releases*. Each release consists of many individual technical specification and

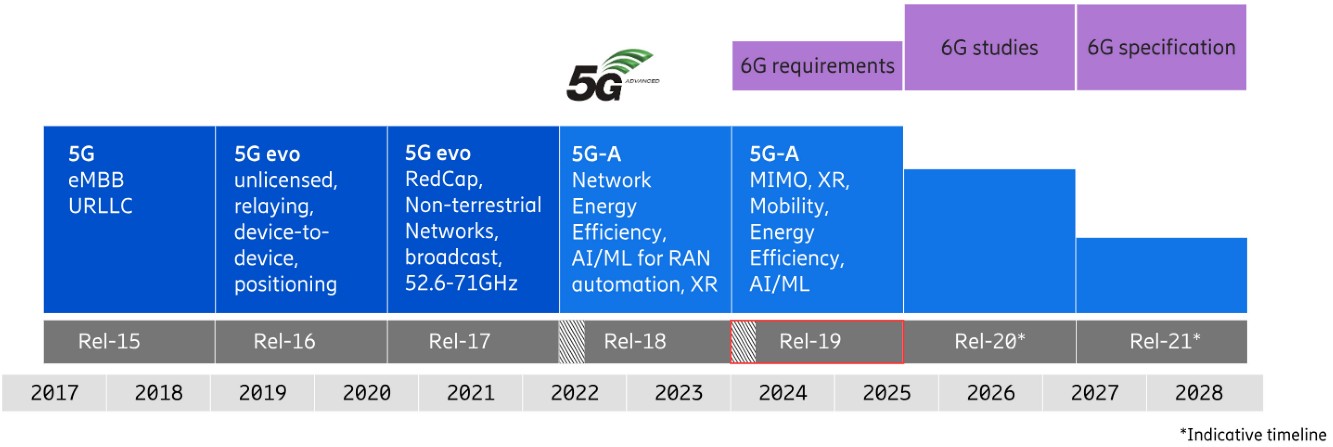


FIGURE 1. Roadmap for 5G standard releases towards 6G [6].

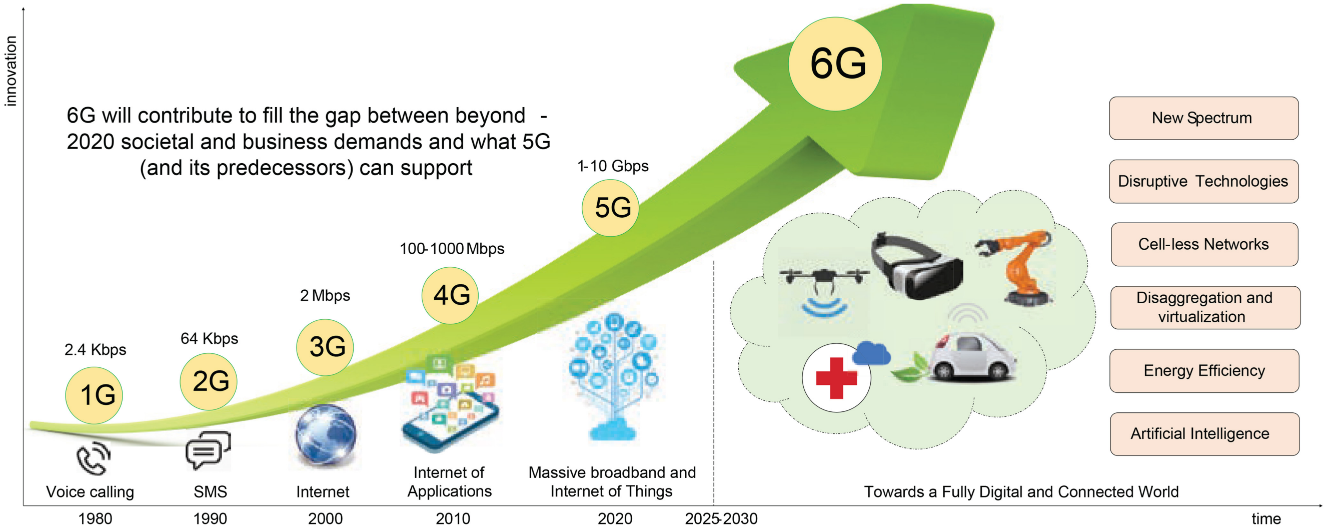

innovation
6G will contribute to fill the gap between beyond - 2020 societal and business demands and what 5G (and its predecessors) can support
6G
1-10 Gbps
5G
100-1000 Mbps
4G
2 Mbps
3G
64 Kbps
2G
2.4 Kbps
1G
Voice calling
SMS
Internet
Internet of Applications
Massive broadband and Internet of Things
Towards a Fully Digital and Connected World
New Spectrum
Disruptive Technologies
Cell-less Networks
Disaggregation and virtualization
Energy Efficiency
Artificial Intelligence
1980
1990
2000
2010
2020
2025-2030
time

FIGURE 2. From 1G to 5G and beyond [7].

technical report documents, not only about new generations, but also including the revised documents of the older generations. For instance, Release 18 was concluded in late 2023 on 5G Advanced (5G-A). Fig. 1 illustrates the roadmap of 5G towards 6G based on 3GPP releases. These documents may go through many revisions over the years. The documents are available free of charge on 3GPP's portal [2]. Note that another collaboration between telecommunications associations called 3GPP2 existed until 2013. 3GPP2 was behind CDMA2000, the 3G upgrade to 2G cdmaOne networks used mainly in the United States. In this paper, our focus is only on the standards developed by 3GPP.

Furthermore, the requirements for future generations of international mobile telecommunications (IMT) are defined by the United Nations' International Telecommunication Union (ITU) radiocommunication sector (ITU-R). For example, the requirements for 3G (called IMT-2000 [3]), 4G (called IMT-Advanced [4]), and 5G (called IMT-2020 [5]) were issued in 1999, 2008, and 2015, respectively. The technical specifications are then delegated to the 3GPP. The 3GPP prioritizes and divides specifications into releases according to the order in which new functionalities will be implemented and deployed in cellular networks. The specifications in the 3GPP releases give operators and manufacturers confidence in their designs and investments.

III. A JOURNEY FROM 1G TO 5G

Since the first generation of mobile communication systems in the 1980s, different channel coding schemes and various technologies have been employed depending on the requirements and availability of the scheme(s) that meet the specifications. Table 2 summarizes the specifications of different generations of mobile networks given in the 3GPP releases. As can be seen, every generation has a specific name in the column "system" where the 5G is called "New Radio" or NR. Fig. 2 illustrates the improvement in data rate, the use cases added over generations, and the innovations expected to achieve the 6G goals.

TABLE 2. Comparison of network specifications from 1G to 5G. In this table, double comma, indicates repetition of the above.

Gen.	3GPP Release	Initial Rollout	System	Access Methodology	Peak Data Rate [bps]	Latency [ms]	Modulation	Freq. Band [MHz]	Carrier Spacing [Hz]
1G			AMPS*	FDMA			FM (voice) FSK (control)	DL:824–849, UL:869–894	30k
2G	96	1991	GSM	F/TDMA	14.4k		GMSK	DL:890–915, UL:935–960	200k
2.5G	97	2000	GPRS	,,	171k				,,
2.75G	98	2003	EDGE	,,	236k	< 150	GMSK, 8-PSK		,,
3G	99,4	2003	UMTS	DS-CDMA	2M	150	DL:QPSK, UL:BPSK	890, 1900, 2100	5M
3.5G	5,6	2005	HSPA	,,	DL:14.4M, UL:5.76M	100	QPSK, 16QAM	,,	,,
3.75G	7	2008	HSPA+	,,	DL:42M, UL:28M	< 50	QPSK, 16QAM, 64QAM	,,	,,
3.9G	8,9	2009	LTE	DL:OFDMA UL:SC-FDMA	DL:300M, UL:75M	10	,,	800, 1900, 2100	15k
4G	10	2014	LTE-A	DL:OFDMA UL:SC-FDMA	DL:3G, UL:1.5G	< 5	QPSK, 16QAM, 64QAM	800, 1900, 2100, 2500, 2600	15k
4.5G	13	2016	LTE-AP	,,	DL:3G, UL:1.5G	< 5	,,	800, 1900, 2100, 2500, 2600, 3500	15k
5G	15,16,17	2018	NR	DL:CP-OFDMA, UL:DFT-S-OFDM	DL:20G, UL:10G	< 1	QPSK, 16QAM, 64QAM, 256QAM, UL: $\pi/2$ BPSK	FR1:410-7125, FR2:24250-52600	FR1:15,30,60k, FR2:60,120,240k

$*$ There were various specifications implemented by different operators in different countries. See Section A.

We will discuss the details and technologies that enable every generation to achieve these specifications as follows.

A. FIRST GENERATION (1G)

Although it was not called 1G at the time, the first generation of mobile cellular networks emerged in Japan in 1979 by Nippon Telegraph and Telephone (NTT) Corporation known as "Car Telephone Service" and later was rolled out by Ameritech in Chicago in 1983, known as the Advanced Mobile Phone System (AMPS) [8]. Soon after, other companies in different regions and countries launched their own networks, each following different specifications. AMPS was based on narrow-band analog frequency modulation (FM) with a usable audio frequency band of 300 Hz - 3 kHz with frequency division multiple access (FDMA) technology, where transmissions are separated in the frequency domain. Subscribers are assigned a pair of voice channels (forward and reverse) for the duration of their call. Analog cellular channels carried both voices using FM, operating between 824–849 MHz (from mobile stations to base stations) and 869-894 MHz (from base stations to the mobile stations). However, digital signaling based on binary frequency-shift keying (FSK) was used to connect mobile customers to the base station as a control channel to place voice calls; these data were transmitted at 10 kbit/s. The 800 MHz band was split into a number of channels (395 voice, 21 control) with FDMA, where each channel was 30 kHz wide. The cells were able to cover a very large area (often a radius of >40 km depending on the terrain (land)). Although the large coverage had low infrastructure costs, it required the base station and the mobile terminal to transmit at high power to bridge large distances. Since voice signals were analog, they could not be efficiently compressed or protected by channel coding techniques like those used for digital data. This inherent constraint often resulted in compromised voice quality and security.

B. SECOND GENERATION (2G)

The second generation of cellular networks, dubbed the global system for mobile communications (GSM), was developed by the European Telecommunications Standards Institute (ETSI) and commercially launched in Finland in 1991 [8]. 2G was an upgrade from 1G's analog radio signals to digital radio signals over a circuit-switched network optimized for full duplex voice telephony to provide a secure and reliable communication link. Furthermore, this generation adopted digital encrypted conversations using 64-bit A5/1 stream cipher and introduced data services such as short message services (SMS) and multimedia messaging services (MMS), Internet access, and also subscriber identity module (SIM) card to securely store the international mobile subscriber identity (IMSI) number and its related key, as well as to allow users to switch networks and handsets at will. The commonly used radio frequency band by the GSM is the 900 MHz and 1800 MHz bands. GSM used a combination of time-division multiple access (TDMA) and frequency-division multiple access (FDMA). The bandwidth of 25 MHz (maximum) was divided into 124 carrier frequencies spaced 200 kHz apart using the FDMA scheme. However, carriers were divided into time slots, using the TDMA scheme, to be allocated to different users. Each time slot and the transmission made within it, called a GSM burst, lasted 0.577 ms. Every eight bursts were grouped and called a TDMA frame lasted approximately 4.615 ms, which formed a basic unit for the definition of logical channels, whereas one physical channel was one burst period in each TDAM frame. For synchronization purposes, the frames were organized into multiframes and the so-called superframes.

The modulation scheme in GSM is Gaussian minimum shift keying (GMSK). The technologies involved in GSM expanded over time to include data communications, first by circuit-switched transport (as in the public switched telephone network, PSTN), then by packet-switched transport via the general packet radio service (GPRS), also known as 2.5G, which was commercially launched in 2000 and could be used for Internet connection. In circuit switching, the bit delay is constant during a connection (as opposed to packet switching, where packet queues may cause varying and potentially indefinitely long packet transfer delays). The second extension of GSM called enhanced data rates for GSM Evolution (EDGE), also known as 2.75G, became operational in 2003 [8]. In theory, the speed limit of GPRS is 115 kbps, but in most networks it is around 35 kbps.

EDGE as a single-carrier standard based on GSM can have a data rate up to 236 kbit/s (with an end-to-end latency of less than 150 ms) for 4 timeslots (for 8 timeslots, the theoretical maximum is 473.6 kbit/s) in packet-switching mode. The theoretical maximum speed is 473 kbps for 8 timeslots, but it is typically limited to 135 kbps to conserve spectrum resources. This is four times as much traffic as GPRS. Therefore, EDGE could meet the 3G requirements defined by the ITU-R known as IMT-2000 [3].

Note that unlike 1G, in 2G the analogue voice signal was first compressed into a digital signal by source coding. As a result, the digitized signal can be protected by error correction codes. The channel codes used for GSM were block codes and convolutional codes with relatively simple structures and moderate performance.

EDGE, like GPRS, uses a rate adaptation algorithm that adapts the modulation and coding scheme (MCS) according to the quality of the radio channel, and thus adjusts the bit rate and robustness of data transmission. It also introduces *incremental redundancy* in which, instead of retransmitting disturbed packets, it sends more redundancy bits to be combined at the receiver. This increases the probability of correct decoding. EDGE uses 8 phase-shift keying (8PSK) in addition to GMSK, for the upper five of its nine MCSs. Since every symbol carries 3 bits in 8PSK, EDGE can effectively triple the gross data rate offered by GSM.

TABLE 3. Convolutional coding parameters in GSM (2G).

Scheme	Code rate	Generator Polynomial	BCS	Tail
CS-1	1/2	$1 + D^2 + D^3$	40	4
CS-2	∼2/3	$1 + D^2 + D^3 + D^4 + D^5$	16	4
CS-3	∼3/4	$1 + D^2 + D^3 + D^4 + D^5 + D^6$	16	4
CS-4	1		16	-

TABLE 4. Convolutional coding parameters in EDGE (2.75G).

Scheme	Code rate	BCS	Tail Bits	Modulation	Family
MCS-1	53/100	12	6	GMSK	C
MCS-2	33/50	12	6		B
MCS-3	17/20	12	6		A
MCS-4	1	12	6		C
MCS-5	37/100	12	6	8-PSK	B
MCS-6	49/100	12	6		A
MCS-7	19/25	2×12	2×6		B
MCS-8	23/25	2×12	2×6		A
MCS-9	1	2×12	2×6		A

According to the ETSI's technical specification TS 45.003 [9], channel coding in 2G is performed in two steps: In the first step, a block check sequence (BCS) is added for error detection. The BCS is either a 40-bit FIRE code or a 16-bit CRC with generator polynomial $g_{\text{CRC16}}(D) = D^{16} + D^{12} + D^5 + 1$. In the second step, four tail bits (TBs) are added and half-rate convolutional coding is performed for error correction. The convolutional codeword may be punctured to obtain the desired code rate. CS-1 to CS-4 are used to specify the length of the BCS and the puncturing rate (in CS-2 and CS-3) of the convolutional code, as shown in Table 3. Note that in CS-4, where no convolutional coding is applied, we have the fastest but least reliable transmission, which is used for communications near a base transceiver station (BTS). Whereas the most robust coding scheme (CS-1) was used when the user equipment (UE) is further away from a BTS. Also, for most of the control channels, CS-1 is used. The bit rate per time slot increases from 8 kbps in CS-1 to 12, 14.4, and 20 kbps in CS-2, CS-3, and CS-4, respectively.

As GSM evolved and 8-PSK modulation was added to enhanced GPRS (EGPRS) in EDGE, the link quality control (LQC) was adapted to adjust the modulation and coding scheme (MCS) to the most suitable one as per the channel condition. There are nine MCSs (MCS-1 to MCS-9) for link adaptation, as shown in Table 4. The MCSs are divided into three families A, B, and C with the basic unit of payload 37 (and 34), 28, and 22 octets, respectively. Different code rates within a family are achieved by transmitting a different number of payload units within one Radio Block.

For families A and B, 1, 2, or 4 payload units are transmitted, for family C, only 1 or 2 payload units are transmitted. Similar to GSM, a BCS is first added to every two payloads for error detection and then interleaved over two or four bursts, depending on the MCS. In the second step, six or twelve tail bits, depending on the MCS, are added, and then a rate-1/3 convolutional coding is performed for error correction, which is punctured to give the desired coding rate. The bit rate per time slot increases from 8.8 kbps in MCS-1 to 59.2 kbps in MCS-9.

C. THIRD GENERATION (3G)

The specifications of the third generation of the cellular network were defined in 3GPP Release 99 to meet the requirements of ITU-R IMT-2000. This generation was called the universal mobile telecommunications system (UMTS). Unlike GSM and evolved GSM, it uses wideband code-division multiple access (W-CDMA) radio access technology to offer greater spectral efficiency and bandwidth to mobile network operators, hence UMTS is comparable with the CDMA2000 standard based on cdmaOne technology widely used in the United States as 2G. The modulation technique employed in UMTS is quadrature phase shift keying (QPSK). 3G added the 2100 MHz frequency band with 5 MHz bandwidth to the 2G frequency bands.

High-speed packet access (HSPA), also known as 3.5G, introduced with 3GPP Release 5, is a combination of uplink (known as HSUPA) and downlink (known as HSDPA) protocols that allow UMTS-based networks to have higher data rates and capacity. The key features of HSDPA are shared channel and multi-code transmission, higher-order modulation, QPSK, 16-quadrature amplitude modulation (16QAM), short transmission time interval (TTI), fast link adaptation and scheduling, and fast hybrid automatic repeat request (HARQ) with incremental redundancy, making retransmissions more effective. Another new feature is the high-speed medium access protocol (MAC-hs) in the base station.

Evolved high-speed packet access (HSPA+), also known as 3.75G, was introduced in 3GPP Release 7, further increased data rates (up to 42 Mbps in the downlink and 28 Mbps in the uplink with a single 5 MHz carrier) by adding 64QAM modulation, 2 × 2 multiple input multiple output (MIMO) technology. With a dual carrier or dual cell (DC) where two base stations with different carrier frequencies are employed to communicate with user equipment, the data rate can be doubled. Note that higher-order modulation schemes such as 64QAM within a limited bandwidth (as bandwidth is a scarce and expensive resource) require higher receiver signal-to-noise or high signal-to-interference ratios, e.g., available in small cells or for user equipment (UE) close to a base station (BS, in the standard documents, it is called NodeB, eNB, or gNB, depending on the generation).

Originally, long term evolution (LTE), also known as 3.9G or pre-4G, was conceived as an IP-based wireless system used purely for carrying data traffic. Network carriers were supposed to provide voice communication through their concurrent 2G/3G networks or using VoIP. However, by popular request, Voice over LTE (VoLTE) was a standardized system for transferring voice traffic over LTE. Currently, the availability of Voice over LTE (VoLTE) depends on the carrier implementation. Theoretically, LTE networks should provide wireless data downlink speeds of up to 300 Mbps and uplink speeds of up to 75 Mbps [10]. Note that LTE was originally marketed as 4G, but it did not meet all 4G requirements, as defined by the ITU. Therefore, it was considered pre-4G, although it is still widely referred to as 4G.

The multiple access methodology used in LTE was orthogonal frequency-division multiple access (OFDMA) for the downlink and single-carrier frequency-division multiple access (SC-FDMA) for uplink. In SC-FDMA, each symbol is precoded by a discrete Fourier transform (DFT) before mapping to subcarriers; hence, unlike OFDMA, multiple subcarriers carry each data symbol. As a result, SC-FDMA offers (1) lower peak-to-average power ratio (PAPR) that benefits the mobile terminal in terms of transmit power efficiency and lower cost of the power amplifier, and (2) spreading gain or frequency diversity gain in a frequency selective channel, hence called DFT-spread OFDM.

The capacity improvement of 3G over 2G is significant, which largely benefit from soft frequency reuse, fast power control, and turbo codes [11]. Turbo codes were proposed in 1993 [12]. They are the first coding schemes that have been practically demonstrated to approach the Shannon limit of a point-to-point channel with moderate decoding complexity. Because of this, turbo codes soon became the standard channel coding schemes for 3G. The invention of turbo codes was considered a major breakthrough in coding and communication theory.

According to the 3GPP's technical specification TS 25.222 [13], to be able to detect any errors that cannot be corrected by channel coding, cyclic redundancy check (CRC) bits of size 24, 16, 12, 8, or 0 (depending on the block size) are added to every transport block. To form data blocks suitable for channel coding, the transport blocks in a transmission time interval (TTI) are serially concatenated. TTI is the time unit from set {5 ms, 10 ms, 20 ms, 40 ms, and 80 ms} for the base station, known as eNodeB in 3G and 4G, to schedule UL and DL data transmissions. After concatenation, if the length X is larger than the maximum size of a data block, then the concatenated block is segmented and zero-padded to suit the appropriate channel coding scheme. The available channel coding schemes are convolutional coding, turbo coding, or no coding. The maximum size of a data block is $Z = 504$ for convolutional coding, $Z = 5114$ for turbo coding, and unlimited for no coding case. If the length of the concatenated block is not multiple of Z, filler bits, by default zeros, are added to the beginning of the first block. Hence, $K = \lceil X/Z \rceil$. Moreover, filler bits are added when the blocklength is less than 40 and turbo coding is employed. Hence, $K = 40$ in this case.

TABLE 5. Coding parameters in UMTS (3G).

Coding Scheme	Code Rate	Data Block Size, K
Convolutional Coding	1/2 & 1/3	≤ 504
Turbo Coding	1/3	$[40, 5114]$
No channel coding	1	unlimited

Convolutional coding with constraint length 9 is performed by generator polynomials (in octal) $G_0 = 561$ and $G_1 = 753$ for code rate 1/2 and $G_0 = 557$, $G_1 = 663$, and $G_2 = 711$ for code rate 1/3. Before encoding, 8 zero-value tail bits are added. For turbo coding with code rate 1/3, a parallel concatenated convolutional coding with two constituent encoders and transfer function $G(D) = [1, g_1(D)/g_0(D)]$, where $g_0(D) = 1 + D^2 + D^3$ and $g_1(D) = 1 + D + D^3$ are employed.

The coding parameters used in 3G are listed in Table 5. Convolutional coding is used for short packets in control channels' signaling, while turbo coding is employed for longer packets. To realize a variable transmission rate and adjust the amount of data to fit the radio frames, rate-matching is performed as well.

D. FOURTH GENERATION (4G)

In 2008, the framework and objectives for the next generation of mobile communications were recommended in the International Mobile Telecommunications-Advanced (IMT-Advanced) by the ITU-R [4] marketed as 4G. Recall from the previous section that LTE is considered pre-4G as it did not meet all the 4G requirements defined in IMT-Advanced. The 4G specifications were determined in 3GPP Release 10. This generation, known as LTE-Advanced (LTE-A), employed additional spectrum and frequency bands, namely around 600 MHz, 700 MHz, 1.7/2.1 GHz, 2.3 GHz, and 2.5 GHz. The LTE channel bandwidth can be 1.4, 3, 5, 10, 15, or 20 MHz. Taking into account about 10% of the bandwidth as a guard band and 15 kHz for OFDM frequency spacing, the effective bandwidth for the 20 MHz bandwidth will be 18 MHz resulting in 18 MHz/15kHz = 1200 subcarriers. Now, since a physical resource block (PRB) consists of 12 consecutive subcarriers for one time slot (0.5 ms), then 1200/12 = 100 PRBs will be available for 20-MHz bandwidth.

Three technologies from LTE-Advanced - namely carrier aggregation, up to 4×4 MIMO in uplink and up to 8×8 MIMO in downlink, and 256QAM modulation in downlink - if used together and with sufficient aggregated bandwidth, can deliver maximum peak downlink speeds approaching 1 Gbps. Such networks are often described as 'Gigabit LTE networks' mirroring a term that is also used in the fixed broadband industry. LTE-A carrier aggregation (CA) is a key technique used to increase the data rate and the capacity in both uplink and downlink by combining up to 5 individual carriers, called component carriers (CC), either in the same or different bands. Note that a similar technique was actually employed in HSPA+ named dual carrier (DC), as discussed earlier in the 3G section. The carrier aggregation (with 3 downlink carriers and 2 uplink carriers) is supported in both duplex schemes; frequency division duplex (FDD) and time division duplex (TDD). By considering the aggregation of up to 5 carriers, the maximum bandwidth of 20 MHz in LTE increases to 100 MHz in LTE-A. Note that mobile worldwide interoperability for microwave access (Mobile WiMAX) based on the IEEE 802.16m standard (a.k.a. WirelessMAN-Advanced) intended to compete with LTE-A as a candidate for 4G cellular networks by fulfilling the ITU-R IMT-Advanced requirements. However, it was not well established at the time.

LTE-Advanced Pro (LTE-AP), also known as 4.5G, was 3GPP Release 13 as an evolution of the LTE-A cellular standard that supports data rates in excess of 3 Gbps using 32-carrier aggregation. It also introduces the concept of License-Assisted Access, which allows for the sharing of licensed and unlicensed spectrum. Similarly to LTE-A, the aggregation of up to 32 carriers in LTE-AP increases the maximum bandwidth of 20 MHz in LTE up to 640 MHz. Furthermore, it incorporates several new technologies that were later used in 5G, such as 256-QAM, Massive MIMO, LTE-Unlicensed, and LTE IoT. Hence, 4.5G facilitated the early migration of existing networks to enhancements promised with the full 5G standard. Recall that 2.75G and 3.9G played a similar role in migrations from 2G to 3G and from 3G to 4G, respectively. To reduce energy consumption, 4G networks were designed with improved network optimization techniques, including strategies such as expansion of the cell range, optimization of the sleep mode, and switching of base stations to minimize power consumption while maintaining coverage and quality of the network.

Channel coding of LTE/LTE-A reuses the turbo codes from 3G traffic channels and convolutional codes from 3G physical control. In contrast to the 3G channel coding, turbo codes were enhanced with improved performance and lower decoding complexity. In addition, tail-biting convolutional codes were introduced to reduce the overhead as previously zero tail bits were added for termination.

The channel coding scheme used for the transport blocks in 4G, which is specified in Evolved Universal Terrestrial Radio Access (E-UTRA), is turbo coding with code rate of $R = 1/3$. This scheme incorporates two 8-state constituent encoders (identical with 3G) and a contention-free quadratic permutation polynomial (QPP) interleaver. Furthermore, trellis termination is used for turbo coding. Prior to turbo coding, transport blocks are segmented into byte-aligned segments with a maximum information block size of 6144 bits.

According to the 3GPP's technical specification TS 36.212 [14], the transport blocks (TBs) are protected by appended CRC bits with a length of 24 (calculated by type-A generator polynomial, CRC24A). The detected error by CRCs is reported to higher layers. The transport block is segmented into code blocks (CBs) of no larger than 6144 bits, where each CB is protected by type-B 24-bit CRC

TABLE 6. Coding parameters in E-UTRA (4G).

Coding Scheme	Code Rate	Data Block Size, K
Tail-biting Convolutional Coding	1/3	
Turbo Coding	1/3	$[40, 6144]$
Block Code	1/16, variable	
Repetition Code	1/3	

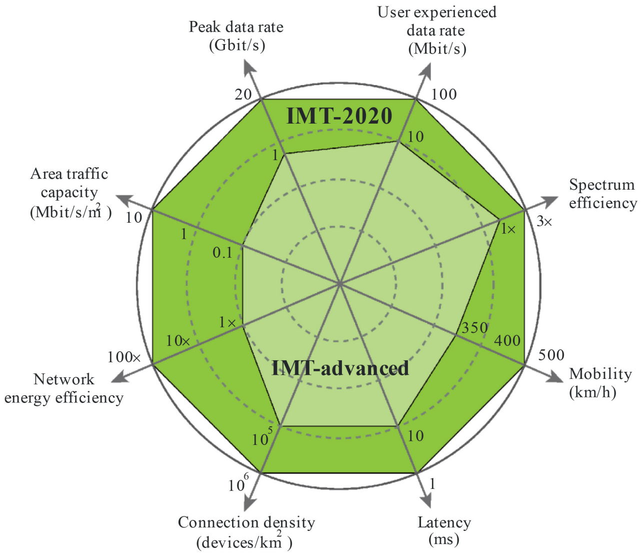


FIGURE 3. Key capabilities from IMT-Advanced (for 4G) to IMT-2020 (for 5G)[5].

(CRC24B). If the code block is smaller than 40 bits, filling bits are appended. The CRC bits for control channels are shorter as broadcast channel (BCH) and physical downlink control channel (PDCCH) use 16-bit CRC, while physical uplink control channel (PUCCH) employs 8-bit CRC. The channel coding scheme for the aforementioned control channels is tail-biting convolutional coding (TBCC) whereas the transport channels (TrCH) use turbo coding. Tail-biting convolutional coding with constraint length 7 is performed by generator polynomials (in octal) $G_0 = 133$, $G_1 = 171$, and $G_2 = 165$ for the code rate 1/3. The details of the coding schemes used in 4G are summarized in Table 6.

E. FIFTH GENERATION (5G)

The ITU framework and objectives for 5G cellular networks, devices, and services, were recommended in the IMT-2020 vision by ITU-R in 2015 [5], to address the requirements of emerging applications. Fig. 3 summarizes the requirements in terms of 8 KPIs. As can be seen, throughput (the peak data rate) of up to 20 Gbps in downlink and 10 Gbps in uplink aims to answer the growing demand for high data rates. This is a significant increase compared to IMT-Advanced (requirements for 4G/LTE-A in Release 14), which offers 1 Gbps in downlink and 50 Mbps in uplink. The latency of 1 ms (compared to 30 - 50 ms in 4G) will allow near real-time responses, and the connection density of 1000 devices per square kilometer (100 times more than 4G) will meet the growing demand for IoT devices and sensors.

5G specifications have been integrated into 3GPP releases 15, 16, and 17. Full deployment of the 5G capabilities defined in IMT-2020 requires the implementation of totally new networks, significant investments by operators, and considerable elapsed time to enable a full rollout. To ease the migration path, 3GPP defined 5G NR non-stand alone (NSA) in release 15 leveraging existing LTE infrastructure. The throughput of existing macro cells can be increased by adding additional MIMO layers, and the spectrum can be dynamically shared between 4G LTE and 5G NR. Operators can use existing spectrum in the so-called "MIMO sweet spot", around 3.5GHz.

The 5G frequency spectrum is divided into frequency range 1 (FR1) spanning from 450 MHz to 7.125 GHz (previously up to 6 GHz, hence this range is still known as sub-6 GHz), and millimeter-wave (mm-Wave) range (FR2) spanning from 24.25 GHz to 52.6 GHz, as well as unlicensed spectrum. The maximum bandwidths in sub-6 GHz and the mm-Wave range are 100 MHz and 400 MHz, respectively. There are 7 subcarrier spacings as $\Delta f = 2^\mu \cdot 15$ kHz, $\mu = 0, \ldots, 6$. The subcarrier spacing of $\Delta f = 2^0 \cdot 15$ (the same as LTE) and $2^1 \cdot 15$ kHz is used only in sub-6 GHz, and the subcarrier spacing of $2^3 \cdot 15$ kHz is used only in the mm-Wave range, while 60 kHz can be used in both ranges. Thus, 5G includes the previous cellular spectrum and further expands it. The additional spectrum addresses the physical limitations associated with throughput and bandwidth. 4G band plans accounted for 5 MHz to 20 MHz of bandwidth per channel, whereas the 5G FR1 allows for 5 to 100 MHz of bandwidth per channel. As bandwidth is directly proportional to maximum throughput, the 5 times increase in bandwidth relates to roughly a 5 times increase in throughput. Furthermore, 3GPP Release 15 established new waveforms and the addition of $\pi/2$ binary phase shift keying (BPSK) as a modulation method in the uplink with the aim of further reducing the peak-to-average power ratio (PAPR) and increasing the power efficiency of the RF amplifier at lower data rates. Additional waveforms are discrete Fourier transform spread orthogonal frequency division multiplexing (DFT-S-OFDM) for FR1 and the cyclic prefix OFDM (CP-OFDM) for FR2.

The key technologies reviewed above were introduced to meet the requirements of new use cases. The 5G services and applications can be classified into three main use scenarios: enhanced mobile broadband (eMBB), ultra-reliable low-latency communications (URLLC) and massive machine-type communications (mMTC), as illustrated in Fig. 4.

- eMBB focuses on delivering significantly higher data rates, increased network capacity, and improved user experiences compared to previous generations of mobile networks. It enables applications that require high-speed and high-bandwidth connectivity, such as ultra-high-definition video streaming, virtual reality (VR), augmented reality (AR), online gaming, and immersive multimedia experiences. eMBB provides

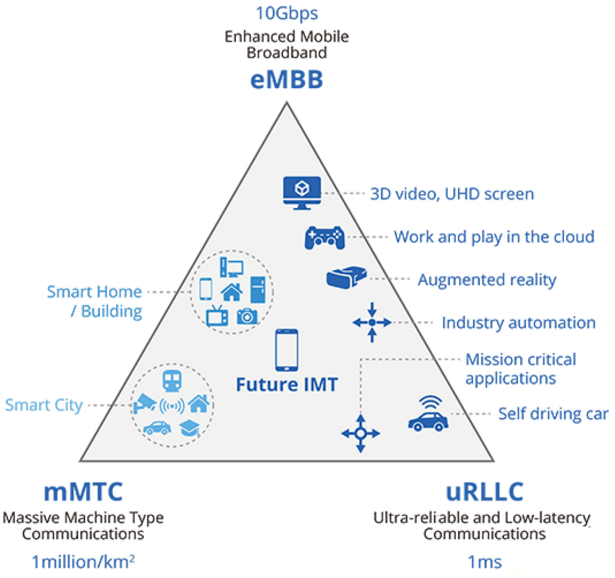

10Gbps
Enhanced Mobile Broadband
eMBB
3D video, UHD screen
Work and play in the cloud
Augmented reality
Smart Home / Building
Industry automation
Mission critical applications
Future IMT
Smart City
Self driving car
mMTC
Massive Machine Type Communications
1million/km²
uRLLC
Ultra-reliable and Low-latency Communications
1ms

FIGURE 4. 5G use cases [5].

users with ultra-fast downloads, seamless video streaming, and enhanced browsing capabilities. However, note that the latency requirements of eMBB depend on the specific service types. For example, stream-type services require a high data rate and can tolerate a delay of 50 to 100 ms. Interactive services, on the other hand, have a more stringent requirement for latency, e.g., 5 to 10 ms.

- URLLC in 5G is a set of technologies and features designed to provide extremely reliable and low-latency communication for mission-critical applications. URLLC ensures that time-sensitive and critical data are transmitted with ultra-low latency and high reliability. Some of the key technologies involved in URLLC are network slicing, edge computing, beamforming, and massive MIMO, quality-of-service (QoS) mechanisms, redundancy, and error correction, time-sensitive networking (TSN), and network synchronization.
- MTC in 5G involves various technologies and features specifically designed to provide efficient massive connectivity for IoT devices. It is needed to support a large number of devices that transmit data sporadically and without coordination with other devices or the network, that is, asynchronously. This can lead to interference between devices and the network, leading to a reduction in the data rate and reliability of communications. Furthermore, since MTC devices are often powered by batteries or energy harvesting devices, they need to have low power consumption, which requires a low computational complexity transmitter/receiver, so that they can operate for long periods of time without having to be recharged. Applications and services of MTC include both low-rate and high-rate data collection, and some delay-insensitive control-type services. Here are some key technologies involved in MTC:
  - NB-IoT: NB-IoT is a low-power, wide-area (LPWA) technology optimized for IoT applications. It enables long-range communication, extended battery life, and deep indoor coverage. NB-IoT operates in licensed spectrum, offering improved security and quality of service for IoT devices.
  - LTE-M (Long-Term Evolution for Machines): LTE-M is another LPWA technology in 5G designed for IoT applications. It provides higher data rates compared to NB-IoT, making it suitable for applications that require more bandwidth. LTE-M offers improved mobility support, voice communication capabilities, and reduced power consumption.
  - Device-to-device (D2D) communication: D2D communication in 5G enables direct communication between nearby IoT devices without routing through the network infrastructure. It improves communication efficiency, reduces latency, and conserves network resources. D2D communication is particularly beneficial for applications that require local coordination and peer-to-peer interaction.

5G supports new channel coding schemes. According to 3GPP's technical specification TS 38.212 [15], 5G incorporates Low-Density Parity-Check (LDPC) codes for data channels including physical downlink shared channel (PDSCH) and physical uplink shared channel (PUSCH), replacing turbo codes in 4G. Likewise, for control channels (to protect the downlink control information (DCI), the uplink control information (UCI), and the system information in the physical broadcast channel (PBCH)), Polar codes are introduced in lieu of the TBCC used in 4G. The key advantages of LDPC codes (compared to turbo codes) are improved performance with very low error floors, reduced decoding complexity and latency, better power and area efficiency, and support of multi-Gbps data rates. Polar coding yields better performance at moderate payload sizes (in the order of $K \leqslant 250$ bits); however, it comes at the cost of higher complexity compared to TBCC. Note that the minimum supported payload size K in 5G polar codes is 12 bits. For the control information sequence $K < 12$ in the uplink, the short block codes employed in 4G, namely, repetition codes, simplex codes and Reed-Muller codes, are used.

Targeting good performance and decoding latency for the full range of code rates and information block sizes, 5G supports two LDPC base matrices (which are, in turn, constructed using a photograph, see Section VI for more information). Base matrix 1 was optimized for large information block sizes K and high code rates $R \geqslant 1/3$. On the other hand, base matrix 2 is suitable for small information block sizes and lower code rates than base matrix 1, down to $1/5$, which is lower than the code rate of 4G turbo codes (where for lower than $1/3$, repetition is used). Fig. 5 illustrates the switching points between the two base matrices based on the pair of (K, R). For K larger than

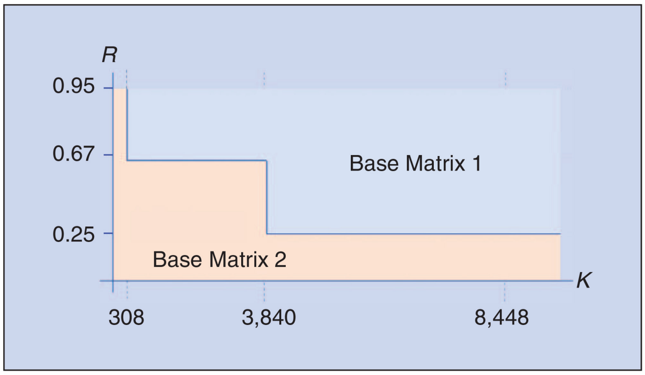


FIGURE 5. Key capabilities from IMT-Advanced to IMT-2020 [16].

TABLE 7. Channel coding schemes in 5G.

	Channel	Coding Scheme	Info. block, A	CRC+PC	Encoded block, G
Downlink	PDSCH	LDPC	[80,8448]	16,24B	$[384, 6480]$
	PDCCH	Polar	$[12, 140]$	24C	$[A+24, 8192]$
	PBCH	Polar	32	24C	864
Uplink	PUSCH	LDPC	$[48, 3840]$	24A,24B	$[192, 15360]$
	PUCCH	Polar	$[12, 19]$	6+3	$[A+9, 8192]$
			$[20, 1706]$	11	$[A+11, 16385]$

the maximum information block size of the base matrices (shown in Table 7), code block segmentation is used.

5G polar codes employ 6, 11, and 24 CRC bits for error detection with the corresponding generator polynomials $g_{\mathrm{CRC6}}(D) = D^6+D^5+1$, $g_{\mathrm{CRC11}}(D) = D^{11}+D^{10}+D^9+D^5+1$, and $g_{\mathrm{CRC24C}}(D) = D^{24}+D^{23}+D^{21}+D^{20}+D^{17}+D^{15}+D^{13}+D^{12}+D^8+D^4+D^2+D+1$. However, the 5G LDPC codes use 16 and 24 CRC bits with generator polynomials $g_{\mathrm{CRC16}}(D) = D^{16}+D^{12}+D^5+1$, $g_{\mathrm{CRC24A}}(D) = D^{24}+D^{23}+D^{18}+D^{17}+D^{14}+D^{11}+D^{10}+D^7+D^6+D^5+D^4+D^3+D+1$, and $g_{\mathrm{CRC24B}}(D) = D^{24}+D^{23}+D^6+D^5+D+1$.

Table 7 lists the parameters of the coding schemes in the 5G standard.

In a recent milestone, 3GPP finalized Release 18, signaling the advent of 5G Advanced. The features integrated into 5G Advanced are poised to elevate the performance of 5G networks, providing augmented support for services such as extended reality (XR), indoor positioning, and non-terrestrial networks. Building upon the foundation laid by 5G Advanced, Release 19 will focus on enhancing performance and addressing vital requirements in commercial 5G deployments. The evolution of 5G Advanced is expected to progress throughout this decade. Concurrently, standardization efforts for 6G are projected to intensify with Releases 20 and 21 commencing in 2025, aligning with the timeline depicted in Fig. 1.

TABLE 8. The major contribution of the technologies used in 3G to 5G standards to the KPIs.

	3G	4G	5G	Reliability	Data Rate	Connection Density	Spectral Efficiency	Energy Efficiency	Latency
Channel Coding	X	X	X	X					
Adaptive Modul. & Coding	X	X	X	X	X		X		
Higher Order Modulation	X	X	X		X		X		
HARQ	X	X	X	X					X
Packet Switching	X							X	X
IP-Based Architecture		X	X						X
Enhanced Packet Core		X	X						X
Discontinuous Tx/Rx		X	X					X	
Power-Saving Techniques			X			X		X	
New Network Architecture	X	X	X			X		X	
QoS Management		X	X						X
Dynamic Spectrum Sharing			X					X	
Network Optimization			X					X	
Edge Computing			X						X
Network Slicing			X	X		X			
Small Cells		X	X	X	X	X			X
Network Densification			X					X	
Wideband CDMA	X			X	X	X	X		
OFDMA		X	X	X	X	X	X		
Carrier Aggregation		X	X		X				
MIMO		X	X	X	X	X	X		
Beamforming		X	X	X					
mm-Wave			X		X				
Massive MIMO			X	X	X	X	X		
Narrowband IoT			X			X		X	
LTE-M			X		X	X		X	

Table 8 summarizes the role of technologies introduced in every generation, which were discussed in this section, in improving the KPIs of the mobile communication network. Note that the improvement is considered relative to the previous generation. As can be seen, the major contribution of channel coding is improving channel reliability in a wireless link, although a low-complexity decoder can also improve energy efficiency and latency. Furthermore, higher order modulation, adaptive modulation, and coding schemes, and HARQ, in collaboration with channel coding, can influence other KPIs except for connection density.

IV. 6G USE CASES, REQUIREMENTS AND CHALLENGES

Since the early discussions on 6G, various organizations and projects have been identifying the potential 6G use cases. Among them, the 6G Flagship research program led the way, publishing a white paper in 2019 [17] that identified the first set of use cases for the various types of devices expected at the time of 6G commercialization. The major communications companies followed suit, releasing their 6G white papers since 2020. Collaborative European projects also began in 2021, including Hexa-X, which aims to develop a 6G vision and an intelligent fabric of technology enablers.

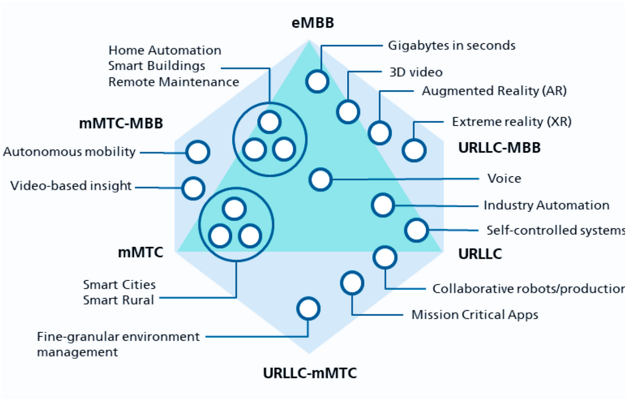


FIGURE 6. 6G use cases [20].

TABLE 9. Comparison between KPIs of 5G and 6G.

Key Performance Indicator	5G	6G
Peak Data Rate	20 Gb/s	1 Tb/s
Experienced Data Rate	0.1 Gb/s	1 Gb/s
Peak Spectral Efficiency	30 b/s/Hz	60 b/s/Hz
Exp. Spectral Efficiency	0.3 b/s/Hz	3 b/s/Hz
Maximum Bandwidth	1GHz	100GHz
Area Traffic Capacity	10Mb/s/m^2	$1\ \text{Gb/s/m}^2$
Connection Density	10^6devices/km^2	10^7devices/km^2
Energy Efficiency	N/A	1 Tb/J
Latency	1 ms	100us
Jitter	N/A	1us
Reliability	1×10^{-5}	1×10^{-7}
Mobility	500Km/h	1000Km/h

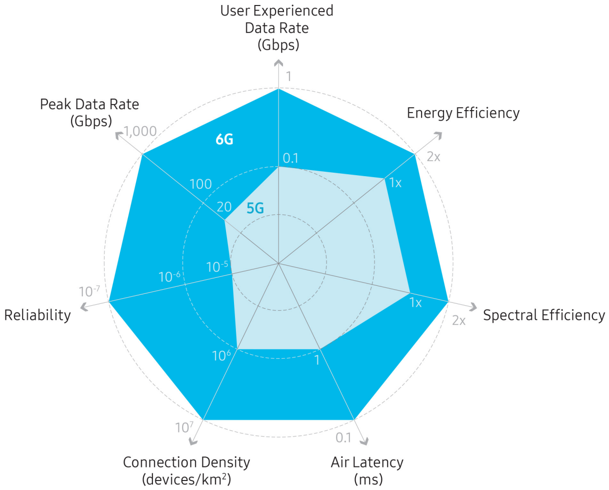


FIGURE 7. The key performance requirements from 5G to 6G [21].

One of Hexa-X's first deliverables was a comprehensive set of 6G use cases [18]. Other organizations, such as the Next Generation Mobile Network Alliance (NGMN) [19], have also published white papers on 6G use cases. The use cases suggested for 6G are shown in Fig. 6. While the potential use cases for 6G technology are still in the early stages of development, the current deployment of 5G networks can help us to identify a set of highly promising directions that current commercial networks cannot meet.

The classification of 6G use cases illustrated in Fig. 6 aims to change the way we think about network transformation. This does not exclude the further evolution of 5G use cases, but rather complements them with a longer-term perspective that addresses the communication needs of 2030s. Starting with 4G MTC and 5G URLLC, we have seen a strong diversification of service classes beyond simply increasing network capacity. This has led to the need for specific characteristics of the network that cannot be met with the same infrastructure. While we still expect to see more devices, higher network capacity, and more reliability or lower latency, the next generation of applications will require a combination of these capabilities. For example, new eMBB services such as Augmented Reality (AR) and eXtended Reality (XR) will require high reliability, low latency, and network capacity. Similarly, mMTC will need to extend itself to high reliability and low latency to handle fine-grained automatic environment management and increase network capacity for autonomous mobility and video-based insight generation.

Based on potential requirements, the key driver for 6G would be the extended network capacity, which may require new Terahertz (THz) spectrum-based technologies. This is necessary to support capacities of 4 Tbit/s for AR/XR, the under 100 μs delay for industrial or holographic presence, a 7-nines (i.e., 99.99999%) reliability, or localization with a precision of less than 1 cm. Even with increasing computing capacity in terminals, the network will need to be significantly enhanced to support these use cases. Note that mobile data traffic has increased by 50%-100% every year over the past decade. This growth is expected to continue further as connected devices, in particular sensors, cars, and home devices, become more and more popular, and the demand for emerging applications will grow exponentially. This alone implies that 6G should address the volume of mobile traffic of from 100 times to 1000 times more than 5G. The increase in the volume of data traffic would require increased energy consumption at the base stations and network nodes. Therefore, we need to significantly improve the energy efficiency of connected devices by reducing the energy use per transported/processed data to keep the power consumption per device comparable to 5G. As some new use cases such as AR/VR, and holographic telepresence or the need for 8K resolution in communications are expected to come into play, the increase in data volume is followed by the need for a remarkably higher end-user experienced data rate of up to 100 Gbps. However, the increase in the number of connected devices would imply the need for an increase in device density per square meter, and consequently an increase in the required capacity to address the number of devices exchanging high data volume. As a result, the cells should be smaller to meet these demands. An example could be a stadium full of spectators using AR glasses.

Toward defining the official specifications for 6G, ITU-R has already formed a group on IMT toward 2030 and beyond, with the aim of completing the study on the 6G vision by the end of 2023. The development process is expected to begin with technical studies in 2026 and the first specifications are aimed to be released by 2028. The 6G networks are expected to be commercially deployed from 2030 [22].

A. CHALLENGES IN CHANNEL CODING FOR 6G

The upcoming generation of cellular networks is expected to impose even more rigorous demands in terms of higher data rates, increased reliability, and lower latency when compared to the current 5G networks. Hence, it is imperative to design new channel coding schemes or improve the ones available to meet those future requirements and KPIs. In the subsequent discussion, we discuss some of the formidable challenges associated with this endeavor.

First, the complexity of codes that perform well under practical constraints such as limited processing delay and high spectral efficiency is still a major hurdle for low-power implementations in integrated circuits. There is a serious need for new methods that simplify code design, construction, storage, and decoder implementation. In particular, the new channel coding schemes will be required to encode and decode data at very high speeds to support the high data rate, for example 1 Tb/s, in 6G. Meanwhile, both code structures and low-complexity decoders should be designed to achieve performance closer to the Shannon limit than the coding schemes in 5G. In addition, channel coding in next-generation cellular networks will be required to handle excessive interference in the presence of a massive number of users and devices, along with signal detection at the receiver end.

The other major research challenge is to design robust channel coding schemes with short blocklength for delay-sensitive services. From an information-theoretic point of view, short blocklength codes are less reliable, such that error-free transmission is no longer guaranteed. Hence, an increase in the error probability can increase the need for retransmissions, which is also not desirable for time-sensitive applications requiring ultra-low latencies. On the other hand, codes with longer block lengths have a larger time/computational complexity, which implies an increase in transmission and processing latency. In addition, the optimal decoders for short blocklength codes generally have high computational complexity, leading to large processing latency. To this end, new short blocklength codes with low-complexity decoders need to be designed to meet the target error probability while satisfying the stringent latency constraints.

As integrated/joint/coexisting sensing and communication is envisioned for 6G, channel coding faces a number of challenges when employed in communication-centric sensing and sensing-centric communications. These challenges include: 1) balancing communication and sensing performance: channel coding schemes need to balance between communication performance (e.g., data rate, reliability) and sensing performance (e.g., accuracy, resolution). For example, a channel coding scheme optimized for communication performance may not be well suited for sensing-centric communications, where sensing accuracy is paramount. 2) dealing with noise and interference: Joint sensing and communications applications are often subject to high levels of noise and interference. This can make it difficult for channel coding schemes to reliably decode the transmitted signal, which can affect both communication and sensing performance. 3) supporting dynamic channel conditions: The channel conditions in joint sensing and communication applications can be very dynamic, due to factors such as the movement of sensor nodes and the presence of obstacles in the environment. This can make it difficult for channel coding schemes to maintain reliable communication and sensing. 4) limited resources: Sensor nodes often have limited resources, such as battery life and processing power. This can make it difficult to implement complex channel coding schemes.

B. POTENTIAL CODING SCHEMES FOR 6G

Improving the channel coding scheme results in the improvement of the reliability (directly) and the spectral efficiency (directly) by allowing more data to be transmitted per unit of (frequency) bandwidth. Moreover, the adoption of higher-order modulation further improves the data rate and spectral efficiency. Hence, the coding and modulation schemes have impacts on three key performance indicators; reliability, data rate, and spectral efficiency. These three performance indicators must be dramatically improved in 6G according to Section IV to make the new applications and use cases possible. The responsibility for improving these indicators is not only on channel coding and modulation schemes, as other components in the physical layer also play a role. Historically, developing and adapting a new coding scheme into a standard takes more than a decade. The reason lies in the difficulty of advancing this field as we approach the theoretical performance bound. It is not a long time since the 3GPP adopted new coding schemes in 2016. We will consider the coding schemes already in the 5G standard in Sections VI and VII; namely, LDPC codes and polar codes. We will investigate these coding schemes in detail, review recent advances, and compare them from different angles. We also consider turbo codes and convolutional codes used in 3G and 4G in Section V. The coding schemes in the upcoming sections will be presented in chronological order, reflecting their development and incorporation into standards over time. Furthermore, other coding schemes such as lattice codes, rateless codes, and sparse regression codes are reviewed in Section IX.

C. FINITE BLOCK LENGTH CODING PERFORMANCE

The Shannon limit is the theoretical maximum rate, which is called *channel capacity*, that can be achieved with arbitrarily small errors by using a code with very long (infinite) blocklength. However, practical systems only allow finite blocklength coding such that the Shannon limit may not be achievable. That is, as we decrease the blocklength, the coding gain is reduced, and consequently the gap to the Shannon limit increases. In this section, we review an approximation called *normal approximation (NA)* for the performance of finite blocklength codes [23].

Consider the real AWGN channel with noise variance 1. For this channel, consider using a length-n code $\mathcal{X}$ with each codeword $\boldsymbol{x} \in \mathcal{X}$ satisfying the maximal power constraint $\|\boldsymbol{x}\|^2 \leqslant n\rho$, where ρ denotes the power or the Signal-to-Noise Ratio (SNR). Under the constraint that the average decoding error probability or the block error rate (BLER) does not exceed ϵ, the following rate is achievable [23]

$$R \approx C(\rho) - \sqrt{\frac{V(\rho)}{n}} Q^{-1}(\epsilon) + \frac{\log_2(n)}{2n}, \tag{1}$$

where $Q(x) = \int_x^\infty \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}} dt$ is the standard Q-function, $C(\rho)$ and $V(\rho)$ are the Gaussian capacity and dispersion functions, respectively, and

$$V(\rho) = (\log_2 e)^2 \cdot \frac{\rho(\rho+2)}{2(\rho+1)^2} \text{ bits}^2 \text{ per channel use.} \tag{2}$$

Note that (1) is known as the NA.

The NA has been shown to be a valid asymptotic approximation for the achievability bound (i.e., random coding union bound [23, Th. 16]) and converse bound (i.e., metaconverse [23, Th. 26]). The achievability bound is intended as a performance that can be achieved by a suitable encoding/decoding couple, while a converse bound is intended as a performance that outperforms any choice of the encoding/decoding couple. However, the computation of the achievability and converse bounds becomes very difficult when n is not small. Hence, the NA is often used as the performance benchmark due to its simpler computation complexity.

Now, assume that each transmitted symbol x of the length-n sequence $\boldsymbol{x}$ is i.i.d. over the BPSK modulation and y is the received noisy symbol. This channel model is also known as the binary-input AWGN (BI-AWGN) channel. The BLER upper bound ϵ can be evaluated by rearranging (1) as [24], [25]

$$\epsilon \approx Q\left(\sqrt{\frac{n(C_{\mathrm{b}}(\rho) - R) + \frac{\log_2 n}{2}}{nV_{\mathrm{b}}(\rho)}}\right), \tag{3}$$

where $R = \frac{k}{n}$ is the code rate with k being the source blocklength, $C_{\mathrm{b}}(\rho)$ and $V_{\mathrm{b}}(\rho)$ are the BIAWGN capacity and dispersion functions, respectively. Let $P_{Y|X}(y|x)$ and $P_Y(y)$ denote the channel transition probability and the probability density function of the channel output Y, respectively. Specifically, $C_{\mathrm{b}}(\rho)$ and $V_{\mathrm{b}}(\rho)$ can be computed by using the information density $i(X;Y) = \log_2 \frac{P_{Y|X}(y|x)}{P_Y(y)}$ such that by [25] we have

$$\begin{aligned} C_{\mathrm{b}}(\rho) &= \mathbb{E}[i(X;Y)] && (4)\\ &= \frac{1}{\sqrt{2\pi}} \int e^{-\frac{z^2}{2}} \left(1 - \log_2\left(1 + e^{-2\rho+2z\sqrt{\rho}}\right)\right) dz, && (5)\\ V_{\mathrm{b}}(\rho) &= \mathrm{Var}[i(X;Y)] && (6)\\ &= \frac{1}{\sqrt{2\pi}} \int e^{-\frac{z^2}{2}} \left(1 - \log_2\left(1 + e^{-2\rho+2z\sqrt{\rho}}\right) - C_{\mathrm{b}}(\rho)\right)^2 dz. && (7) \end{aligned}$$

The extension of NA to high-order modulations has been investigated in [26], [27].

Given the code parameters and the channel SNR, the design of coding schemes should target the BLER predicted by the NA. However, as we shall observe in the following sections, some well-known codes can perform very close to the NA when the blocklength is short.

V. TURBO CODES

Turbo codes or parallel concatenated convolutional codes (PCCCs) were introduced in 1993 by Berrou, Glavieux, and Thitimajshima [12]. The invention of turbo codes marked a major breakthrough in coding theory [28]. It was the first class of codes that was practically demonstrated to achieve the near-Shannon-limit performance with modest decoding complexity. Owing to the close-to-capacity performance and moderate decoding complexity, turbo codes have been the standard channel coding in standard 3G [29], [30] and 4G mobile communication standards [31]. In addition, turbo codes have also been adopted in the IEEE 802.16 WiMAX (worldwide interoperability for microwave access) [32] and DVB-RCS2 (2nd generation digital video broadcasting - return channel via satellite) [33] standards. For the list of standards that have adopted turbo codes, we refer the reader to Table IV in [34, Ch. 5.3.3]. During the study phase of 5G NR, an enhanced version of turbo codes with better waterfall and error floor performance than LTE turbo codes, was one of the candidate channel coding schemes [35]. Besides error correction coding, the concept of the turbo principle has been applied to the decoding of product codes [36], [37] and iterative detection [38], [39].

In this section, we first review the properties of turbo codes and their component codes, i.e., convolutional codes. Then, we provide a comprehensive survey covering state-of-the-art designs on interleavers, puncturing patterns, and decoding algorithms. Several variants of turbo codes and future research directions are presented in the end.

A. CONVOLUTIONAL CODES

Convolutional codes [40] are the building blocks of turbo codes. Besides, convolutional codes were adopted as the coding schemes in 3G UMTS [29], [30] and 4G LTE standards [31] for control channels. Unlike block codes, the information and codeword sequences for convolutional codes may or may not be terminated [41].

1) ENCODING OF CONVOLUTIONAL CODES

A rate-k/n convolutional code is specified by $k \times n$ generator polynomials, which form the generator matrix

$$\boldsymbol{G}(D) = \begin{bmatrix} \boldsymbol{g}_1^{(1)}(D) & \boldsymbol{g}_1^{(2)}(D) & \ldots & \boldsymbol{g}_1^{(n)}(D) \\ \boldsymbol{g}_2^{(1)}(D) & \boldsymbol{g}_2^{(2)}(D) & \ldots & \boldsymbol{g}_2^{(n)}(D) \\ \vdots & \vdots & \ldots & \vdots \\ \boldsymbol{g}_k^{(1)}(D) & \boldsymbol{g}_k^{(2)}(D) & \ldots & \boldsymbol{g}_k^{(n)}(D) \end{bmatrix}, \tag{8}$$

where D is known as a unit delay operator. Consider k input sequences $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k$. For $i \in \{1, \ldots, k\}$, the i-th input sequence can be represented by $\boldsymbol{u}^{(i)}(D) = \sum_{l=0}^{\infty} u_l^{(i)} D^l$. A collection of these sequences can be arranged into $\boldsymbol{u}(D) = [\boldsymbol{u}^{(1)}(D) \ \ldots \ \boldsymbol{u}^{(k)}(D)]$. The encoder output consists of n polynomials $\boldsymbol{c}(D) = [\boldsymbol{c}^{(1)}(D) \ \ldots \ \boldsymbol{c}^{(n)}(D)] = \sum_{j=1}^{n} D^{j-1}\boldsymbol{c}^{(j)}(D^n)$, which is generated by

$$\boldsymbol{c}(D) = \boldsymbol{u}(D)\boldsymbol{g}(D). \tag{9}$$

For $j \in \{1, \ldots, n\}$, the j-th code sequence is generated by

$$\boldsymbol{c}^{(j)}(D) = \sum_{i=1}^{k} \boldsymbol{u}^{(i)}(D)\boldsymbol{g}_i^{(j)}(D). \tag{10}$$

The memory of the encoder is defined as

$$m \triangleq \max_{i \in \{1,\ldots,k\}, j \in \{1,\ldots,n\}} \left\{ \deg\left(\boldsymbol{g}_i^{(j)}(D)\right) \right\}. \tag{11}$$

For turbo codes, *recursive systematic convolutional (RSC)* codes are mostly employed as the component codes. A recursive convolutional code means that at least one entry in $\boldsymbol{G}(D)$ is a rational function or equivalently the encoder realization has feedback. In addition, it is common to represent the generator matrix in *octal* notation. For example, the generator polynomial for the LTE turbo code $G(D) = [1, \frac{1+D+D^3}{1+D^2+D^3}]$ can be represented by $[1, 15/13]_8$. In addition to turbo codes, convolutional codes have been used as the precoding stage for polar codes, resulting in polarization-adjusted convolutional (PAC) codes as shown in Section VII-H.

2) TRELLIS DIAGRAM AND TERMINATION

A convolutional encoder can be modeled as a finite state machine for which the input-output relation and state transition can be described by a state-transition table or a state diagram. The time evolution of the state diagram of the state machine can be described by a trellis diagram [41]. For a rate-k/n convolutional code with memory m and K information symbols, the trellis diagram contains 2^m states and $\frac{K}{k} + 1$ time instants. The connection between two states is called a branch. Essentially, it visualizes how the output bits of the current state s are computed based on the input bits and the previous state s.

In the case of packet transmission, termination of convolutional codes is required. There are several approaches for termination. The first method is *direct truncation*, which stops the encoding process when all the information bits have been applied to the encoder input. Although it does not have rate loss, the final state is unknown to the decoder, leading to some performance degradation on the convolutional code. That said, this method is employed for the convolutional precoding of PAC codes, see Section VII-H. The second method is called *zero termination* [42]. It adds m additional bits to the original message to force the encoder trellis to the all-zero state. The turbo codes in 3G UMTS [29], [30] and 4G LTE standards [31]. However, this termination method leads to a small rate loss. In addition, the termination bits are not turbo coded [34, Ch. 3.2]. The third approach is the *tail-biting termination* [43], which does not incur any rate loss. Specifically, the initial state of the trellis diagram is obtained based on the input sequence such that the final state is the same as the initial state. Tail-biting termination has been adopted in the turbo codes in WIMAX [32], DVB-RCS2 [33], and the enhanced turbo codes originally proposed for 5G NR [35]. The drawback is that it requires an additional step to determine the initial state compared to the first and second methods.

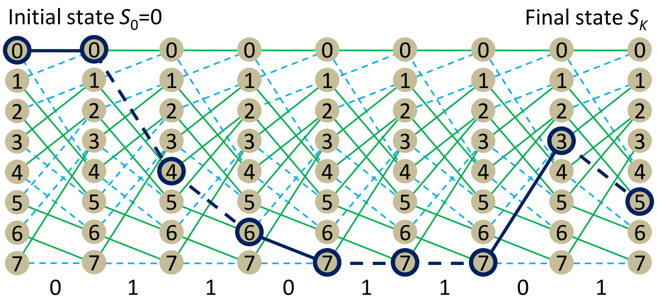


FIGURE 8. Tail-biting encoding step 1: initial state $S_0 = 0$ and the final state $S_K = 5$ [35].

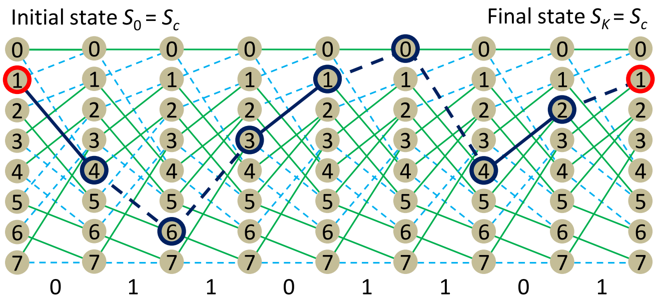


FIGURE 9. Tail-biting encoding step 2: initial state $S_0 = 1$ and the final state $S_K = 1$ [35].

As an example, we show the two-step tail-biting encoding for a K bit message $\boldsymbol{u}$ with component encoder with generator polynomial $[1, 15/13]_8$ in Figs. 8-9. Assume that $K = 8$ and $\boldsymbol{u} = [0, 1, 1, 0, 1, 1, 0, 1]$. In the first step, the message bits are encoded from the initial state $S_0 = 0$ to the final state $S_K = 5$. Then, by using Table 10, we find the value of the circular state S_C. In the second step, the encoder starts from the initial state $S_0 = S_C = 1$ and finishes in the final state $S_K = S_C = 1$. The contents of the circulation state table depend on the code memory and the recursion polynomial. The computation principle is described in [44], [45], and [46,

TABLE 10. Table for circular states S_C of the RSC code $[1, 15/13]_8$ as a function of K mod 7 and the final state obtained in the first encoding step [35].

S_K \ K mod 7	1	2	3	4	5	6
0	0	0	0	0	0	0
1	6	4	3	2	5	7
2	3	5	4	6	7	1
3	5	1	7	4	2	6
4	7	2	1	5	6	3
5	1	6	2	7	3	4
6	4	7	5	3	1	2
7	2	3	6	1	4	5

Ch. 5.5.1]. Note also that for the example in Figs. 8-9, the circulation exists if and only if K is not a multiple of 7.

3) DECODING OF CONVOLUTIONAL CODES

The decoding of convolutional codes can be realized by two popular algorithms: the *Viterbi algorithm (VA)* [47] and the *Bahl–Cocke–Jelinek–Raviv (BCJR) algorithm* [48]. The VA was recognized as an ML decoder in [49]. To enable soft-output, [50] introduced soft-output VA (SOVA). We note that tail-biting convolutional codes (TBCC) with wrap-around VA (WAVA) [51] have been adopted in 4G LTE for control channel [31]. Notably, for $(n, k) = (128, 64)$, the TBCC with $m = 14$ under WAVA has the best performance among all other short codes [25]. When using CRC-aided list Viterbi decoding [52], TBCCs also remain competitive [53].

The BCJR decoder is the bit-wise maximum a posteriori (MAP) decoder [48], which has been widely used as the decoding algorithm for the convolutional component codes of turbo codes in most communication standards. It was shown in [54] that turbo codes with iterative BCJR decoding have a gain of 0.7 dB of those with iterative SOVA. For the interest of turbo codes, we only describe the BCJR algorithm here. Particularly, we consider the *log domain* implementation of the BCJR algorithm (Log-MAP) [54] as it allows efficient implementation.

Log-MAP decoding: Recall that $\boldsymbol{u}_t$ and $\boldsymbol{c}_t$ are the information and codeword sequences at time $t = 1, \dots, K$. Let $\boldsymbol{x}_t \in \{-1, 1\}^n$ be the transmitted BPSK modulated codeword at time t and $\boldsymbol{y}_t$ be the corresponding received vector. The branch metric of the trellis edge departing from state s' at time $t-1$ to state s at time t is computed as

$$\gamma_t(s', s) = u_t(s', s)L_{\mathrm{A}}(u_t) + \sum_{j=1}^{n} c_t^{(j)}(s', s)L\Big(y_t^{(j)}|x_t^{(j)}\Big), \tag{12}$$

where (s', s) indicates the association to the state transition from s' to s, $L_{\mathrm{A}}(u_t)$ is the *a priori* LLR of u_t, and $L(y_t^{(j)}|x_t^{(j)})$ is the j-th channel LLR at time t. In turbo decoding, $L_{\mathrm{A}}(u_t)$ is obtained from the extrinsic information from another component decoder as shown in Section V-B2. For stand-alone convolutional decoding without *a priori* information, $L_{\mathrm{A}}(u_t) = 0$. The forward and backward metrics of the log-MAP decoder, denoted by α and β, are computed in (13) and (14), respectively, where

$$\alpha_t(s) = \max{}^*_{s'}\big(\alpha_{t-1}(s') + \gamma_t(s', s)\big), \tag{13}$$

$$\beta_{t-1}(s') = \max{}^*_{s}\big(\beta_t(s) + \gamma_t(s', s)\big), \tag{14}$$

where the max star function is defined as

$$\max{}^*(x, y) \triangleq \ln(e^x + e^y) = \max(x, y) + \ln\Big(1 + e^{-|x-y|}\Big). \tag{15}$$

The soft-decision output for the LLR of u_t is

$$\begin{aligned} L(u_t) = {} & \max{}^*_{(s',s):u_t=1}\big(\alpha_{t-1}(s') + \gamma_t(s', s) + \beta_t(s)\big) \\ & - \max{}^*_{(s',s):u_t=0}\big(\alpha_{t-1}(s') + \gamma_t(s', s) + \beta_t(s)\big), \end{aligned} \tag{16}$$

Assume that the encoder is initialized and terminated to the zero state. We have the following initial conditions for the forward and backward metrics

$$\alpha_0(s) = \beta_k(s) = \begin{cases} 0, & s = 0 \\ -\infty, & s \neq 0. \end{cases} \tag{17}$$

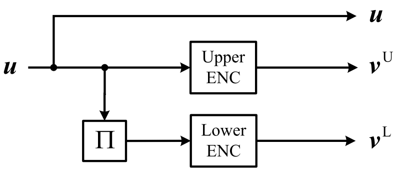


FIGURE 10. Turbo code encoder.

B. PROPERTIES OF TURBO CODES

In this section, we first introduce the encoding and decoding of turbo codes. In addition, we will also show that turbo codes can be represented by a tanner graph. Finally, we will discuss the distance properties of turbo codes.

1) ENCODING OF TURBO CODES

As shown in Fig. 10, the encoder consists of two RSC encoders and one interleaver.

A length K information sequence $\boldsymbol{u}$ is encoded by the upper convolutional encoder. The upper encoder outputs a length N_1 codeword $\boldsymbol{c}_1 = [\boldsymbol{u}, \boldsymbol{v}^{\mathrm{U}}]$, where $\boldsymbol{v}^{\mathrm{U}}$ denotes the parity bits generated by the upper encoder. Meanwhile, the information sequence $\boldsymbol{u}$ is interleaved and become $\Pi(\boldsymbol{u})$, where $\Pi(.)$ represents the interleaving function. The interleaved information sequence is encoded by the lower convolutional encoder and becomes a length N_2 codeword $\boldsymbol{c}_2 = [\Pi(\boldsymbol{u}), \boldsymbol{v}^{\mathrm{L}}]$, where $\boldsymbol{v}^{\mathrm{L}}$ represents the parity bits generated by the lower encoder. The final turbo codeword to be transmitted is $\boldsymbol{c} = [\boldsymbol{u}, \boldsymbol{v}^{\mathrm{U}}, \boldsymbol{v}^{\mathrm{L}}]$. The rate of turbo code is

$$R = \frac{K}{N_1 + N_2 - K} = \frac{1}{1/R_1 + 1/R_2 - 1}, \tag{18}$$

where $R_1 = \frac{K}{N_1}$ and $R_2 = \frac{K}{N_2}$ are the code rates of constituent convolutional codes [55].

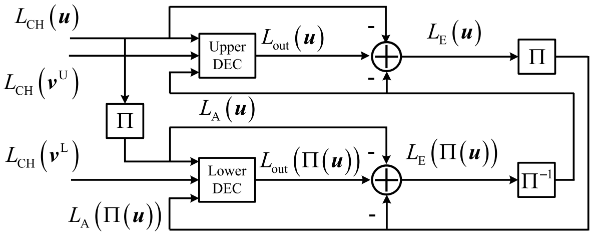

$L_{\text{CH}}(\boldsymbol{u})$
$L_{\text{CH}}(\boldsymbol{v}^{\text{U}})$
$L_{\text{CH}}(\boldsymbol{v}^{\text{L}})$
$L_{\text{A}}(\Pi(\boldsymbol{u}))$
Upper DEC
Lower DEC
$L_{\text{out}}(\boldsymbol{u})$
$L_{\text{E}}(\boldsymbol{u})$
$L_{\text{A}}(\boldsymbol{u})$
$L_{\text{out}}(\Pi(\boldsymbol{u}))$
$L_{\text{E}}(\Pi(\boldsymbol{u}))$
Π
Π^{-1}
-

FIGURE 11. Turbo code decoder.

To increase the code rate, a puncturer is required to reduce the codeword bits. Denote by $\rho \in [\frac{1}{(1/R_1+1/R_2-1)}, 1]$ the portion of surviving bits after puncturing. The code rate of a punctured turbo code is

$$R = \frac{1}{(1/R_1 + 1/R_2 - 1)\rho}. \tag{19}$$

Both interleaving and puncturing patterns affect the waterfall and error floor performance of a turbo code. The design of interleaving and puncturing patterns will be introduced in Sections V-E and V-F.

2) DECODING OF TURBO CODES

The diagram of a turbo decoder is depicted in Fig. 11. The turbo decoder consists of two constituent soft-in soft-out (SISO) decoders, an interleaver Π and a deinterleaver Π^{-1}. The decoding of turbo codes is performed iteratively between the two SISO decoders.

Let $L_{\text{CH}}(.)$, $L_{\text{A}}(.)$, and $L_{\text{E}}(.)$ represent the channel, a-priori, and extrinsic LLRs, respectively. Moreover, let $L_{\text{in}}(.)$ and $L_{\text{out}}(.)$ represent the input and output LLRs of the BCJR decoder, respectively. The SISO decoding function is denoted by $D_{\text{SISO}}(.)$. At the ℓ-th iteration, $\ell \in \{1, \ldots, \ell_{\max}\}$, the inputs to the upper SISO decoder are

$$L_{\text{in}}^{(\ell)}(\boldsymbol{u}) = L_{\text{A}}^{(\ell)}(\boldsymbol{u}) + L_{\text{CH}}(\boldsymbol{u}) \tag{20}$$

$$= \Pi^{-1}\left(L_{\text{E}}^{(\ell-1)}(\Pi(\boldsymbol{u}))\right) + L_{\text{CH}}(\boldsymbol{u}), \tag{21}$$

$$L_{\text{in}}^{(\ell)}(\boldsymbol{v}^{\text{U}}) = L_{\text{CH}}(\boldsymbol{v}^{\text{U}}). \tag{22}$$

The key idea is that the extrinsic information from the lower decoder from the previous iteration is used as the *a priori* information at the upper decoder. The outputs of the upper SISO decoder are

$$\left[L_{\text{out}}^{(\ell)}(\boldsymbol{u}), L_{\text{out}}^{(\ell)}(\boldsymbol{v}^{\text{U}})\right] = D_{\text{SISO}}^{\text{U}}\left(L_{\text{in}}^{(\ell)}(\boldsymbol{u}), L_{\text{in}}^{(\ell)}(\boldsymbol{v}^{\text{U}})\right), \tag{23}$$

$$L_{\text{E}}^{(\ell)}(\boldsymbol{u}) = L_{\text{out}}^{(\ell)}(\boldsymbol{u}) - L_{\text{in}}^{(\ell)}(\boldsymbol{u}). \tag{24}$$

Similarly, the extrinsic information from the upper SISO decoder is used as the *a priori* information at the lower SISO decoder as follows

$$L_{\text{in}}^{(\ell)}(\Pi(\boldsymbol{u})) = L_{\text{A}}^{(\ell)}(\Pi(\boldsymbol{u})) + L_{\text{CH}}(\Pi(\boldsymbol{u})) \tag{25}$$

$$= \Pi\left(L_{\text{E}}^{(\ell)}(\boldsymbol{u})\right) + L_{\text{CH}}(\Pi(\boldsymbol{u})), \tag{26}$$

$$L_{\text{in}}^{(\ell)}(\boldsymbol{v}^{\text{L}}) = L_{\text{CH}}(\boldsymbol{v}^{\text{L}}). \tag{27}$$

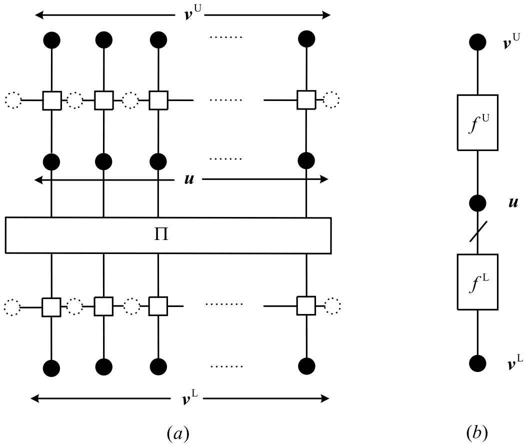


FIGURE 12. (a) Factor graph representation of a turbo code. (b) Compact graph representation of a turbo code.

The outputs of the lower SISO decoders are

$$\left[L_{\text{out}}^{(\ell)}(\Pi(\boldsymbol{u})), L_{\text{out}}^{(\ell)}(\boldsymbol{v}^{\text{L}})\right] = D_{\text{SISO}}^{\text{L}}\left(L_{\text{in}}^{(\ell)}(\Pi(\boldsymbol{u})), L_{\text{in}}^{(\ell)}(\boldsymbol{v}^{\text{L}})\right), \tag{28}$$

$$L_{\text{E}}^{(\ell)}(\Pi(\boldsymbol{u})) = L_{\text{out}}^{(\ell)}(\Pi(\boldsymbol{u})) - L_{\text{in}}^{(\ell)}(\Pi(\boldsymbol{u})). \tag{29}$$

Finally, the hard-decision estimation is performed based on the *a posteriori* LLR of $\boldsymbol{u}$, which is

$$L(\boldsymbol{u}) = L_{\text{CH}}(\boldsymbol{u}) + L_{\text{E}}(\boldsymbol{u}) + \Pi^{-1}(L_{\text{E}}(\Pi(\boldsymbol{u}))). \tag{30}$$

It was proved in [56] that the output L-values of a turbo decoder cannot grow to infinity. Hence, the optimal stopping rule is to stop iterations when the output probabilities do not change anymore [56].

3) GRAPH REPRESENTATION

Turbo codes are a class of codes on graphs. The graph representations of turbo codes can help to simplify their analysis. The factor graph [57] of a turbo code is shown in Fig. 12(a). The information and parity nodes can be regarded as variable nodes (VNs), which are represented by black circles. The trellis factor nodes are presented by white boxes. In addition, the states of the convolutional encoder are regarded as hidden VNs, represented by white circles with dash lines since they do not correspond to code bits.

To simplify the factor graph representation, [58] introduced a compact graph representation, which is shown in Fig. 12(b). The main idea is that each sequence of information and parity bits are represented by a single variable node while the trellis factor nodes are represented by a single factor node, e.g., f^{U} and f^{L}. The interleaving function is represented by a line that crosses the edge connecting node $\boldsymbol{u}$ and lower factor node f^{L}.

4) DISTANCE BOUND

It was analytically shown in [59] that there exists a length-N turbo code $\mathcal{C}$ whose minimum distance satisfies

$$d_{\min}(\mathcal{C}) \geqslant \alpha \log N, \tag{31}$$

where $\alpha > 0$ is a constant depending on the types of constituent encoders. Meanwhile, [60] proved that the minimum distance of a length-N turbo code $\mathcal{C}$ is upper bounded by the inequality

$$d_{\min}(\mathcal{C}) \leqslant O(\log N). \tag{32}$$

For a length-N turbo code $\mathcal{C}$ with two state-2 constituent convolutional codes, its distance is upper bounded by [41, Ch. 9.2]

$$d_{\min}(\mathcal{C}) \leqslant 6 \log N. \tag{33}$$

The above results indicate that designing turbo codes with a large minimum distance may be achieved by using constituent codes with small states.

C. TOOLS FOR DESIGN AND ANALYSIS

In this section, we introduce several tools to analyze both waterfall and error floor performance of turbo codes. These tools have been used for designing the component codes, interleavers, and puncturers for turbo codes. As we will see in the later sections, using a combination of different tools may be required.

1) DISTANCE EVALUATION

We introduce three main approaches that are commonly used to evaluate the distance of turbo codes in the literature.

1a) Distance Spectrum Search: Conventionally, the distance of turbo codes is based on searching the free distance of the constituent convolutional codes [61], [62] or turbo codes [63]. Particularly, [63] introduced an algorithm to determine the total or partial enumeration of a turbo codeword with input weight smaller than or equal to a given minimum distance. This method is based on the use of constrained subcodes, i.e., a subset of a code defined via constraints on the edges of its trellis. An improved method of [63] with lower computational complexity was introduced in [64]. However, these methods are more suitable for small blocklengths and small minimum distances.

1b) Error Impulse Method: The second approach is based on the error correction capability of the decoder, which is known as the error impulse method [46, Ch. 7.6.2]. It first considers transmitting an all-zero codeword, which becomes a vector $\boldsymbol{x}$ with all of its elements being -1 after BPSK mapping. Then, it introduces an error impulse to the i-th symbol of the systematic part of an all-zero codeword, i.e., $x_i = -1 + A_i$ for some amplitude A_i, where $i \in \{1, \ldots, K\}$. Let A_i^* be the maximum amplitude such that the decoded codeword is an all-zero sequence. Then, we have $d_{\min}(\mathcal{C}) = \min_{i\in\{1,\ldots,K\}}\{A_i^*\}$ if the decoder is an ML decoder. Since the turbo decoder is not an ML decoder, this method produces only an approximation of the true minimum distance. An improved method was proposed in [65], where a high amplitude error impulse is placed in a specific information bit position of $\boldsymbol{x}$ before decoding, i.e., set $P(c_i = 1) = 1$ or $\ln \frac{P(c_i=1)}{P(c_i=0)} = \infty$. In addition, AWGN noise is added to the all-zero codeword to help the decoder converge towards a low-weight codeword concurrent to the all-zero. An upper bound on the minimum distance for all codewords having a 1 in the specific data index being tested can be found. Several methods based on multiple error impulses that follow the same strategy were reported and compared in [66], which increase the accuracy of the turbo code minimum distance estimation at the cost of increased complexity.

1c) Input-Parity Weight Enumerator Function: The third approach is based on analyzing the weight enumerator functions (WEFs) of turbo codes [67, Ch. 4.3], [68, Ch 6.9]. The first step is to identify the transfer matrix associated with the state transitions of each constituent convolutional code $\boldsymbol{M}$, where the element in the i-th row and j-th column $\boldsymbol{M}[i, j]$ corresponds to the trellis branch from the i-th state to the j-th state, $i, j \in \{1, \ldots, s\}$, and s denotes the number of states. Then, we derive the average input-parity WEF (IP-WEF) of turbo codes by using the transfer matrix. Consider a rate-$1/3$ turbo code $\mathcal{C}(N, K)$ with interleaver length K and two identical rate-$1/2$ constituent convolutional codes as an example. Assume that the trellis is initialized and terminated to the all-zero state. The average IP-WEF for the upper and lower constituent codes is defined as the following polynomial

$$A(I, P) = \boldsymbol{M}^K[1, 1] = \sum_i \sum_P A_{i,p} I^i P^p, \tag{34}$$

which shows that there are $A_{i,p}$ codewords with weight-i input bits and weight-p parity bits. The coefficients of the average IP-WEF of turbo code $\mathcal{C}(N, K)$ is obtained by averaging over all possible permutation [69]

$$A_{i,p}(\mathcal{C}) = \frac{\sum_{p'} A_{i,p'} \cdot A_{i,p-p'}}{\binom{K}{i}}, \tag{35}$$

where $A_{i,p'}$ and $A_{i,p-p'}$ are obtained from the IP-WEF of convolutional component codes in (34).

After obtaining the distance profile of a turbo code $\mathcal{C}$, the BER and FER can be evaluated as

$$\mathrm{BER}(\mathcal{C}) \leqslant \sum_{i=1}^{K} \sum_{p=1}^{N-K} \frac{i}{N} A_{i,p}(\mathcal{C}) \cdot Q\left(\sqrt{2(i+p)R\frac{E_b}{N_0}}\right), \tag{36}$$

$$\mathrm{FER}(\mathcal{C}) \leqslant \sum_{i=1}^{K} \sum_{p=1}^{N-K} A_{i,p}(\mathcal{C}) \cdot Q\left(\sqrt{2(i+p)R\frac{E_b}{N_0}}\right). \tag{37}$$

2) DENSITY EVOLUTION

The asymptotic decoding threshold of turbo codes with iterative BCJR decoding can be analyzed by using density evolution (DE). DE naturally assumes infinite blocklength and ideal random interleaving. Although DE has been proposed to analyze the performance of LDPC codes a decade ago [70], the application of DE to the design and analysis of turbo codes and other turbo-like codes only started to gain some attention recently [58].

2a) Density Evolution on the Binary Erasure Channel: For the binary erasure channel (BEC), the exact DE equations for turbo codes can be derived to track the evolution of the erasure probability with the number of decoding iterations. First, the exact transfer functions between input and output erasure probabilities on both information and parity bits for a rate-k/n convolutional code under BCJR decoding are derived by following [71]

$$p_l^{\text{ext}} = f_l(p_1, \ldots, p_n), \tag{38}$$

where p_l, $l \in \{1, \ldots, n\}$, is the input erasure probability of the l-th code bit, p_l^{ext} denotes the output extrinsic erasure probability of the l-th code bit, and $f_l(.)$ is the transfer function of the BCJR decoder for the l-th code bit. The DE equations of a turbo code can then be easily obtained from the derived transfer functions of the underlying convolutional codes [58], [72]. Consider a rate-1/3 turbo code as an example. Let ϵ be the erasure probability of the BEC. The DE equations for the information bit and parity bit at the ℓ-th iteration are

$$\begin{cases} p_{\text{U,s}}^{(\ell+1)} = f_{\text{s}}^{\text{L}}\left(\epsilon \cdot f_{\text{s}}^{\text{U}}\left(\epsilon \cdot p_{\text{U,s}}^{(\ell)}, p_{\text{U,q}}^{(\ell)}\right), p_{\text{L,q}}^{(\ell)}\right), \\ p_{\text{U,q}}^{(\ell)} = p_{\text{L,q}}^{(\ell)} = \epsilon \end{cases} \tag{39}$$

where f_{s}^{U} and f_{s}^{L} are the transfer functions for the information bit at the upper and lower BCJR decoders, respectively, $p_{\text{U,s}}^{(\ell)}$ is the input erasure probability for the information bit at the upper BCJR decoder, $p_{\text{U,q}}^{(\ell)}$ and $p_{\text{L,q}}^{(\ell)}$ are the input erasure probabilities for the parity bit at the upper and lower BCJR decoders, respectively. The belief propagation (BP) decoding threshold[1] of the turbo code on the BEC is defined as

$$\epsilon^* \triangleq \sup\left\{\epsilon > 0 \middle| \lim_{\ell \to \infty} p_{\text{U,s}}^{(\ell)} = 0\right\}. \tag{40}$$

Having obtained the BEC decoding threshold, the finite blocklength performance on the BEC can be predicted by using the scaling law in [73].

2b) Density Evolution on the AWGN Channel: The DE analysis of turbo codes on the AWGN channel is similar to that on the BEC, except that the densities instead of probabilities are tracked. Let a represent the channel density experienced by the information bits, which is the same as for the parity bits. The DE equation for the information bits at the ℓ-th iteration is [68, Lemma 6.33]

$$p_{\text{U,s}}^{(\ell+1)} = f_{\text{s}}^{\text{L}}\left(a \star f_{\text{s}}^{\text{U}}\left(a \star p_{\text{U,s}}^{(\ell)}, a\right), a\right), \tag{41}$$

where $\star$ denotes the convolution operation. However, the transfer function of the BCJR decoder $f(.)$ on the AWGN channel is difficult to obtain [74]. Hence, the DE analysis of turbo codes on the AWGN channel can only be performed by Monte Carlo simulation [75]. Alternatively, given a BP threshold on the BEC ϵ^*, the computation of BP threshold on the AWGN channel σ^* can be performed by the following approximation [75]

$$\sigma^* \approx C_{\text{G}}^{-1}(1 - \epsilon^*), \tag{42}$$

where $C_{\text{G}}(\sigma)$ is the capacity of the binary-input AWGN channel with noise following $\mathcal{N}(0, \sigma^2)$.

3) EXTRINSIC INFORMATION TRANSFER (EXIT) CHART

EXIT chart is a well-known technique to visualize the exchange of extrinsic information between a pair of SISO constituent decoders [76], [77]. The characteristics of exchange information transfer are based on mutual information. EXIT chart can be used for designing the component convolutional codes for turbo codes based on the waterfall performance on the AWGN channel.

Assume BPSK signaling. Let $I_{\text{A}} \triangleq I(X; L_{\text{A}})$ denote the average *a priori* information input to the decoder, which is also the mutual information between the transmitted symbol X and the log-likelihood ratio (LLR) of the *a priori* decoder input associated to that symbol. Moreover, L_{A} can be modeled as $L_{\text{A}} = \mu_{\text{A}} x + z_{\text{A}}$, where $z_{\text{A}} \sim \mathcal{N}(0, \sigma_{\text{A}}^2)$, and $\mu_{\text{A}} = \frac{\sigma_{\text{A}}^2}{2}$ [76], which can be evaluated as

$$I_{\text{A}} = 1 - \int_{\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma_{\text{A}}}} e^{-\frac{\left(l - \frac{\sigma_{\text{A}}^2}{2}\right)^2}{2\sigma_{\text{A}}^2}} \log\left(1 + e^{l}\right) dl. \tag{43}$$

The closed-form expression for I_{A} can be found on [78, Appendix]. Let $I_{\text{E}} \triangleq I(X; L_{\text{E}})$ denote the average extrinsic information output from the decoder, which is also the mutual information between the transmitted symbol X and the LLR of the extrinsic decoder output associated to that symbol. It can be evaluated as [79]

$$I_{\text{E}} = 1 - \mathbb{E}\left[\log\left(1 + e^{-L_{\text{E}}}\right)\right]. \tag{44}$$

For a given SNR, we express I_{E} as a function of I_{A} as follows

$$I_{\text{E}}^{\text{U}}(\ell) = T^{\text{U}}\left(I_{\text{A}}^{\text{U}}(\ell)\right), \tag{45}$$

$$I_{\text{E}}^{\text{L}}(\ell) = T^{\text{L}}\left(I_{\text{A}}^{\text{L}}(\ell)\right), \tag{46}$$

where the superscripts U and L indicate the upper and lower decoders, respectively, $T(.)$ is the constituent convolutional decoder transfer function which is determined by Monte Carlo simulation. During decoding, the extrinsic output of one decoder is forwarded to the other decoder and becomes its *a priori* input, such that

$$I_{\text{A}}^{\text{L}}(\ell) = I_{\text{E}}^{\text{U}}(\ell), \tag{47}$$

$$I_{\text{A}}^{\text{U}}(\ell + 1) = I_{\text{E}}^{\text{L}}(\ell). \tag{48}$$

To analyze the iterative decoding process, we can plot two EXIT curves, i.e., I_{E}^{U} versus I_{A}^{U} and I_{A}^{L} versus I_{E}^{L}, on the same plot. The decoding is successful if $(I_{\text{E}}^{\text{U}}, I_{\text{E}}^{\text{L}})$ reach $(1, 1)$ and both curves do not intersect. The minimum required SNR for this to happen is the BP threshold on the AWGN channel. An example of the EXIT chart will be provided in Section V-F3.

[1] The decoding of turbo-like codes comprises BCJR decoding for convolutional component codes while the message exchange between BCJR component decoders follows the extrinsic message passing rule. Hence, we refer to the threshold under iterative message passing decoding with BCJR component decoding as BP threshold.

D. CONSTITUENT CODES DESIGN

The 3G [29], [30] and 4G [31] turbo codes, as well as the enhanced turbo codes [35] all have two 8-state convolutional codes with generator polynomial $[1, 15/13]_8$. In addition, the enhanced turbo codes adopt the convolutional generator polynomial $[1, 15/13, 17/13]_8$ for rates between $1/5$ and $1/3$ [35]. The design of constituent convolutional codes affects a turbo code's threshold and error floor performance. To this end, the design criteria can be divided into distance-based and threshold-based.

For the distance-based design approach [69], [80], one can adopt the methods in Section V-C1 to find the constituent convolutional codes that lead to a larger minimum distance for the turbo code. To focus on the convolutional encoder design, it is commonly assumed the use of uniform random interleaving [80]. Essentially, the distance-based design criteria place more emphasis on the error floor region of turbo codes.

For the threshold-based approach, the choice of convolutional component codes can be determined by using either DE or EXIT charts [76] to find the turbo code ensemble that has the best decoding threshold. Both DE and EXIT charts assume infinite blocklength and ideal interleaving. At rate $1/3$, the convolutional component codes of several turbo ensembles that have larger BEC thresholds than the LTE turbo ensemble were reported in [81, p117]. In addition, for rates $1/2$ and $1/3$, turbo ensembles that have AWGN thresholds within 0.3 dB from the capacity were reported in [76].

E. INTERLEAVER DESIGN

The first purpose of interleaving is the time-spreading of errors that could be produced in bursts over the transmission channel. Secondly, the interleaver design has a great impact on the error correction performance of the turbo code and especially on its minimum Hamming distance. Denote the bits input to the turbo code interleaver by $\boldsymbol{u} = [u_0, \ldots, u_{K-1}]$. The output bits of the interleaver is then denoted by $\Pi(\boldsymbol{u}) = [u_{\Pi(0)}, \ldots, u_{\Pi(K-1)}]$.

Interleavers can be divided into random interleavers and deterministic interleavers [82]. Random interleavers are usually used for analysis or simulation purposes [42], [83]. From the implementation point of view, deterministic interleavers are more favorable compared to random interleavers. In what follows, we first present several criteria for interleaver design. Then, we introduce three popular deterministic interleaver classes. For these interleavers, only a few parameters rather than the whole interleaver indices need to be stored.

1) MINIMUM SPREAD

When designing an interleaver, an important parameter we need to consider is the minimum spread. It is defined as [84], [85]

$$S_{\min} \triangleq \min_{i_1 \neq i_2 \in \{1,\ldots,K\}} |i_1 - i_2|_K + |\Pi(i_1) - \Pi(i_2)|_K, \quad (49)$$

where $|a-b|_K \triangleq \min\{|a-b|, K-|a-b|\}$. The minimum spread measures the minimum cumulated spatial distance between two bits before and after interleaving. It was shown in [86] that $S_{\min} \leqslant \lfloor \sqrt{2K} \rfloor$. A larger $S_{\min}$ can yield a larger minimum Hamming distance for turbo codes.

2) PUNCTURE-CONSTRAINED DESIGN

In LTE, puncturing and interleaver are separately designed such that the interleaver does not change with code rates [11]. However, when puncturing information bits, the effective puncturing pattern for the interleaved information sequence may be catastrophic or nearly catastrophic [87]. An effective way to combat this issue is to constrain the interleaver such that the detrimental arrangements of information bits in the lower code are not possible. There are two types of puncturing constraints on interleavers: fully-constrained and partially-constrained [87].

2a) Fully Puncture-Constrained Interleaver: When the interleaver is fully constrained under the information bits puncturing pattern, the effective puncturing pattern for the interleaved information sequence is the same as the actual puncturing pattern used for the uniterleaved information sequence. In other words, for an information puncturing mask of length M, all bits at offset $k \in \{1, \ldots, M\}$ in the mask, go to positions that are also at offset k. In effect, the information bits are divided into M sets and are forced to be interleaved only within their own sets. This can avoid bad information puncturing patterns in the lower component codeword. However, this can increase the difficulty in searching for the interleaver that achieves a given spread.

2a) Partially Puncture-Constrained Interleaver: Partially puncture-constrained interleavers include fully puncture-constrained interleavers as special cases. The punctured (unpunctured) indices at the upper encoder are interleaved only to the punctured (unpunctured) indices at the lower encoder. In effect, the information bits are divided into two sets (rather than M) and interleaved within these sets. Partially puncture-constrained interleavers typically have less difficulty in achieving a given spread and distance than fully puncture-constrained interleavers. However, for rate-compatible designs, fully puncture-constrained interleavers are preferred [88].

3) CORRELATION GIRTH

In the turbo decoding process, there are two types of information exchanges contributing to the error correcting process. The first one is the extrinsic information exchange between the two component decoders via the interleaver. The other one is the local information exchange between neighboring symbols within each component decoder due to the convolutional nature of the component codes. These two types of exchanges create some dependency loops between sets of symbols, which can be represented by cycles in a correlation graph [89], [90], in the same way as in the Tanner graph for LDPC codes [91].

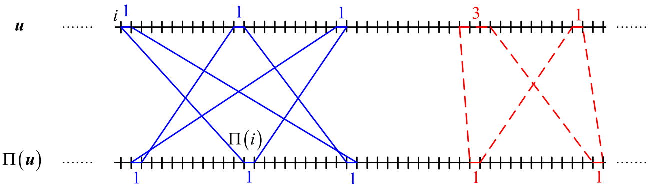


FIGURE 13. A correlation graph with two correlation cycles with length-6.

To see this, notice that the output information bit at position i from the upper decoder depends on the received bit at the same position. Moreover, it is also affected by the received bits at positions in the vicinity of i. In addition, it also depends on the *a priori* information provided by the lower decoder from position $\Pi(i)$. Similar observation can also be made for the output information bit at position $\Pi(i)$ from the lower decoder. As a result, a correlation graph can be established to design turbo code interleavers. An example of a correlation graph with two length-6 cycles is shown in Fig. 13.

Based on the correlation graph, it is natural to think about designing the interleaver by improving the minimum girth in the graph and reducing the number of minimum girths [89]. Note also that in Fig. 13, for the cycle in the dashed red line, each symbol benefits from other symbols coming from three other trellis sections. In contrast, for the cycle in the blue solid line, each symbol benefits from symbols coming from five single distant trellis sections. As pointed out by [90], the first case has a higher code diversity such that a larger number of different and distant trellis sections participate in the cycle. Hence, [90] introduced a criterion of maximizing the number of non-contiguous trellis sections participating in short correlation cycles. In addition, the number of symbols that participate in different short correlation cycles should be minimized [90].

4) PROTOGRAPH-BASED DESIGN

The concept of protograph [92] was applied to interleaver design under periodic information puncturing patterns [89]. The protograph defines a set of inter-period permutations. Specifically, given an information puncturing pattern with period M, the protograph is represented by M different sub-interleavers $\Pi_0, \ldots, \Pi_{M-1}$, where the i-th sub-interleaver Π_i, $i \in \{0, \ldots, M-1\}$, ensures that the symbols at a position within a puncturing period in $\boldsymbol{u}$ are interleaved (or connected) to a specified position within a puncturing period in $\Pi(\boldsymbol{u})$. This, combined with other design constraints, can also significantly limit the search space for the interleaver design. Fig. 14 shows an example of a protograph for puncturing period $M = 4$. The connection design will be discussed in Section V-E7.

5) QUADRATIC PERMUTATION POLYNOMIAL (QPP) INTERLEAVER

One of the popular classes of deterministic interleavers is based on QPP over integer rings [93]. The interleaving function satisfies the following quadratic form

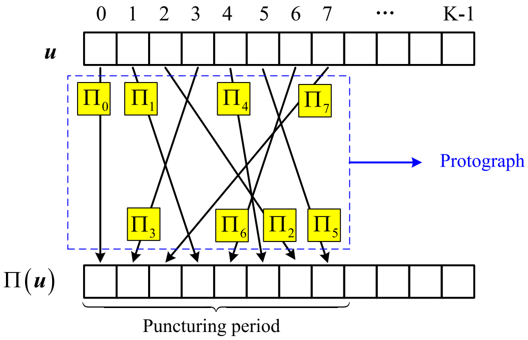


FIGURE 14. A protograph for puncturing period $M = 8$.

$$\Pi(i) = \left(f_1 \cdot i + f_2 \cdot i^2\right) \bmod K, \tag{50}$$

where f_1 and f_2 are the interleaver coefficients. These polynomial coefficients are selected based on the maximization of the minimum Hamming distance of a subset of low-weight input sequences with weights of the form $2n$, for small $n > 0$.

QPP interleavers have been shown to have large minimum Hamming distances [94] and spread [95]. In addition, [95] established many useful properties regarding the spread to reduce the search space in searching for QPP interleavers with the largest available spread. Later, [96] provides new searching methods for finding QPP interleavers for a target spread with reduced complexity. More importantly, QPP interleavers can be designed to have contention-free property, allowing a high degree of freedom for parallel processing [97]. Due to these advantages, QPP interleaver has been adopted in the LTE standard turbo coding whose interleaver coefficients f_1 and f_2 can be found in [31, Tables 5.1.3-3].

6) DITHERED RELATIVE PRIME (DRP) INTERLEAVER

DRP interleavers [98] were among the best-known interleavers for turbo codes according to [95]. A DRP interleaver consists of the following interleaving stages [98]

$$\Pi_1(i) = F\lfloor i/F \rfloor + f_{(i \bmod F)}, \tag{51}$$
$$\Pi_2(i) = (s + P \cdot i) \mod K, \tag{52}$$
$$\Pi_3(i) = G\lfloor i/G \rfloor + g_{(i \bmod G)}, \tag{53}$$
$$\Pi(i) = \Pi_1(\Pi_2(\Pi_3(i))), \tag{54}$$

where $\boldsymbol{f}$ and $\boldsymbol{g}$ are the read and write dither vectors with lengths F and G, respectively, P is the regular permutation period, and s is a constant shift. Note that K must be a multiple of both F and G. Let E be the least common multiple of F and G. Then, the DRP interleaving function can be simplified [98]

$$U(i) = (\Pi(i) - \Pi(i-1)) \bmod K, \tag{55}$$
$$\Pi(i) = (\Pi(i-1) + U(i \bmod E)) \bmod K, \tag{56}$$

where $\boldsymbol{U} = [U_{(0)}, \ldots, U_{(E-1)}]$ is the index increments vector that needs to be stored and $\Pi(0)$ is arbitrary and can be set to 0. DRP interleavers also allow a parallelism of degree C equal to any multiple of the dither vector length, provided that C remains a factor of K [99].

The design criteria for DRP interleaver parameters are similar to that for a QPP interleaver. First, a regular interleaver with high scattering properties is identified. Then, the dither vectors of the DRP interleaver are selected in order to maximize the minimum Hamming distance of a subset of low-weight input sequences [100]. To ensure that the interleaver also performs well under puncturing, [88] and [101] designed partially and fully puncture-constrained DRP interleavers, respectively, based on distance search and error impulse method as introduced in Section V-C1. Notably, turbo codes with 4-state convolutional constituent codes and rate larger than $2/3$ can have decoding thresholds within 0.1 dB to the BI-AWGN capacity under fully puncture-constrained DRP interleavers [101].

7) ALMOST REGULAR PERMUTATION (ARP) INTERLEAVER

ARP interleavers were originally proposed in [102]. It is based on a regular permutation of period P and a vector of shifts $\boldsymbol{S} = [S_{(0)}, \ldots, S_{(Q-1)}]$, where Q represents the number of shifts. The interleaving function is given by

$$\Pi(i) = \left(P \cdot i + S_{(i \bmod Q)}\right) \bmod K. \tag{57}$$

Note that K must be a multiple of Q. Moreover, P and K must be mutually prime numbers. A major advantage of ARP interleaving is that it naturally offers a parallelism degree of Q in the decoding process [34, Ch. 3.3.2]. It is worth noting that ARP interleavers were adopted in the IEEE 802.16 WiMAX [32] and DVB-RCS2 [33] standards.

It is worth pointing out that the ARP interleavers adopted in enhanced turbo codes [35] are protograph based [89], where the values of Q, P, and $[S_{(0)}, \ldots, S_{(Q-1)}]$ for various K and R can be found in Tables VI.1.2, 7.1.2, 7.1.3, 7.2.3, and 7.2.4 in [35]. Significant gains in terms of better waterfall and error floor performance over LTE interleaving and puncturing were reported in [103]. To design the protograph-based ARP interleavers, the first step is to classify the positions from the least reliable information bit to the most reliable bit based on their distance spectrum of the punctured RSC code [89]. The protograph involves connecting the least reliable bit position of one RSC code to the most reliable bit position of the other one, the second least reliable bit position of one RSC code to the second most reliable bit position of the other one, and so on [89]. This spreads the error correction capability of the turbo code over the whole information block.

To allow the error floor of turbo codes to approach the union bound based on their Hamming distance spectrum, [90] incorporated additional criteria in terms of minimizing the multiplicity of small correlation cycles, the number of symbols that participate in different short correlation cycles, and maximizing the number of non-contiguous trellis sections participating in short correlation cycles.

8) EQUIVALENCE BETWEEN ARP, QPP, AND DRP INTERLEAVERS

Interestingly, it was analytically shown in [104] that DRP and QPP interleavers can be expressed in terms of ARP interleavers. Later, the equivalence between cubic permutation polynomial and APR interleavers has been shown [105]. These results imply that ARP interleavers are capable of achieving at least the same interleaving properties and the same distance spectra as QPP or DRP interleavers. Consequently, a unified design on the interleaverd for turbo codes becomes possible.

F. PUNCTURER DESIGN

Puncturing can be classified into random puncturing and deterministic puncturing. Random puncturing is usually used for decoding threshold analysis. In this section, we focus on deterministic puncturing. Moreover, we consider a periodic puncturing pattern. Note that the performance difference between the periodic puncturing and the non-periodic puncturing becomes negligible when the puncturing period is large [106].

Consider a rate-1/3 turbo code. The puncturing pattern or puncturing mask can be represented by

$$\boldsymbol{p}_i = [p_{i,0}, p_{i,1}, \ldots, p_{i,M_i-1}], \tag{58}$$

where $i \in \{s, U_p, L_p\}$ indicates that the puncturing pattern is for information, upper, and lower parity sequences, respectively, and M_i is the puncturing period such that the puncturing is performed for every M_i bit. Note that M_i should be a divisor of the information length K. Moreover, each element in $\boldsymbol{p}_i$ only takes either 0 or 1, meaning that the corresponding code bit is not transmitted or transmitted, respectively. After puncturing, the code rate becomes

$$R = \frac{1}{\frac{\sum_{j=0}^{M_s-1} p_{s,j}}{M_s} + \frac{\sum_{j=0}^{M_{U_p}-1} p_{U_p,j}}{M_{U_p}} + \frac{\sum_{j=0}^{M_{L_p}-1} p_{L_p,j}}{M_{L_p}}}. \tag{59}$$

Conventionally, the puncturing patterns were designed based on the distance properties of turbo codes, see Section V-E1, [88, Sec. 3], and the references therein. Apart from the distance-based designs, we also present other puncturing design criteria in the following.

1) PUNCTURER DESIGN BASED ON DENSITY EVOLUTION

The design of puncturing patterns can be carried out by using DE on the BEC [72]. The puncturing pattern is determined by optimizing the BEC decoding threshold. The derivation of DE equations mostly follows Section V-C2, except that the fixed puncturing pattern is incorporated into the transfer functions of the constituent convolutional codes under BCJR decoding. To illustrate the key idea, consider a rate-1/2 convolutional code with parity puncturing pattern $\boldsymbol{p} = [p_1, \ldots, p_M]$ with period M. Then, the transfer function of the BCJR decoder for the l-th code bit, $l \in \{1, 2\}$ is given by [72]

$$f_{\boldsymbol{p},l}(x_1, x_2) = \frac{1}{M-1} \sum_{j=0}^{M-1} f_{p_j,l}(x_1, x_2), \tag{60}$$

where $f_{p_j,l}(x_1, x_2)$ is the transfer function for the l-th code bit under the condition of whether the corresponding parity bit is punctured $(p_j = 0)$ or not $(p_j = 1)$. It is worth noting that the DE based approach can detect a catastrophic puncturing case where infinite error events occur [72].

2) PUNCTURER DESIGN BASED ON EXIT CHARTS

Puncturing patterns can also be designed by the EXIT chart analysis [87], [89]. When only parity bits are punctured, the convergence behavior of the resultant turbo codes can be predicted by the EXIT charts [76]. However, when information bits are punctured, the distribution of the extrinsic information related to punctured information positions is different from that related to the unpunctured counterpart [87]. Rather than relying on the Gaussian approximation of the *a priori* information I_{A} in the conventional EXIT chart analysis, the *a priori* mutual information within the actual turbo decoding is measured by Monte Carlo simulations [89]. The best puncturing pattern in terms of convergence performance is the one providing the closest crossing point $(I_{\mathrm{E}}^{\mathrm{U}}, I_{\mathrm{E}}^{\mathrm{L}})$ to $(1, 1)$.

3) PUNCTURER DESIGN BASED ON DISTANCE AND THRESHOLD CRITERIA

For some configurations of block sizes and code rates, the current LTE puncturing does not lead to a good error floor and decoding threshold due to the undesirable interactions between the puncturer and the interleaver [107]. To address this issue, a joint design of puncturing and interleaving is necessary [89]. More importantly, the design of puncturing patterns needs to take into account both distance and threshold criteria.

To ensure achieving a good minimum distance and decoding threshold at the same time, a set of candidate puncturing patterns are selected according to the distance criteria as in Section V-C1. In [89], the FAST algorithm [61] was employed to evaluate the truncated distance spectrum of each constituent convolutional code punctured by a periodic pattern of period M. Moreover, the FAST algorithm needs to run M times, where each time starts from position $j \in \{0, \ldots, M-1\}$ in the puncturing pattern. The resulting M distance spectra are accumulated to obtain the Hamming distance spectrum of the punctured convolutional code. The set of puncturing pattern candidates is selected by finding the largest distance values in the first spectrum terms and the minimal number of codewords at these distances. Then, the EXIT analysis following Section V-F2 is carried out for these puncturing patterns. For example, Fig. 15 shows the EXIT curves for the LTE turbo code with uniform interleaving under data puncturing constraint (DPC uniform) [89], where puncturing patterns with different data puncturing ratios D_p are applied. From Fig. 15, we see that the puncturing pattern with $D_p = 2/16$ has the best convergence performance. Finally, the puncturing pattern that has the best trade-off between performance in the waterfall and error floor regions is selected. Under the constraints introduced by the selected puncturing pattern (see Section V-E2), the interleaver that achieves the best turbo code Hamming distance spectrum is selected. The puncturing patterns for the enhanced turbo codes proposed for 5G can be found in [35, Tables 6.1.1, 7.1.1, 7.2.1, and 7.2.2].

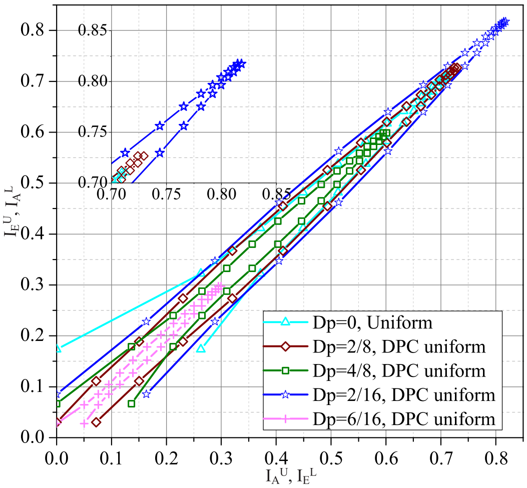


FIGURE 15. Extrinsic information exchange between two constituent codes of the LTE turbo code at $E_b/N_0 = 1.6$ dB. Five puncturing patterns with different data puncturing ratios are applied to reach code rate $2/3$ [89].

G. TURBO CODE DECODING ALGORITHMS

In Section V-B2, we know that the turbo decoder is an iterative BCJR decoder. However, even though the BCJR decoder is recursive, it poses implementation challenges because of the necessity of non-linear functions, and a large number of addition and multiplications [54]. In this section, we introduce several variants of turbo decoding algorithms. Note that we focus on the AWGN channel in this section. Recently, machine learning based turbo decoders have also been proposed in [108], [109], [110] to improve the robustness and adaptivity of non-AWGN channels.

1) THE LOG-MAP DECODER

As introduced in Section V-A3, the Log-MAP decoder is the log domain implementation of the original BCJR decoder. By converting into the logarithmic domain, the Log-MAP algorithm replaces the multiplications and additions from the original BCJR decoder by additions and the max star functions defined in (15).

Notice that the term in (15), i.e., $\ln(1 + e^{-|x-y|})$, is called the correction term. Since direct computation is costly to implement in hardware, it is often desirable to approximate the correction term. Since the value of $\ln(1 + e^{-|x-y|})$ is always in the range $[0, 0.69]$ [111], it can be easily

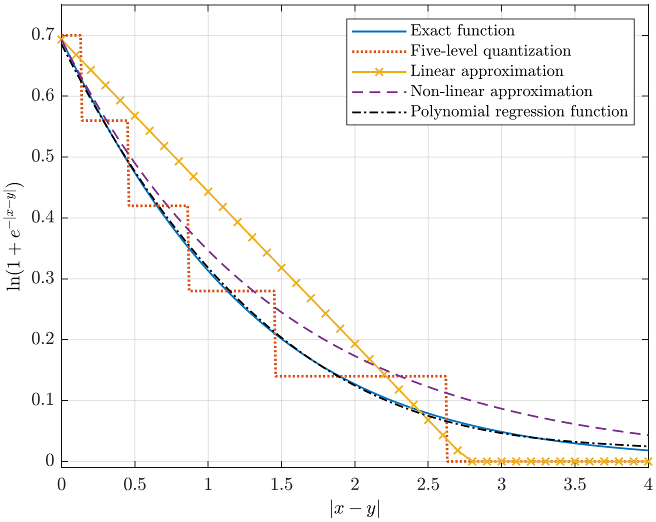


FIGURE 16. Plot of $\ln(1+e^{-|x-y|})$, showing the exact values and four approximated values.

pre-calculated and stored in a lookup table for efficient implementation [112]. Specifically, a table size of eight is usually sufficient to keep the same performance as the original MAP decoder [54]. In [113], a linear approximation for the correction term was suggested as follows

$$\ln\left(1+e^{-|x-y|}\right) \approx \max\left(\log 2 - \frac{|x-y|}{4}, 0\right). \tag{61}$$

Inspired by the linear approximation, [114] introduced the following non-linear approximation with higher accuracy

$$\ln\left(1+e^{-|x-y|}\right) \approx \frac{\log 2}{2^{|x-y|}}. \tag{62}$$

In [115], a more accurate approximation based on polynomial regression functions was proposed as follows

$$\ln\left(1+e^{-|x-y|}\right) \approx \begin{cases} -0.0098(x-y)^3 + 0.1164(x-y)^2 \\ -0.474(x-y) + 0.6855, & x_y \leqslant 5 \\ 0, & \text{otherwise} \end{cases}. \tag{63}$$

Fig. 16 shows the comparison between the exact correction term and the approximated correction terms computed by the above four methods.

2) THE MAX-LOG-MAP DECODER

The Max-Log-MAP decoder simplifies the Log-MAP decoder by approximating the max star functions in (13)-(16) as [54]

$$\max{}^*(x, y) = \ln(e^x + e^y) \approx \max(x, y). \tag{64}$$

At the cost of some performance degradation, the Max-Log-MAP algorithm does not require the computation of the correction terms in the max star function. Moreover, it is the most currently used in hardware implementations of turbo decoders [34]. An interesting advantage of the Max-Log-MAP decoder compared to the original MAP decoding is that the estimation of signal-to-noise ratio is not required [116]. This makes the Max-Log-MAP decoder more robust to the channel estimation error than the Log-MAP decoder.

To reduce the performance loss, [117] suggested multiplying the extrinsic information, i.e., (24) and (29), by a scaling factor $s^{(\ell)} \in [0, 1]$ before being used by a SISO decoder. For easy hardware implementation, the scaling factor is fixed to 0.7 [117] or 0.75 [107] for all iterations. For the decoding of the enhanced turbo codes [103], the following scaling factor is used: $s^{(1)} = 0.6$, $s^{(\ell_{\max})} = 1$, $s^{(\ell)} = 0.7, \forall \ell \in \{2, \dots, \ell_{\max} - 1\}$. One may also design the scaling factors adaptive to SNR as in [118].

3) THE LOCAL SOVA DECODER

To improve the decoding throughput, high-radix decoding, i.e., several successive symbols are decoded at once, is often considered [119]. However, for the Max-Log-MAP algorithm, the computational complexity increases rapidly with the radix orders. Inspired by the early discovery on the equivalence between the Max-Log-MAP decoding and SOVA decoding in [120], the local SOVA decoder was introduced in [121] as a low complexity alternative to the Max-Log-MAP decoder. It relies on a new low-complexity soft-output calculation unit that applies a path-based decoding variant of the Max-Log-MAP algorithm. It was revealed that the soft output generated by the Max-Log-MAP algorithm is a special case of the local SOVA method [121]. Hence, the local SOVA decoder can achieve the same performance as that of the Max-Log-MAP decoder but with lower complexity.

The local SOVA decoder uses the same recursive metrics calculations in (13) and (14). However, the LLRs are computed using path-based local update rules known as the Hagenauer rule [50] and the Battail rule [122], [123]. Consider a radix-2^q trellis where there are 2^q branches coming in and out of a state s at a time index. For each trellis section, a path P_s for state s is defined as

$$P_s \triangleq \{M_s, u_s, L_s\} \in \mathbb{R} \times \{0, 1\}^q \times \{\mathbb{R}^+\}^q, \tag{65}$$

where M_s is the path metric, $u_s \triangleq \{u_{s,0}, \dots, u_{j,q-1}\}$ is the q hard decisions labeling this path, and $L_s \triangleq \{L_{s,0}, \dots, L_{s,q-1}\}$ are the reliability values associated with the hard decisions. Two paths P_a and P_b can be merged into path P_c according to

$$M_c = f_0(M_a, M_b) \triangleq \max(M_a, M_b), \tag{66}$$
$$u_c(l) = f_1(u_a(l), u_b(l)), \forall l \in \{0, \dots, q-1\}, \tag{67}$$

where f_1 selects the hard decision of the winning path resulting from (66). Define $p \triangleq \arg\max(M_a, M_b)$, $p' \triangleq \arg\min(M_a, M_b)$, and $\Delta_{p,p'} \triangleq M_p - M_{p'}$. The reliability of $u_c(l)$ is updated as

$$\begin{aligned} L_c(l) &= f_2(L_a(l), L_b(l)), \forall l \in \{0, \dots, q-1\}, &(68)\\ &= \min\big(L_p(l), \Delta_{p,p'} + \mathbf{1}(u_a(l) = u_b(l)) \cdot L_{p'}(l)\big). &(69) \end{aligned}$$

where $\mathbf{1}(.)$ is the indicator function, and (69) is the result of combining both the Hagenauer rule [50] and the Battail rule [122], [123] for the cases $u_a(l) \neq u_b(l)$ and $u_a(l) = u_b(l)$, respectively.

Motivated by the feature of convolutional codes that all trellis paths merge to the maximum likelihood path after some trellis steps, [124] further proposed a low-complexity local SOVA decoder. Specifically, the reliability values L for all previous trellis steps during the computation of the recursion metrics for each new radix-2^q trellis segment are updated instead of being recursively computed. To see this, let $P_1^{\mathrm{f}}(i), \ldots, P_{2q}^{\mathrm{f}}(i)$ be the paths at trellis position i to be merged to compute $P_{\mathrm{out}}^{\mathrm{f}}(i+q)$ at trellis position $i+q$. The merge operation computes

$$M_{\mathrm{out}}^{\mathrm{f}} = f_0(M_1, \ldots, M_{2q}) = \max\left(M_1^{\mathrm{f}}, \ldots, M_{2q}^{\mathrm{f}}\right), \quad (70)$$

$$u_{\mathrm{out}}^{\mathrm{f}}(l) = f_1\left(u_1^{\mathrm{f}}(l), \ldots, u_{2q}^{\mathrm{f}}(l)\right), \forall l \in \{1, \ldots, i+q\}, \quad (71)$$

where f_1 is defined in (67) below. The reliability values are updated via

$$L_{\mathrm{out}}^{\mathrm{f}}(l) = f_2\left(L_1^{\mathrm{f}}(l), \ldots, L_{2q}^{\mathrm{f}}(l)\right), \forall l \in \{1, \ldots, i+q\}, \quad (72)$$

where f_2 is defined in (68). The merge operations of backward recursions are performed similarly. Compared to the original local SOVA algorithm, a complexity reduction of the add-compare-select units in the order of 50% can be achieved at the price of 0.2 dB performance degradation [124].

4) CRC-AIDED TURBO DECODING

In the LTE standard [31], a CRC code is used as the outer error detection on top of a turbo code. Hence, it is natural to leverage CRC to improve the performance of turbo decoding [125], [126], [127], [128].

4a) Flip and Check Decoding: The most intuitive method is the Flip and Check (FC) algorithm proposed in [127]. The turbo decoder first iterates $\ell_{\min}$ times without using CRC verification. At iteration $\ell_{\min} + 1$ and when the CRC check fails, the reliability of the k-th information bit, $k \in \{1, \ldots, K\}$, is characterized according to the following extrinsic-information-based metric

$$\Delta_{\mathrm{E}}^{(\ell)}(u_k) \triangleq \left| L_{\mathrm{E}}^{(\ell)}(u_k) + \Pi^{-1}\left(L_{\mathrm{E}}^{(\ell)}(u_{\Pi(k)})\right)\right|, \quad (73)$$

where we adopt the same notations of the extrinsic information as in Section V-B2. A set of test patterns $\boldsymbol{\tau}_j$, $j \in \{1, \ldots, 2^{q_{\mathrm{FC}}} - 1\}$, are generated by identifying the least q_{FC} reliable bits based on the smallest values of $\Delta_{\mathrm{E}}^{(\ell)}(u_k)$. We denote by $\hat{\boldsymbol{u}}^{(\ell)}$ the least q_{FC} reliable bits after ℓ turbo decoding iterations. Then, the candidate CRC codeword is generated by flipping the decoded values of those bits

$$\tilde{\boldsymbol{u}}_j = \hat{\boldsymbol{u}}^{(\ell)} \oplus \boldsymbol{\tau}_j, \quad (74)$$

where $\oplus$ denotes the modulo 2 sum. The CRC check is performed on all candidates $\tilde{\boldsymbol{u}}_j$ for $j \in \{1, \ldots, 2^{q_{\mathrm{FC}}} - 1\}$. The one that passes the CRC check is kept. Finally, the turbo decoding process and the FC principle are repeated, until the CRC is verified or $\ell = \ell_{\max}$. To reduce the undetected error rate due to wrongly codewords satisfying the CRC, [128] proposed a new reliability metric based on choosing the smallest Euclidean distance between the CRC codeword after flipping and the received information.

4b) Blind Candidate Decoding: Recently, a new method called Blind Candidate Decoding (BCD) is proposed in [128], which can combine with the FC principle above. The key idea is to generate a set of candidates within an Euclidean sphere around the original received information LLR $L^{(0)}(\boldsymbol{y})$. Let $G_\gamma(k)$ be a length K vector filled with equally spaced values centred at 0

$$G_\gamma(k) = \gamma \frac{2k}{K-1} - 1, k \in \{1, \ldots, K\}, \quad (75)$$

where γ is a parameter that controls the norm $d_\gamma = \gamma^2 E_K$ of vector G_γ, and E_K is a constant given by [128]

$$E_K = \frac{K}{3}\left(\frac{2}{K-1} + 1\right). \quad (76)$$

Then, generate n_c number of candidates as

$$L^{(\ell)}(\tilde{y}_{k,i}) = L^{(0)}(y_{k,i}) + G_\gamma(\Pi_i(k)), \ k = 1, \ldots, K, \quad i = 1, \ldots, n_c, \quad (77)$$

where $\Pi_i(.)$ is a permutation function for indices set $\{1, \ldots, K\}$ randomly generated for each candidate $i \in \{1, \ldots, n_c\}$, and n_c denotes the number of candidates. Each candidate LLR vector $L^{(\ell)}(\tilde{\boldsymbol{y}}_i)$ is input to the turbo decoder with CRC check. The optimal distance d_γ for a given code configuration and SNR is determined by Monte-Carlo simulation [128].

H. HIGH-THROUGHPUT TURBO DECODER HARDWARE ARCHITECTURES

The BCJR algorithm is serial in nature due to the recursive calculation of the state metrics. As a result, it requires a relatively large amount of memory for storing the state metrics, and its throughput can be very limited. To this end, several works have proposed new hardware architectures for high-throughput turbo decoders beyond 100 Gb/s.

1) PARALLEL MAP (PMAP) ARCHITECTURE

Turbo decoders with a PMAP architecture divide a length-K trellis into P sub-trellis of length K_P. Each sub-trellis is decoded independently by a sub-decoder core in parallel [129]. Additionally, each sub-decoder core splits the sub-blocks further into smaller blocks with size W called windows [130], to enable a parallel processing of the forward and backward recursions. The PMAP architecture with $P = 4$ is illustrated in Fig. 17.

Let n_{HI} be the number of half iterations performed by the decoding process. By neglecting the I/O latency and the latency due to metric initialization, the latency of the turbo decoder with PMAP architecture can be estimated in terms of clock cycles as $(K_P + W)n_{\mathrm{HI}}$ and the throughput is $\frac{Kf}{(K_P+W)n_{\mathrm{HI}}}$ [131], where f is the maximum operating

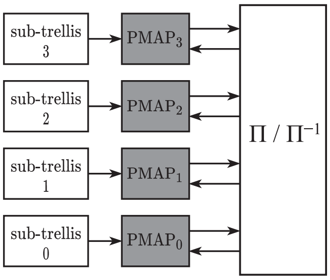


FIGURE 17. The PMAP architecture with $P = 4$ [131].

frequency. With a fixed frame size K, the throughput increases with the number of parallel sub-trellises P. The asymptotic throughput of the PMAP architecture is $\lim_{k\to\infty} \frac{1}{(K_P/K+W/K)n_{\mathrm{HI}}} = \frac{Pf}{n_{\mathrm{HI}}}$. Implementations with current silicon technologies achieve a throughput in the order of single-digit Gb/s [132], [133]. Note that the maximum degree of parallelism is limited since the decoding of small sub-blocks leads to an error correction performance loss [134]. In addition, the amount of hardware resources increases by a factor P since each sub-trellis is processed by an independent SISO decoder [131].

2) PIPELINED MAP (XMAP) ARCHITECTURE

Given P sub-trellises of length K_P, the XMAP architecture processes a single sub-trellis at a time [135], [136]. Sub-trellises are time-multiplexed into a pipeline consisting of a chain of computation units (branch metric unit (BMU), add-compare-select unit (ACSU), and soft-output unit (SOU)) connected through pipeline registers. As a result, for each clock cycle, the decoder can produce K_P soft-output values. Note that the XMAP architecture only differs from the PMAP architecture in the way each individual sub-trellis is processed.

By neglecting the I/O latency and the latency due to metric initialization, the latency of the turbo decoder with XMAP architecture in clock cycles is $(K_P + P - 1)n_{\mathrm{HI}}$ and the throughput is $\frac{Kf}{(K_P+P-1)n_{\mathrm{HI}}}$ [131]. A throughput of over 1 Gb/s has been demonstrated in [136], [137]. Since the XMAP core consists of a chain of computation units set up in a pipeline fashion, its complexity increases linearly with the sub-trellis length K_P. Similarly to PMAP, the maximum degree of parallelism of XMAP is also limited due to the decoding of small sub-blocks leading to an error correction performance loss [134].

3) FULLY PARALLEL MAP (FPMAP) ARCHITECTURE

The FPMAP architecture [119] can be seen as an extreme case of the PMAP architecture with $P = K$ such that the size of the sub-trellises is reduced to $K_P = 1$. It uses a shuffled decoding scheme [138]. It employs $2K$ processing elements (PE). Each PE computes the branch metrics, the forward and backward state metrics, and the extrinsic information for one trellis step in one clock cycle. The calculated state metrics at the border are then exchanged with neighboring PEs. Meanwhile, the calculated extrinsic information is fed to the interleaved/deinterleaved PEs. Consequently, a complete turbo code iteration is processed in parallel in each clock cycle. For the LTE turbo codes, the use of the odd-even QPP interleaver helps split the PEs into two groups, where each group consists of $K/2$ PEs of the component SISO decoder and can be processed independently. The FPMAP architecture is illustrated in Fig. 18.

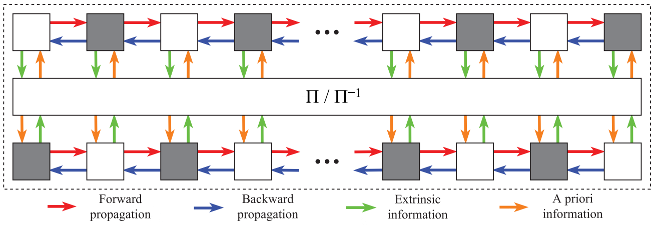


FIGURE 18. The FPMAP architecture [131].

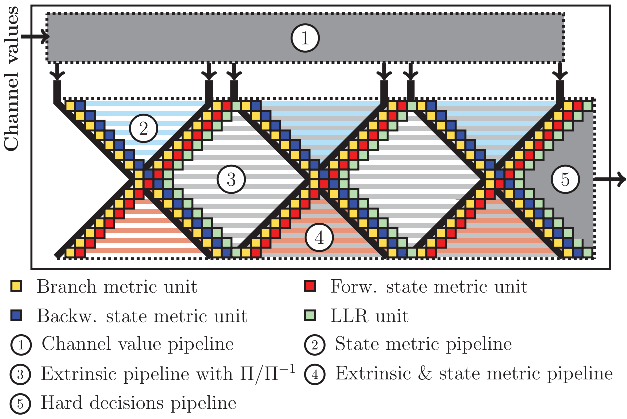

Channel values
Output bits
1
2
3
4
5
🟨 Branch metric unit
🟥 Forw. state metric unit
🟦 Backw. state metric unit
🟩 LLR unit
1 Channel value pipeline
2 State metric pipeline
3 Extrinsic pipeline with Π/Π^{-1}
4 Extrinsic & state metric pipeline
5 Hard decisions pipeline

FIGURE 19. Iteration unrolled pipelined decoder architecture [139].

By neglecting the I/O latency, the decoder latency in clock cycles of the FPMAP is $2n_{\mathrm{I}}$, where n_{I} is the number of full iterations. The throughput of the FPMAP decoder can be calculated as $\frac{Kf}{2n_{\mathrm{I}}}$ [131]. Although the FPMAP can achieve a high throughput, the price to pay is a decrease in area efficiency compared to the PMAP architectures [131]. In addition, the combination of sub-trellis size of 1 and shuffle decoding degrades the error correction performance of the decoder [139].

4) ITERATION UNROLLED XMAP (UXMAP) ARCHITECTURE

Further pipelining of the decoding by unrolling the individual half iteration of the turbo decoding leads to the fully pipelined UXMAP decoder architecture [139]. This allows for the output of a complete decoded frame per clock cycle resulting in a very high throughput, which is only limited by the achievable clock frequency and frame size. The architecture is illustrated in Fig. 19. By neglecting the I/O latency and the latency due to the metric initialization, the throughput of the UXMAP decoder architecture is Kf. The latency of the decoder can be derived in clock cycles as

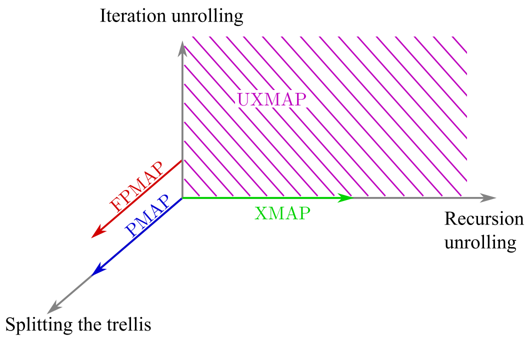


FIGURE 20. Parallelism in turbo decoder hardware architectures [140].

$n_{\mathrm{HI}}(T_{\mathrm{BMU}} + K_P + T_{\mathrm{SOU}})$, where T_{BMU} and T_{SOU} are the number of clock cycles required for processing the first BMU and the last SOU, respectively. Moreover, the ACSU takes one clock cycle to finish a trellis section, thus, it takes KP clock cycles to finish a sub-trellis.

Employing high radix schemes in the fully pipelined UXMAP architecture has a particular impact. Increasing the number of trellis sections processed in a clock cycle leads to a reduction in the number of pipeline stages for all the pipeline registers (state metrics, channel, and extrinsic information). Hence, using radix-4 instead of radix-2 yields an area saving of about 50% and halves the overall pipeline latency. However, as we increase the radix order, the area overhead in the computation units also increases. In addition, [134] investigated the rate and frame flexibility aspect of the UXMAP. A throughput of 409.6 Gb/s was reported in [140].

Finally, different methods of parallel processing are illustrated in Fig. 20.

I. TURBO CODED MODULATIONS

Unlike LDPC codes, turbo coded modulations receive less attention. Among those works, most of them focus on bit-interleaved turbo coded modulation (BITCM) schemes [141], [142]. Alternatively, one can use non-binary turbo codes [143] constructed from convolutional codes over rings [144] to map codeword symbols to modulation symbols directly.

BITCM follows the principles of BICM, where the turbo codeword is interleaved and mapped to higher-order modulations. Reference [141] introduced a greedy algorithm for designing bit interleavers to lower the error floor of BITCM on the AWGN channel. It requires an accurate list of low-weight turbo codewords as the input to the algorithm. Reference [142] exploited unequal error protection caused by the binary labeling of the equally spaced constellations in the interleaver design. By using Gaussian approximation for the L-values in QAM and the generalized transfer function of a code, the union bound of the coded bit error rate was derived and used as the metric for bit interleaver design [142]. It can be seen that these methods can also be applied to the bit interleaver design for other codes.

J. OTHER VARIANTS OF TURBO CODES

In the following two subsections, we introduce several variants of turbo codes that may be the candidate channel coding schemes for future communication systems. In this subsection, we introduce irregular turbo codes and serially concatenated convolutional codes. In the next subsection, we introduce different classes of spatially coupled turbo codes.

1) IRREGULAR TURBO CODES

Irregular turbo codes were first introduced in [145] as a generalization of the regular turbo codes [12]. It was numerically demonstrated in [146] that a rate-$1/3$ irregular turbo code with $[1, 21/37]_8$ convolutional component codes can achieve a threshold within 0.03 dB from the BI-AWGN capacity. Reference [72] adopted density evolution to design capacity-approaching irregular turbo codes and periodic puncturing patterns on the BEC.

The conventional regular turbo encoder with two identical constituent convolutional encoders can be seen that the information bit is repeated twice, interleaved, and fed to the constituent encoder. In this regard, an irregular turbo encoder consists of a non-uniform repetition, an interleaver, and an RSC component encoder. First, the information bits can be divided into d classes with $d = 2, \ldots, d_{\max}$, where $d_{\max}$ is the maximum bit-node degree. The number of bits in class d is a fraction f_d of the total number of information bits at turbo encoder input. Moreover, each information bit in class d is repeated d times. After irregular repetition, the length-K information sequence becomes a length-N sequence. As a result, the following equalities hold

$$\sum_{d=2}^{d_{\max}} f_d = 1, \sum_{d=2}^{d_{\max}} d \cdot f_d = \bar{d}, N = K \sum_{d=2}^{d_{\max}} d \cdot f_d = K\bar{d}, \quad (78)$$

where $\bar{d}$ denotes the average bit-node degree. The length-N sequence is interleaved and fed into a constituent convolutional encoder with rate R_C. The code rate of the irregular turbo code is

$$R = \frac{K}{K + \frac{N}{R_C} - N} = \frac{1}{1 + \left(\frac{1}{R_C} - 1\right)\bar{d}}. \quad (79)$$

2) SERIALLY CONCATENATED CONVOLUTIONAL CODES (SCCCS)

Serially concatenated convolutional codes (SCCCs) were introduced in [147]. Its encoder consists of an outer and inner RSC component encoder and an interleaver.

Consider a rate-$1/4$ SCCC with two rate-$1/2$ constituent encoders. A length K information sequence $\boldsymbol{u}$ is encoded by the outer encoder to produce the parity sequence $\boldsymbol{v}^{\mathrm{O}}$. Then, the sequences $\boldsymbol{u}$ and $\boldsymbol{v}^{\mathrm{O}}$ are multiplexed and interleaved become $\Pi([\boldsymbol{u}, \boldsymbol{v}^{\mathrm{O}}])$. Then, the inner encoder takes this interleaved sequence as the input and generates the parity sequence $\boldsymbol{v}^{\mathrm{I}}$. The transmitted codeword is $\boldsymbol{c} = [\boldsymbol{u}, \boldsymbol{v}^{\mathrm{O}}, \boldsymbol{v}^{\mathrm{I}}]$. In general, PCCCs have better waterfall performance than SCCCs. However, in some cases and with a careful puncturing design, SCCCs can achieve a lower error floor and comparable waterfall performance compared to PCCCs [148].

K. SPATIALLY COUPLED TURBO CODES

Recently, spatially coupled turbo codes have gained some interest since the work of [58]. It has been demonstrated both theoretically and numerically that spatial coupled turbo codes outperform their uncoupled counterpart in terms of better waterfall and error floor performance [58], [149], [150], [151], [152]. This makes them appealing to future communication systems where both close-to-capacity and lower error floor performance will be required. In this section, we first introduce several important properties of spatially coupled turbo codes. Then, we introduce several classes of spatially coupled turbo codes, including the one that has been proved to be capacity-achieving.

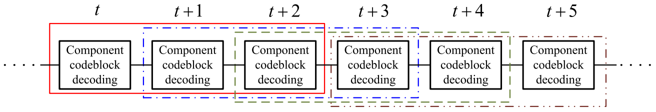
1) PROPERTIES OF SPATIALLY COUPLED TURBO CODES

Spatial coupling has been mainly applied to PCCCs [12], SCCCs [147], and braided convolutional codes (BCCs) [153]. These spatially coupled turbo codes are constructed by applying spatial coupling on the systematic component encoders, which share some structural similarity with spatially coupled product-like codes [154], [155], [156], [157]. One of the appealing features of such a construction is that the encoding of spatially coupled turbo codes can be performed in a *streaming* fashion. Note that this is different from spatially coupled LDPC codes [158], [159], where the spatial coupling is defined based on their parity-check matrices. The main idea is to construct powerful long turbo codes by using short turbo component codes. For the rest of this subsection, we let L and m represent the coupled chain length and coupling memory, respectively.

1a) Sliding Window Decoding: It is worth noting that the decoding of spatially coupled turbo codes can leverage sliding window decoding [160], Within the decoding window size $W < L$, the component codewords at time $t, \dots, t + W - 1$ are decoded by using the constituent decoder, e.g., turbo decoder. The windowed decoder outputs the decoded codeword at time t and moves to the next decoding window from time $t + 1$ to $t + W$. The windowed decoding process continues up to the point when the coupled code chain is terminated. As a result, the decoding delay of the coupled codes is [161]

$$\mathcal{L} = W \cdot K, \tag{80}$$

where K denotes the component code information length. One can see that the sliding window decoder enables a continuous streaming fashion compared to decoding the conventional block codes. An example of the sliding window decoding with window size $W = 3$ is illustrated in Fig. 21.



t
t+1
t+2
t+3
t+4
t+5
Component codeblock decoding
Component codeblock decoding
Component codeblock decoding
Component codeblock decoding
Component codeblock decoding
Component codeblock decoding

FIGURE 21. Sliding window decoding with window size $W = 3$.

1b) Threshold Saturation: The DE recursion of most spatially coupled turbo ensembles on the BEC with erasure ϵ can be described by the following [58]

$$x_t^{(\ell)} = \frac{1}{1+m}\sum_{j=0}^{m} f\left(\frac{1}{1+m}\sum_{k=0}^{m} g\left(x_{t+j-k}^{(\ell-1)}\right); \epsilon\right), \tag{81}$$

where $t \in \{1, \dots, L\}$ denotes the time instant or spatial position and (f, g) forms a scalar admissible system [162, Def. 1] defined by the recursion

$$x^{(\ell)} = f\left(g(x^{(\ell-1)}); \epsilon\right). \tag{82}$$

From Section V-C2, we know that (82) is the DE recursion of the underlying uncoupled turbo code ensemble such that the DE recursion in (39) is a special case of (82). By [162, Th. 1], the spatially coupled turbo codes defined by the recursion in (81) have the threshold saturation property [159]: the suboptimal BP decoding threshold converges to the optimal MAP decoding threshold as $L \to \infty$, $m \to \infty$ and $L \gg m$. This implies that one can design good spatially coupled turbo codes by simply increasing L and m without the need for meticulous optimization as in irregular turbo codes. Since the MAP thresholds of the coupled turbo ensembles and the corresponding uncoupled ensembles are identical [162], [163], we can design spatially coupled turbo codes by optimizing the MAP threshold of the underlying uncoupled turbo codes.

1c) MAP Decoding Threshold: On the BEC, the MAP threshold of the uncoupled turbo ensemble ϵ_{MAP} can be computed by using the area theorem [81, Lemma 4.4]

$$R = \int_{\epsilon_{\mathrm{MAP}}}^{1} R\bar{p}(\epsilon) + (1-R)\bar{q}(\epsilon)d\epsilon, \tag{83}$$

where $\bar{p}(\epsilon)$ and $\bar{q}(\epsilon)$ denote the average extrinsic erasure probability for information bits and parity bits, respectively. They are obtained from the fixed point solutions of the DE equations for information and parity bits, respectively, e.g., the solution to equation $x = f(g(x; \epsilon))$ or equivalently the value of $x^{(\ell=\infty)}$ from (82). Strictly speaking, the MAP threshold given by the area theorem is an upper bound. However, we opt to drop the term "upper bound" for simplicity as various works [58], [149], [150], [151], [152] show that the BP thresholds of the coupled turbo ensembles converge to the upper bound of their MAP thresholds.

Alternatively, we can obtain the so-called potential threshold [162, Def. 6] since it coincides with the MAP threshold [58]. To do so, we first obtain the potential

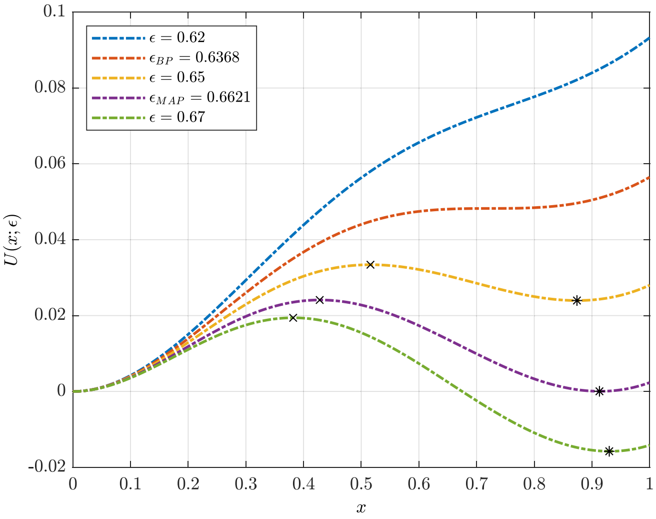


FIGURE 22. The potential function of a LTE turbo ensemble. The minimum unstable fixed point $u(\epsilon)$ and the energy gap $\min_{x\in[u(\epsilon),1]} U(x;\epsilon)$ for $\epsilon > \epsilon_{\mathrm{BP}}$ are represented by × and $*$, respectively.

function [162, Def. 2] of the scalar admissible system in (82) as

$$U(x;\epsilon) = xg(x) - G(x) - F(g(x);\epsilon), \tag{84}$$

where $F(x;\epsilon) = \int_0^x f(z;\epsilon)dz$ and $G(x) = \int_0^x g(z)dz$. Then the potential threshold ϵ_{c} is obtained as [162], [163]

$$\epsilon_{\mathrm{c}} = \sup\left\{\epsilon \in [0,1]\colon \min_{x\in[u(\epsilon),1]} U(x;\epsilon) \geqslant 0, u(\epsilon) > 0\right\}, \tag{85}$$

where

$$u(\epsilon) = \sup\{\tilde{x} \in [0,1]\colon f(g(x);\epsilon) < x, x \in (0,\tilde{x})\}, \tag{86}$$

is the minimum unstable fixed point for $\epsilon > \epsilon_{\mathrm{s}}$, and ϵ_{s} is single system threshold [162, Def. 4]

$$\epsilon_{\mathrm{s}} = \sup\{\epsilon \in [0,1]\colon U'(x;\epsilon) > 0, \forall x \in (0,1]\}, \tag{87}$$

which is also the BP threshold defined in (40). In general, the computation of the potential threshold from the potential function is simpler than that of the MAP threshold from the area theorem for turbo codes. This can be seen by noting that the integration in (83) is over the fixed point solutions of the DE equations of turbo codes. Thus, the closed-form expressions are very difficult to derive. As an example, the potential function of the uncoupled LTE turbo ensemble is shown in Fig. 22 for several values of ϵ.

Estimating the MAP threshold and proving threshold saturation on the AWGN channel for turbo codes are difficult. The only progress so far is due to the recent work in [164], which relies on Monte Carlo simulation to compute the MAP threshold from the generalized area theorem [165, Th. 1] and observes threshold saturation for SC-SCCCs on the AWGN channel numerically. In fact, various works [58], [149], [150], [151], [152] suggest that the good performance of spatially coupled turbo codes from the BEC can be carried over to the AWGN channel.

1d) Minimum Distance: Consider an uncoupled turbo-like code $\mathcal{C}'$ which can belong to PCCCs, SCCCs, and BCCs. Let $\mathcal{C}$ represent the spatially coupled turbo-like code with $\mathcal{C}'$ as the component code. Assume that the permutations of $\mathcal{C}$ are time-invariant and satisfy certain conditions. Moreover, both $\mathcal{C}$ and $\mathcal{C}'$ have the same length. The minimum distances of the coupled and uncoupled codes satisfy [149]

$$d_{\min}(\mathcal{C}) \geqslant d_{\min}(\mathcal{C}'). \tag{88}$$

Thus, spatial coupling either preserves or improves the minimum distance of turbo codes. That said, the exact analysis of the error floor performance of spatially coupled turbo codes is difficult. By (88), one can use the minimum distance of the uncoupled turbo codes as the lower bound to study the minimum distance behavior of the coupled codes [149].

2) SPATIALLY COUPLED PCCCS (SC-PCCCS)

SC-PCCCs [58] are obtained by performing spatial coupling on the PCCC encoders. The design rate of SC-PCCCs is the same as that of the underlying uncoupled PCCCs.

The encoding procedures for SC-PCCCs are as follows. An information sequence $\boldsymbol{u}_t$ at time t is divided into $m+1$ subsequences with equal length, i.e., $\boldsymbol{u}_t = [\boldsymbol{u}_{t,0}^{\mathrm{U}}, \ldots, \boldsymbol{u}_{t,t+m}^{\mathrm{U}}]$, where $t \in \{1,\ldots,L\}$. Meanwhile, at the lower encoder, $\boldsymbol{u}_t$ is interleaved becoming $\Pi_t(\boldsymbol{u}_t)$ and also divided into $m+1$ subsequences with equal length, i.e., $\Pi_t(\boldsymbol{u}_t) = [\boldsymbol{u}_{t,t}^{\mathrm{L}}, \ldots, \boldsymbol{u}_{t,t+m}^{\mathrm{L}}]$, where $\Pi_t(.)$ denotes the interleaving function at time t before coupling. The coupling is performed such that inputs to the upper and lower convolutional encoders at time t are $\Pi_t^{\mathrm{U}}([\boldsymbol{u}_{t-m,t}^{\mathrm{U}}, \ldots, \boldsymbol{u}_{t,t}^{\mathrm{U}}])$ and $\Pi_t^{\mathrm{L}}([\boldsymbol{u}_{t-m,t}^{\mathrm{L}}, \ldots, \boldsymbol{u}_{t,t}^{\mathrm{L}}])$, respectively, where $\Pi_t^{\mathrm{U}}(.)$ and $\Pi_t^{\mathrm{L}}(.)$ are the permutation functions of the upper and lower encoders, respectively, at time t. The codeword obtained at time t is $\boldsymbol{c}_t = [\boldsymbol{u}_t, \boldsymbol{v}_t^{\mathrm{U}}, \boldsymbol{v}_t^{\mathrm{L}}]$, where $\boldsymbol{v}_t^{\mathrm{U}}$ and $\boldsymbol{v}_t^{\mathrm{L}}$ are the parity sequences as the result of upper and lower convolutional component encoding at time t. It is worth mentioning that the encoding of SC-PCCCs can be performed in parallel, i.e., encoding L information sequences in parallel.

Density evolution analysis shows that SC-PCCCs have a strictly larger BP threshold than uncoupled PCCCs on the BEC [58]. Moreover, the decoding threshold of SC-PCCCs improves when employing convolutional component codes with larger states. Most importantly, by using the potential function argument [162], it was also analytically shown that SC-PCCCs have threshold saturation property [58]. In addition, the required coupling memory for observing threshold saturation numerically is small, i.e., $m \leqslant 2$. In other words, a small coupling memory is sufficient for achieving the optimal MAP decoding performance. However, the BP threshold of SC-PCCCs still has a noticeable gap to the BEC capacity when punctured.

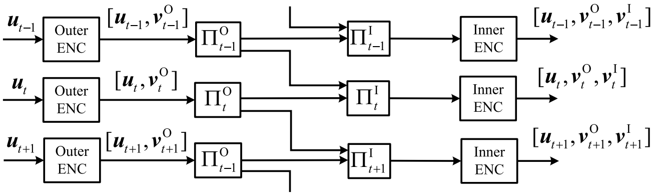

$\boldsymbol{u}_{t-1}$
Outer ENC
$[\boldsymbol{u}_{t-1}, \boldsymbol{v}^{\mathrm{O}}_{t-1}]$
Π^{O}_{t-1}
Π^{I}_{t-1}
Inner ENC
$[\boldsymbol{u}_{t-1}, \boldsymbol{v}^{\mathrm{O}}_{t-1}, \boldsymbol{v}^{\mathrm{I}}_{t-1}]$
$\boldsymbol{u}_{t}$
Outer ENC
$[\boldsymbol{u}_{t}, \boldsymbol{v}^{\mathrm{O}}_{t}]$
Π^{O}_{t}
Π^{I}_{t}
Inner ENC
$[\boldsymbol{u}_{t}, \boldsymbol{v}^{\mathrm{O}}_{t}, \boldsymbol{v}^{\mathrm{I}}_{t}]$
$\boldsymbol{u}_{t+1}$
Outer ENC
$[\boldsymbol{u}_{t+1}, \boldsymbol{v}^{\mathrm{O}}_{t+1}]$
Π^{O}_{t-1}
Π^{I}_{t+1}
Inner ENC
$[\boldsymbol{u}_{t+1}, \boldsymbol{v}^{\mathrm{O}}_{t+1}, \boldsymbol{v}^{\mathrm{I}}_{t+1}]$

FIGURE 23. SC-SCCC encoding with coupling memory $m = 1$.

3) SPATIALLY COUPLED SCCCS (SC-SCCCS)

The encoder of SC-SCCCs [58] with $m = 1$ is illustrated in Fig. 23. The design rate of SC-SCCCs is the same as that of the underlying uncoupled SCCCs.

The encoding procedures for SC-PCCCs are as follows. Similar to SC-PCCCs, at time t, information sequence $\boldsymbol{u}_t$ is encoded by the outer convolutional encoder, and parity sequence $\boldsymbol{v}_t^{\mathrm{O}}$ is obtained. Both sequences $[\boldsymbol{u}_t, \boldsymbol{v}_t^{\mathrm{O}}]$ are interleaved and divided into $m+1$ equal length subsequences, i.e., $\Pi_t^{\mathrm{O}}([\boldsymbol{u}_t, \boldsymbol{v}_t^{\mathrm{O}}]) = \tilde{\boldsymbol{v}}^{\mathrm{O}}_{t,t}, \dots, \tilde{\boldsymbol{v}}^{\mathrm{O}}_{t,t+m}$, where $\Pi_t^{\mathrm{O}}(.)$ is the permutation function of the outer component code at time t. The coupling is performed such that the input to the inner convolutional encoder at time t is $\Pi_t^{\mathrm{I}}([\tilde{\boldsymbol{v}}^{\mathrm{O}}_{t-m,t}, \dots, \tilde{\boldsymbol{v}}^{\mathrm{O}}_{t,t}])$, where $\Pi_t^{\mathrm{I}}(.)$ is the permutation function of the inner component code at time t. The parity sequence generated by the inner decoder at time t is $\boldsymbol{v}_t^{\mathrm{I}}$. Finally, the codeword at time t is $\boldsymbol{c}_t = [\boldsymbol{u}_t, \boldsymbol{v}_t^{\mathrm{O}}, \boldsymbol{v}_t^{\mathrm{I}}]$. Different from SC-PCCCs, the encoding of SC-SCCCs can only be sequential.

Interestingly, although SCCCs have a worse BP threshold than PCCCs, after coupling SC-SCCCs have a much better BP threshold than SC-PCCCs when m is large [159]. Density evolution results show that the BP threshold SC-SCCCs is within 0.001 to the BEC capacity for a wide range of code rates under random parity puncturing. The superior performance of SC-SCCCs over SC-PCCCs is due to the threshold saturation property as well as the fact that uncoupled SCCCs have a larger MAP threshold than PCCCs [159]. Simulation results in [149] show that SC-SCCCs have a lower error floor than SC-PCCCs. The performance of SC-SCCCs can be further improved by coupling a fraction of inner parity sequences from the previous time instant [166]. In addition, hardware architectures of SC-SCCC decoders are recently investigated in [167]

4) SPATIALLY COUPLED BCCS (SC-BCCS)

BCCs [153] are inherently spatially coupled codes with coupling memory $m = 1$ [58]. Spatially coupled BCCs refer to the generalization of BCCs to a large coupling memory [58]. In most cases, rate-$2/3$ convolutional codes are selected as the component codes for SC-BCCs, leading to rate $1/3$ SC-BCCs. In addition, there are two types of SC-BCCs according to [58]. Here, we only consider the type II SC-BCCs as the type I SC-BCCs can be seen as a special case of type II SC-BCCs without spatial coupling on information sequences. The encoder of type II SC-BCCs with $m = 1$ is depicted in Fig. 23.

The encoding of SC-BCCs is performed as follows. At time t, the upper parity sequences $\boldsymbol{v}_t^{\mathrm{U}}$, and the lower parity sequence $\boldsymbol{v}_t^{\mathrm{L}}$, are divided into m equal length subsequences, respectively, such that $\boldsymbol{v}_t^{\mathrm{U}} = [\boldsymbol{v}^{\mathrm{U}}_{t,t}, \dots, \boldsymbol{v}^{\mathrm{U}}_{t,t+m}]$ and $\boldsymbol{v}_t^{\mathrm{L}} = [\boldsymbol{v}^{\mathrm{L}}_{t,t}, \dots, \boldsymbol{v}^{\mathrm{U}}_{t,t+m}]$. SC-BCCs consist of information and parity coupling, where the coupled information and parity sequences are the first and second inputs of the convolutional component encoders, respectively. Specifically, the information coupling part is the same as in SC-PCCCs such that the upper and lower coupled information becomes the first inputs of the upper and lower encoders, respectively. In addition, the parity coupling is performed such that the parity sequences $[\boldsymbol{v}^{\mathrm{L}}_{t-m,t-1}, \dots, \boldsymbol{v}^{\mathrm{L}}_{t-1,t-1}]$ and $[\boldsymbol{v}^{\mathrm{U}}_{t-m,t-1}, \dots, \boldsymbol{v}^{\mathrm{U}}_{t-1,t-1}]$ are interleaved and become the second inputs of the upper and lower encoders, respectively. At time t, the component codeword is $\boldsymbol{c}_t = [\boldsymbol{u}_t, \boldsymbol{v}_t^{\mathrm{U}}, \boldsymbol{v}_t^{\mathrm{L}}]$. The encoding of SC-BCCs is sequential in nature.

It was analytically shown that SC-BCCs have the threshold saturation property. Moreover, the MAP threshold of SC-BCCs is within 0.001 to the BEC capacity for rates between $1/2$ to $9/10$ [58]. When m is small, e.g., $m = 1$, SC-BCCs have larger BP thresholds than SC-PCCCs and SC-SCCCs [58]. In addition, [149] shows that SC-BCCs exhibit a linear minimum distance growth rate, which is faster than that of SC-PCCCs and SC-SCCCs [149]. Recently, the research on mitigating error propagation in sliding window decoding of SC-BCCs was carried out in [168].

5) PARTIALLY INFORMATION-COUPLED TURBO CODES

Partially information-coupled turbo codes (PIC-TCs) were introduced in [150], [151] to improve the error performance of transport block (TB)-based HARQ in LTE. Instead of using a very long code to encode the entire information of a TB into a codeword, in PIC-TCs the information sequence of a TB is divided into several small subsequences. Each sub-sequence as well as a part of the information bits from consecutive sub-sequences are encoded into a component codeword. In other words, a faction of information bits are shared between consecutive component codewords. This introduced coupling between component codewords improves the reliability of the transmitted TBs while the spatial coupling nature of PIC-TCs allows low latency decoding via sliding window decoding. It is worth noting that the coupling of PIC-TCs is on the turbo code level while the coupling of SC-PCCCs is on the convolutional code level. In other words, existing turbo encoders and decoders can be directly employed in PIC-TCs without changing the architectures, whereas SC-PCCCs require some modifications.

The encoder diagram of PIC-TCs with $m = 1$ is shown in Fig. 25. At time t, the information sequence $\boldsymbol{u}_t$ is decomposed into m equal length subsequences $\boldsymbol{u}_{t,t}, \dots, \boldsymbol{u}_{t,t+m}$. The coupling is performed such that the input sequence of the turbo encoder at time t is $[\boldsymbol{u}_{t-m,t}, \dots, \boldsymbol{u}_{t-1,t}, \boldsymbol{u}_t]$. We denote by λ the ratio of the length of the coupled information sequence over the total information length. The

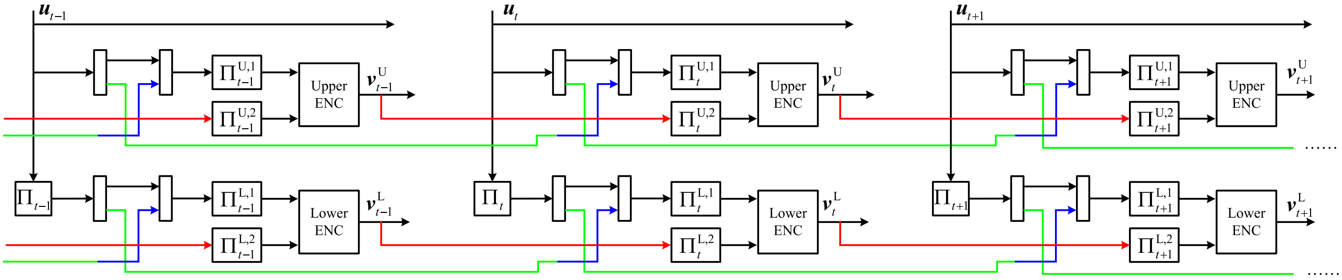


FIGURE 24. SC-BCC encoding with coupling memory $m = 1$.

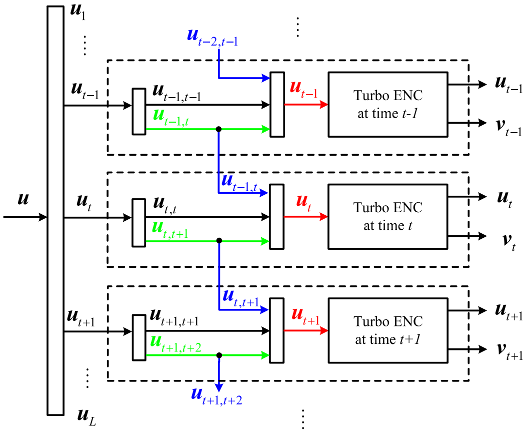


FIGURE 25. PIC-TC encoding with $m = 1$.

coupling ratio λ is an important parameter that affects the code rate, waterfall, and error floor performance of PIC-TCs. After the turbo encoding, we obtain the component codeword $\boldsymbol{c}_t = [\boldsymbol{u}_t, \boldsymbol{v}_t]$, where $\boldsymbol{v}_t$ is the parity sequence generated by the turbo encoder at time t. The design rate of PIC-TCs is

$$R = \frac{R_0(1-\lambda)}{1-\lambda R_0}, \tag{89}$$

where R_0 is the code rate of the mother turbo code.

It was demonstrated in [151] that PIC-TCs can achieve a larger BEC decoding threshold than SC-PCCCs with or without puncturing. One can also generalize PIC-TCs by coupling parity sequences to attain a decoding threshold within 0.0002 of the BEC capacity for code rates ranging from 1/3 and 9/10 [151]. Extension of PIC-TCs by employing duo-binary convolutional component codes [55] has been investigated in [169]. However, it requires a large coupling memory for these codes to achieve a BP threshold close to the MAP threshold.

6) GENERALIZED SC-PCCCS (GSC-PCCCS) AND CAPACITY-ACHIEVING

To improve the decoding threshold, [152], [170] generalized SC-PCCCs by allowing a fraction of information bits to be repeated q times before performing coupling and component code encoding. The resultant codes are called generalized SC-PCCCs (GSC-PCCCs). In fact, GSC-PCCCs inherit many useful properties from PIC-TCs and SC-PCCCs, such as that the repeated and coupled information bits are protected by component turbo codewords at multiple time instants and the threshold saturation property.

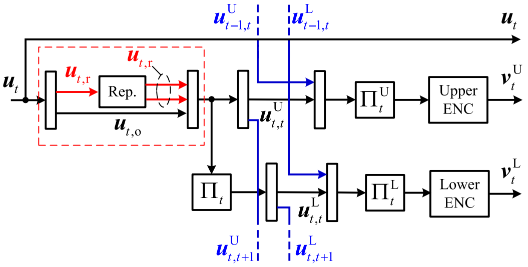


FIGURE 26. GSC-PCCCs encoding with coupling memory $m = 1$. Note that the encoder of SC-PCCCs is without the structures inside the dash line box.

The encoding of GSC-PCCCs with $m = 1$ is illustrated in Fig. 26. At time t, $\boldsymbol{u}_t$ is decomposed into $\boldsymbol{u}_{t,\mathrm{r}}$ and $\boldsymbol{u}_{t,\mathrm{o}}$. Sequence $\boldsymbol{u}_{t,\mathrm{r}}$ is repeated q times and combined with $\boldsymbol{u}_{t,\mathrm{o}}$ to form sequence $[\boldsymbol{u}_{t,\mathrm{r}}, \ldots, \boldsymbol{u}_{t,\mathrm{r}}, \boldsymbol{u}_{t,\mathrm{o}}]$. We define λ the *repetition ratio* as the length of $\boldsymbol{u}_{t,\mathrm{r}}$ over the length of $[\boldsymbol{u}_{t,\mathrm{r}}, \ldots, \boldsymbol{u}_{t,\mathrm{r}}, \boldsymbol{u}_{t,\mathrm{o}}]$. The resultant sequence is then decomposed into $m + 1$ sequences of equal length, denoted by $\boldsymbol{u}^{\mathrm{U}}_{t,t+j}$, $j = 0, \ldots, m$. The information sequence $\boldsymbol{u}^{\mathrm{U}}_{t,t+j}$ is used as a part of the input of the upper convolutional encoder at time $t + j$. Then, the coupling is performed in the same way as for SC-PCCCs. The design rate of GSC-PCCCs is

$$R = \frac{1-(q-1)\lambda}{\frac{1}{R_0}-(q-1)\lambda}, \tag{90}$$

where R_0 is the code rate of the mother turbo code. In addition, the encoding of GSC-PCCCs can also be performed in parallel.

Note that SC-PCCCs can be regarded as a special case of GSC-PCCCs with $q = 1$. Most importantly, it was rigorously proved that the rate-R GSC-PCCC ensemble with $R \in [1/(q(1/R_0-1)+1), 1)$ and 2-state convolutional component codes under suboptimal BP decoding achieves at least a fraction $1 - \frac{R}{R+q}$ of the BEC capacity for repetition factor $q \geqslant 2$, where the multiplicative gap vanishes as q tends to

TABLE 11. Simulation parameters for evaluation the performance of various interleavers and puncturing patterns.

Channel	AWGN		
Modulation	BPSK		
Generator Polynomial	$[1, 15/13]_8$		
Interleavers	ARP	DRP	LTE
Code rate	$1/3, 2/3, 4/5$		
Information Length	1504		
Decoding Algorithm	Log-MAP		
Maximum Decoding Iterations	16		

infinity [152]. This indicates that GSC-PCCCs can achieve a threshold all the way to the BEC capacity by increasing q. To the best of our knowledge, this is the first class of turbo codes that are proved to be capacity-achieving.

L. COMPARISONS: PERFORMANCE AND COMPLEXITY

In this section, we first compare the performance between interleavers and punctures designed for the enhanced turbo codes and those with LTE turbo codes. Then, we illustrate the impacts of different convolutional component codes as well as decoders on the error performance. In addition, a comparison between convolutional codes and turbo codes at short blocklength is provided. The performance of different turbo decoders and the comparison between different hardware implementations of high throughput turbo decoders are discussed. Finally, we compare the BER between different classes of spatially coupled turbo codes.

1) INTERLEAVERS AND PUNCTURING PATTERNS COMPARISON

We consider a turbo code with generator polynomial $[1, 15/13]_8$. The turbo decoder is the iterative Log-MAP decoder with 16 maximum iterations. The BER and FER of turbo codes with rates $1/3$, $2/3$, and $4/5$ with the protograph-based ARP interleavers and punctures in [89], the puncture-constrained DRP interleavers and punctures in [88], and LTE interleavers and punctures [31], are shown in Fig. 27. Note that the interleaver lengths for the ARP and LTE interleavers are 1504 while for the DRP interleaver is 1512. The simulation parameters are summarized in Table 11.

Thanks to the joint interleaving and puncturing design, both DRP and ARP interleavers lead to better waterfall and error floor performance than the LTE interleavers. Moreover, the protograph-based ARP interleavers achieve the best overall performance among all three interleavers. In particular, at rate $4/5$, the protograph-based ARP interleaver provides a gain about 0.1 dB over the DRP interleaver and 0.4 dB over the LTE interleaver at a FER of 10^{-4}. Therefore, the new ARP interleavers designed for the enhanced turbo codes can provide substantial performance gains in both low and high rates.

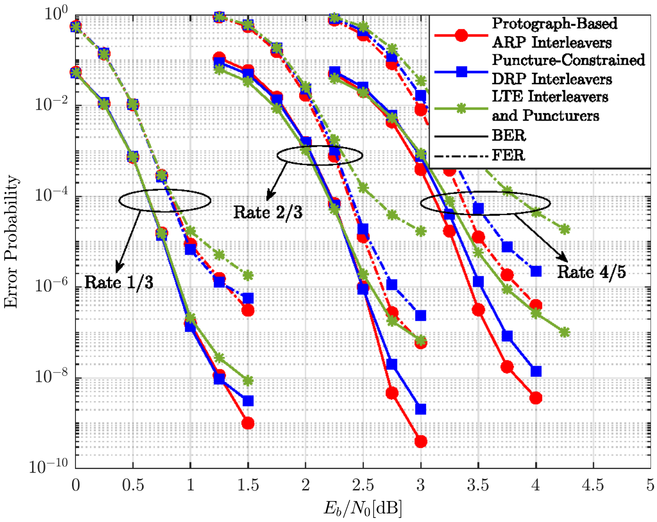


FIGURE 27. BER and FER comparison between turbo codes with information $K = 1504$ and different interleavers and puncturing patterns on the AWGN channel.

TABLE 12. Simulation parameters for evaluation the impacts on the choice of convolutional component codes.

Channel	AWGN
Modulation	BPSK
Generator Polynomial	$[1, 15/13]_8$ $[1, 15/13, 17/13]_8$ $[1, 37/25]_8$
Interleavers	ARP
Code rate	$1/3$
Information Length	$1504, 8000$
Decoding Algorithm	Log-MAP
Maximum Decoding Iterations	16

2) COMPONENT CODE COMPARISON

We investigate the impacts of convolutional component codes on the performance of turbo codes. We consider two information lengths $K = 1504, 8000$. Same as in the previous section, we set the maximum iteration of the Log-MAP turbo decoder to be 16. The BER of turbo codes with generator polynomials $[1, 15/13]_8$, $[1, 15/13, 17/13]_8$, and $[1, 37/25]_8$, which we referred to as TC1, TC2, and TC3, respectively, are shown in Fig. 28. All turbo codes with the same information length adopt the same ARP interleavers from [35]. Since TC2 is with rate $1/5$, we use the puncturing patterns from the third row of [171, Table 2.8] to achieve rate $1/3$. The simulation parameters are summarized in Table 12.

Fig. 28 shows that TC3 has the best waterfall performance. However, it suffers from a higher error floor. A tailored interleaver design for TC3 is necessary to lower its error floor. It is also interesting to note that TC2 achieves slightly better waterfall and error floor performance than TC1. This implies that a low-rate turbo code with carefully designed puncturing patterns can outperform a high-rate turbo code without puncturing.

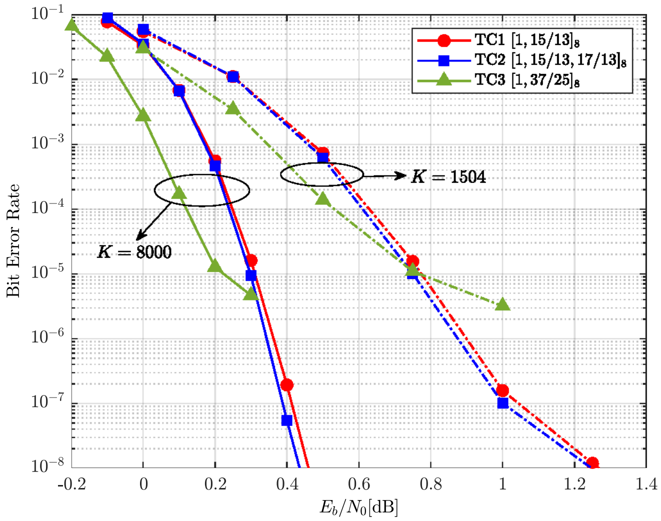


FIGURE 28. BER comparison between turbo codes with rate-1/3, information length $K \in \{1504, 8000\}$, and different convolutional component codes.

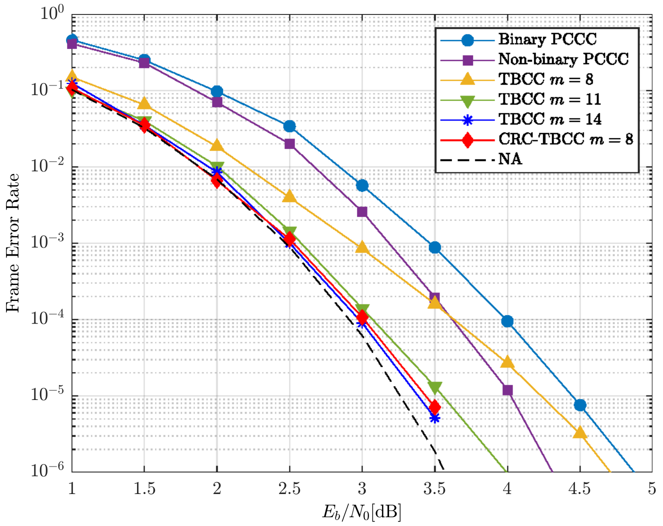


FIGURE 29. FER Comparison between convolutional and turbo codes with rate 1/2 and information length $K = 64$.

TABLE 13. Simulation parameters for evaluation the performance of short convolutional and turbo codes.

Channel	AWGN
Modulation	BPSK
Coding schemes	Binary TC, non-binary TC, TBCC, CRC-TBCC
Code rate	1/2
Information Length	64
Decoding Algorithm	Log-MAP, WAVA, CA-LVA

3) SHORT BLOCKLENGTH CONVOLUTIONAL CODES AND TURBO CODES COMPARISON

We compare the error performance between turbo codes and convolutional codes with information length $K = 64$ and rate 1/2. We consider a binary turbo with $m = 4$ tail-biting RSC component codes from [172] and a non-binary turbo code with $m = 1$ tail-biting RSC component codes over $\mathbb{F}_{256}$ from [173], where all component codes are under BCJR decoding. We further consider three TBCCs with memories $m = 8, 11, 14$ under WAVA [25]. In addition, we include a TBCC with $m = 8$ under CRC-aided list Viterbi algorithm (CRC-LVA) with a 10-bit CRC optimized based on distance spectrum [53]. The simulation parameters are summarized in Table 13. Fig. 29 shows the FER versus E_b/N_0 in dB for the aforementioned candidate codes. Normal approximation (NA) (see Section IV-C) with the third order term [23], [174] based on BPSK signaling is used as the benchmark.

As can be seen in this figure, the TBCC with $m = 14$ closely approaches the NA and is within 0.1 dB at a FER of 10^{-5}. The TBCC under CRC-LVA provides comparable performance to the TBCC with $m = 14$ but with 98% reduction in decoding complexity [53, p20]. At short blocklength, TBCC can significantly outperform turbo codes provided that m is very large. However, at a FER of 10^{-6}, the non-binary turbo code is about 0.4 dB better than the TBCC with $m = 8$. Finally, the binary turbo code has the worst FER performance but the lowest decoding complexity. The above promising results confirm that TBCCs are very powerful short blocklength codes.

4) TURBO DECODERS COMPARISON

We compare the error correction performance of the Log-MAP decoder [54], scaled Max-Log-MAP decoder [54] with a fixed scaling factor of 0.75, the local SOVA decoder with the third configuration in [121, Sec. IV-C] and the original SOVA decoder [50]. The simulations were carried out for rate-1/3 LTE turbo codes with $K = 1056$. The maximum number of iterations for each decoder is set to 5.5, where one convolutional component decoding is regarded as a half iteration. The BER curves for all decoders are shown in Fig. 30.

We see that the performance of the local SOVA decoder and the scaled Max-Log-MAP decoder is almost identical. However, the local SOVA has a lower computational complexity than the scaled Max-Log-MAP decoder when high-radix decoding is employed [121]. It was shown that for radix-4 and radix-8, using the local-SOVA reduces the complexity by 27% compared to the Max-Log-MAP decoder [131]. Moreover, the local SOVA decoder outperforms the original SOVA decoder and has about 0.1 dB loss compared to the Log-MAP decoder. Hence, the local SOVA decoder provides a good trade-off between error correction performance and computational complexity.

5) COMPARISON BETWEEN DIFFERENT DECODER HARDWARE ARCHITECTURES

We compare the turbo decoder architectures UXMAP, PMAP, FPMAP, and XMAP with a throughput of more than 1 Gb/s

TABLE 14. Comparison of implementation results for different turbo decoder architectures. Frequency scaling to 28 nm (capped at 1000 MHz): † 2.52; ‡ 1.95; $^{\mathcal{S}}$ 1.46. Area scaling to 28 nm: $^{\flat}$ 0.40; $^{\natural}$ 0.51; $^{\sharp}$ 0.69.

Implementation	[140]	[134]	[133]	[176]	[177]	[137]	[136]
Architecture	Radix-4 UXMAP		Radix-2 PMAP	Radix-4 PMAP	Radix-2 FPMAP	Radix-4 XMAP	
Information length	512	128/64/32	6144	6144	6144	6144	6144
Rate	1/3	1/3	1/3	0.95	1/3	1/3	0.94
Parallelism	-	128	64	32	6144	64	32
Channel values quantization	6 bits	6 bits	7 bits	-	4 bits	5 bits	6 bits
Extrinsic values quantization	-	7 bits	7 bits	-	6 bits	6 bits	7 bits
State metrics quantization	-	-	9 bits	10 bits	6 bits	10 bits	11 bits
Branch metrics quantization	-	-	8 bits	-	6 bits	9 bits	-
Max iterations	2.5	4	5.5	5.5	39	5.5	7
Technology	28 nm	28 nm	90 nm $^{\dagger\flat}$	65 nm $^{\ddagger\natural}$	65 nm $^{\ddagger\natural}$	45 nm $^{\mathcal{S}\sharp}$	28 nm
Frequency [MHz]	800	800	625 (1000)	410 (1000)	100 (252)	600 (1000)	625
Throughput [Gb/s]	409.6	102.4	3.3 (5.29)	1.01 (2.47)	15.8 (39.86)	1.67 (3.2)	1.13
Area [mm^2]	30	16.54	19.75 (2.44)	2.49 (0.55)	109 (24.09)	2.43 (1.03)	0.49
Area efficiency [Gb/s/mm^2]	13.65	6.19	0.17 (2.17)	0.41 (4.49)	0.14 (1.65)	0.69 (2.68)	2.32

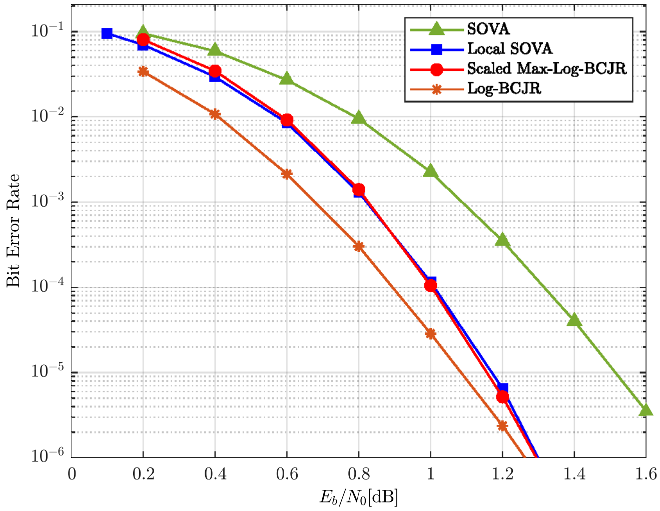


FIGURE 30. BER comparison between different turbo decoders for decoding rate-$1/3$ LTE turbo codes with information length $K = 1056$.

reported in the literature. The decoding is the scaled Max-Log-MAP decoder with a scaling factor of 0.75. Table 14 shows different implementation results of the above four types of architectures, where the term "parallelism" refers to the number of bits decoded in parallel. Since results from the literature are reported for different technology nodes, we provide a scaling to 28 nm technology in the caption of Table 14. To this end, we cap the frequency scaling to a reasonable 1000 MHz, which allows to preserve single cycle accesses to static random access memory. In addition, except for the UXMAP architectures [134], [140] that consider ARP interleavers [89], the rest of the implementations focus on the LTE interleavers.

Although with scaling to 28 nm technology, neither the PMAP decoders nor the XMAP decoders have a throughput close to 15 Gb/s. The FPMAP implementation from [176] can achieve a throughput close to 40 Gb/s when scaled to 28 nm technology. In contrast, the UXMAP can achieve a throughput of more than 100 Gb/s and also the highest area efficiency.

6) COMPARISON BETWEEN SPATIALLY COUPLED TURBO CODES AND UNCOUPLED TURBO CODES

We compare spatially coupled turbo codes and uncoupled turbo codes on the BEC. We fix all code rates to be $1/3$. For the uncoupled codes, we consider a regular turbo code with generator polynomial $[1, 15/13]_8$ and the irregular turbo codes from [72, Fig. 4] that have the largest BEC threshold. For the spatially coupled turbo codes, we consider SC-PCCCs [58], PIC-TCs with coupling ratio $\lambda = 0.4$ [151, Table II], and GSC-PCCC with repetition ratio $\lambda = 0.165$ and repetition factor $q = 4$ [152, Table I]. All coupled codes have convolutional component codes with generator polynomial $[1, 5/7]_8$, component code information length $K = 10000$, coupling memory $m = 1$, and coupling length $L = 100$. Moreover, all coupled codes are under full decoding of the entire spatial code chain whereas the maximum component code decoding iterations are 20. In addition, we adopt random interleavers and periodic parity puncturing patterns for all coupled codes. The bit erasure rates are shown in Fig. 31. Note that for uncoupled turbo codes, we plot their BEC decoding thresholds, representing their asymptotic performance as the blocklengths go to infinity. The decoding thresholds for the coupled codes and the BEC capacity are also included in the same figure.

It can be seen that all coupled codes under finite blocklength outperform the regular turbo code with infinite blocklength at a BER of 10^{-5}. Moreover, all coupled codes have decoding thresholds close to the BEC capacity. Most notably, the GSC-PCCC under finite blocklength also has a noticeable performance gain over the irregular turbo

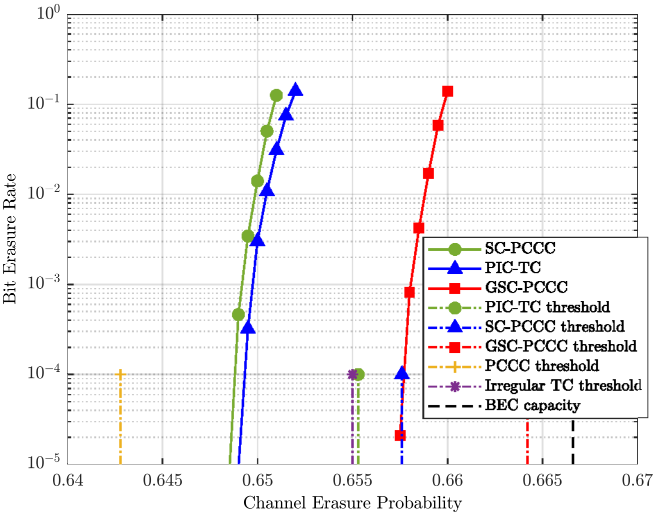


FIGURE 31. BER Comparison between rate-1/3 spatially coupled turbo codes and uncoupled turbo codes on the BEC.

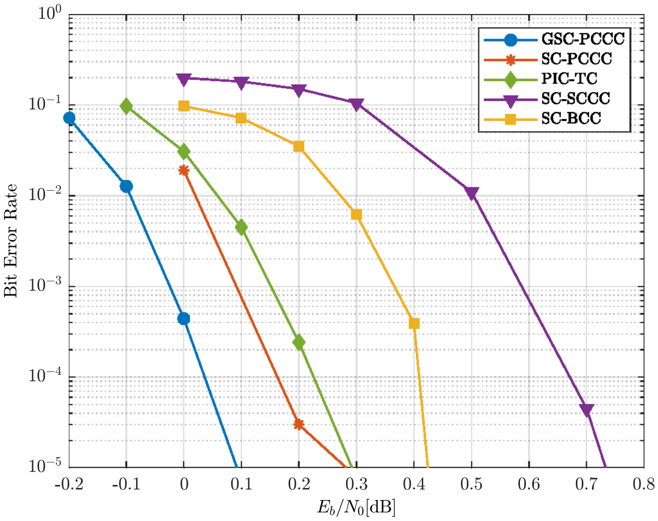


FIGURE 32. BER Comparison between different spatially coupled turbo codes with $K = 1000$, $m = 1$, and $L = 100$ under sliding window decoding with window size 10.

code with infinite blocklength. It is worth noting that the performance of all coupled codes can be further improved by increasing K and m. Hence, spatial coupling provides new degrees of freedom in designing good codes based on existing component codes.

7) SPATIALLY COUPLED TURBO CODES COMPARISON

We compare the BER performance between various coupled codes, including SC-PCCCs [58], SC-SCCCs [58], SC-BCCs [58], PIC-TCs with $\lambda = 0.24$ [151], and GSC-PCCCs with $\lambda = 0.11, q = 4$ [152], on the AWGN channel. All coupled codes have $K = 1000$, $m = 1$, $L = 100$, and are under sliding window decoding with window size 10. The convolutional component codes for SC-PCCCs, PIC-TCs, and GSC-PCCCs are with generator polynomial $[1, 15/13]_8$, for SC-SCCCs are $[1, 5/7]_8$, and for SC-BCCs are $\begin{bmatrix} 1 & 0 & 1/7 \\ 0 & 1 & 5/7 \end{bmatrix}_8$. For PIC-TCs and GSC-PCCCs, we adopt the LTE interleavers and periodic parity puncturing patterns. The BER is shown in Fig. 32, where the BER plots for SC-PCCCs, SC-SCCCs, and SC-BCCs are taken from [177], [178].

The coupled codes whose component codes are PCCCs or turbo codes have better performance than those with other component codes. Among them, the GSC-PCCC has the best waterfall performance. However, the SC-SCCC and SC-BCC have better error floor performance as their BER slopes at 10^{-5} are steeper compared to the other three coupled codes. Note that the finite length performance of spatially coupled turbo codes depends on various factors such as K, m, and the decoding window size, interleavers, the choice of coupling bit indices, etc. [179]. Thus, the design of spatially coupled turbo codes to attain the best finite length performance for a given decoding latency [161] is still an ongoing research topic.

Finally, we present the comparison between spatially coupled turbo codes and SC-LDPC codes. Since GSC-PCCCs achieve the best BER as shown in Fig. 32, we pick them as the candidate codes and set $q = 2$, $\lambda = 0.335$, $k = 1000$, $m = 1$, and $L = 50$. The benchmark regular $(3, 6)$ protograph SC-LDPC codes are constructed by following [180], with $m = 2$, $L = 50$, and a lifting factor of 1000. In addition, we consider four code rates, i.e., 1/2, 2/3, 3/4, and 4/5. The BER results of rate-1/2 codes are available in [152]. For other code rates, the GSC-PCCCs are interleaved and punctured by using the ARP interleavers and puncturers in [89], respectively. For SC-LDPC codes, we note that randomly punctured SC-LDPC codes are capacity-approaching provided that the mother codes are capacity-approaching [181]. However, we use fixed and period puncturing patterns following the design in [182, Sec. VII-A], which have slightly better error performance than random puncturing patterns. The maximum intra-block and inter-block decoding iterations for the GSC-PCCCs are set to 20 while the maximum BP decoding iterations for the SC-LDPC codes are set to 1000. The BER performance of these codes on the AWGN channel is shown in Fig. 33.

It can be seen that GSC-PCCCs have better waterfall performance than SC-LDPC codes for all the considered code rates. Interestingly, the performance gains of GSC-PCCCs over SC-LDPC codes at high rates are larger than those at low rates. Hence, the interleavers and puncturers designed for uncoupled turbo codes are also effective for coupled turbo codes. However, SC-LDPC codes have better error floor than GSC-PCCCs. Since the candidate GSC-PCCCs have a fraction of information bits repeated twice before component turbo encoding, additional design criteria are required to obtain the optimal interleavers and puncturers in this case.

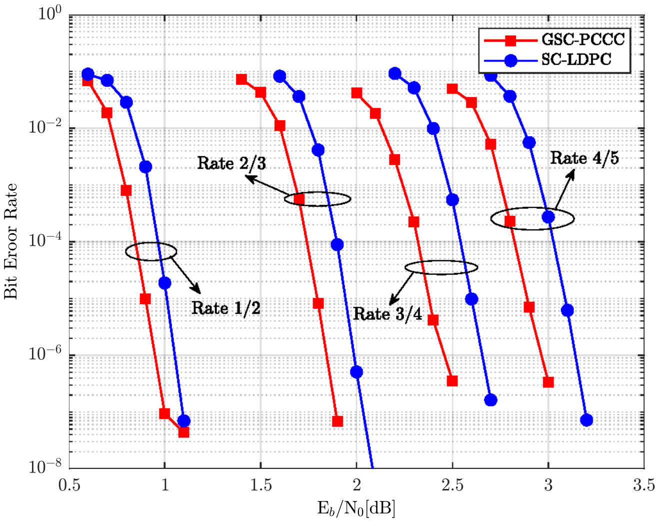


FIGURE 33. BER Comparison between GSC-PCCCs with $m = 1$ and SC-LDPC codes with $m = 2$ at different rates. For both codes, the component code information length and coupling length are $K = 1000$ and $L = 50$, respectively.

M. NEW RESEARCH DIRECTIONS

In light of all the above sections, the following new directions can be considered to further improve the classes of turbo codes suitable for future communication systems.

- As shown in Fig. 28, the choice of convolutional component codes affects the waterfall and error floor of the resultant turbo codes. It is worth extending the puncture-constrained ARP interleavers design methods from [89] to turbo codes with other convolutional component codes that have better decoding thresholds.
- Various works have demonstrated the necessity of joint interleaving and puncturing pattern designs to ensure that the resultant turbo codes have a good waterfall and error floor performance. However, to fulfill the threshold requirements, most designs use the EXIT charts which rely on Monte Carlo simulation. Alternatively, one can explore new design methods based on the density evolution for which the exact transfer function can be derived.
- It has already been demonstrated that the local SOVA decoder can achieve the same performance as the Max-Log-MAP decoder with lower complexity. It is worth to investigate efficient hardware implementations of high radix schemes for the local SOVA decoder with ultra-high throughputs, e.g., over 1 Tb/s.
- Spatial coupling can boost the performance of turbo codes without the need for meticulous optimization as in irregular codes. Several classes of spatially coupled turbo codes exhibit the threshold saturation property such that the threshold under suboptimal decoding can approach to that under the optimal MAP decoding. In this regard, it would be interesting to investigate whether the use of suboptimal Max-Log-MAP or the local SOVA algorithm as the component code decoders can still have threshold saturation. Finally, the designs of interleavers, puncturers, and the choice of coupling bits for spatially coupled turbo codes require further investigation.

VI. LOW-DENSITY PARITY-CHECK (LDPC) CODES

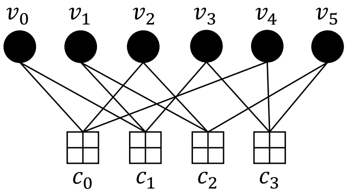
LDPC codes, invented by Gallager in the early 1960s [183] and rediscovered by MacKay and Neal in the mid-1990s [184], [185], [186], are a class of linear block codes capable of performing extremely close to the channel capacity. More importantly, this performance is achieved under the iterative *belief propagation* (BP) decoding [187] (also known as message-passing (MP) decoding), which has complexity linear in the blocklength of the codes. Quasi-cyclic (QC) LDPC codes are a type of LDPC codes with notable importance in the efficient encoding and practical implementation of the corresponding decoder due to their compact representation of the parity-check matrix and high level of parallelism in the decoder architecture. In the new era of mobile communications, where high data rates, low latency, and extremely reliable transmission become more demanding, QC-LDPC codes have been selected as one of the channel coding schemes in the 5th generation NR standards by 3GPP for the usage scenarios of eMBB [188].

In this section, we will start with the fundamentals of various LDPC code ensembles and the design techniques. This will be followed by a comprehensive examination of different designs of QC-LDPC codes and variations of BP decoding. Furthermore, we will discuss the currently standardized LDPC codes and provide an overview of the state-of-the-art implementations of LDPC codes for achieving ultra-high throughput in future applications. The section concludes by presenting several research directions for future 6G.

A. LDPC BLOCK CODES

An $[N, K]$ LDPC code is defined by an $M \times N$ binary parity-check matrix $\boldsymbol{H}$, where $M = N - K$ represents the number of parity bits, N is the code length, and K is the length of the information bits. The rate of the code is $R = K/N$. The code is the set of all length-N binary vectors $\boldsymbol{c}$, called *codewords*, satisfying $\boldsymbol{c}\boldsymbol{H}^T = \boldsymbol{0}$. Such a code is said to be *linear* in that a linear combination of codewords yields another codeword. The *Tanner graph* [91] representation of an LDPC code is a bipartite graph consisting of a set $\{v_0, v_1, v_2, \ldots, v_{N-1}\}$ of N variable nodes (VNs) and a set $\{c_0, c_1, c_2, \ldots, c_{M-1}\}$ of M check nodes (CNs). An edge connecting VN v_j to CN c_i if and only if $\boldsymbol{H}_{i,j} = 1$. The number of 1s of each row of $\boldsymbol{H}$ is the degree of the corresponding CN. Similarly, the number of 1s of each column of $\boldsymbol{H}$ is the degree of the corresponding VN.

In LDPC code design, an *ensemble* of codes is a family of codes that is characterized by the same set of parameters.



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

FIGURE 34. An example of an LDPC code in Tanner graph and matrix representations.

The parity-check matrix $\boldsymbol{H}$ of the $[N, K]$ LDPC code is (randomly) chosen from the ensemble of $M \times N$ binary matrices. Any code from the ensemble will have a code rate equal to or greater than $R = 1 - M/N$, and hence R is commonly known as the *design rate*. Let $\lambda(x) = \sum_{d=1}^{d_v} \lambda_d x^{d-1}$ and $\rho(x) = \sum_{d=1}^{d_c} \rho_d x^{d-1}$ be the *degree distribution* of an LDPC code, where d_v and d_c denote the maximum VN and CN degrees, respectively. The constant λ_d represents the fraction of edges of the Tanner graph which are connected to VNs of degree d, and ρ_d represents the fraction of edges of the Tanner graph which are connected to CNs of degree d.

Fig. 34 shows an example of Tanner graph and matrix representation of a $[N, K] = [6, 3]$ LDPC code. The code has a constant VN and CN degrees of 2 and 3, respectively, which corresponds to a constant number of 1s in each column and each row of the parity-check matrix $\boldsymbol{H}$. Since the rank of $\boldsymbol{H}$ is 3, the rate of this code is $1/2$ which is larger than the design rate of this code $R = 1 - 4/6 = 1/3$.

1) REGULAR ENSEMBLE

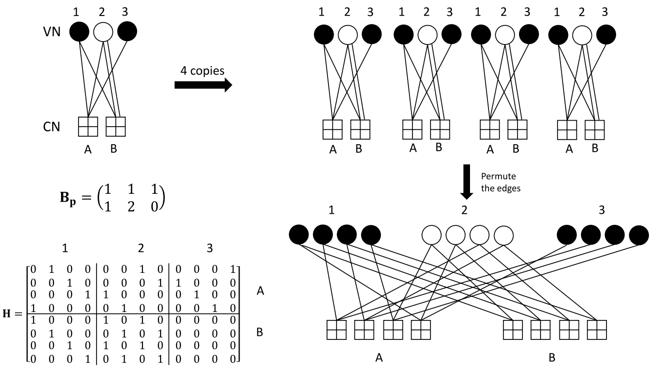
An LDPC code is said to be regular if all the $M \times N$ parity-check matrix $\boldsymbol{H}$ has the same number of 1s in each row and each column. The degree distribution of a regular ensemble is defined as $\lambda(x) = x^{d_v - 1}$ and $\rho(x) = x^{d_c - 1}$ with $\lambda_{d_v} = \rho_{d_c} = 1$, or (d_v, d_c) for short. The design rate of a regular LDPC ensemble is $R = 1 - d_v/d_c$.

2) IRREGULAR ENSEMBLE

An irregular LDPC code has a range of VN and CN degrees, that is, $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_{d_v})$ and $\boldsymbol{\rho} = (\rho_1, \rho_2, \ldots, \rho_{d_c})$ with $\sum_d \lambda_d = 1$ and $\sum_d \rho_d = 1$. The design rate of the ensemble is then $R = 1 - \int_0^1 \rho(x) / \int_0^1 \lambda(x)$. It is well-known from the literature that irregular LDPC codes can approach the channel capacity if properly designed [70], [189], [190], [191].

3) PROTOGRAPH ENSEMBLE

An ensemble of protograph LDPC codes [92] is defined by a Tanner graph with a relatively small number of VNs and CNs, namely *protograph*. A protograph $\mathcal{P} = (\mathcal{V}, \mathcal{C}, \mathcal{E})$ consists of a set of variable nodes $\mathcal{V} = \mathcal{V}_{pun} \cup \mathcal{V}_{tran}$, a set of check nodes $\mathcal{C}$, and a set of edges $\mathcal{E}$, each VN and CN nodes is of its own type. Note that $\mathcal{V}_{pun}$ and $\mathcal{V}_{tran}$ denote the set of punctured VNs and the set of transmitted VNs, respectively. Each edge $e \in \mathcal{E}$ connects a variable node of type $v \in \mathcal{V}$ to a check node of type $c \in \mathcal{C}$. The protograph is equivalently described by an $m_p \times n_p$ non-binary integer matrix $\boldsymbol{B}_p$, namely *protomatrix*, with $m_p = |\mathcal{C}|$ and $n_p = \hat{n}_{pu} + \hat{n}_t = |\mathcal{V}|$, where $\hat{n}_{pu}$ and $\hat{n}_t$ denote the number of punctured and transmitted columns in $\boldsymbol{B}_p$, respectively. Each entry $\boldsymbol{B}_p(i, j)$, $i = 0, 1, \ldots, m_p - 1$, $j = 0, 1, \ldots, n_p - 1$, represents the number of edges connecting variable node type v_j to check node type c_i. The design rate of the protograph ensemble is $R = (n_p - m_p)/\hat{n}_t$.



$$\mathbf{B_p} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 0 \end{pmatrix}$$

$$\mathbf{H} = \begin{array}{c} \left[\begin{array}{cccc|cccc|cccc} \multicolumn{4}{c}{1} & \multicolumn{4}{c}{2} & \multicolumn{4}{c}{3} \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ \hline 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \end{array}\right] \end{array} \begin{array}{c} \\ \\ \mathrm{A} \\ \\ \\ \\ \mathrm{B} \\ \\ \\ \end{array}$$

FIGURE 35. An example construction of a protograph LDPC code.

3a) Graph Representation: The Tanner graph of a protograph-based LDPC code with length $N = Zn_p$ is obtained by *lifting* the protograph $\mathcal{P}$. The lifting process is described as follows: each edge in the protograph becomes a bundle of Z edges, connecting Z copies of a VN to Z copies of a CN. The connections within each bundle are then permuted between the variable and check node pairings. This process is equivalent to replacing every element in $\boldsymbol{B}_p$ by an $Z \times Z$ square matrix, while the connections between different node types remain unchanged. The square matrix can be considered as a permutation matrix of each edge type that connects a VN type to a CN type. The lifted Tanner graph is also known as *derived graph*. Since $\boldsymbol{B}_p$ can contain integers greater than 1, it is usually required that the lifting factor Z is greater than or equal to the largest value in $\boldsymbol{B}_p$ so that no overlapping edges in the resulting derived graph after the permutation. A protograph code ensemble is defined by randomizing over all possible permutations during the lifting process.

Fig. 35 illustrates the process of obtaining a derived graph through the lifting of a protomatrix $\boldsymbol{B}_p$ with $m_p = 2$ and $n_p = 3$. The design rate of the code is $R = (3 - 2)/(3 - 1) = 1/2$ since the VN 2 is punctured. The parity-check matrix $\boldsymbol{H}$ is obtained by lifting (replacing) each entry in $\boldsymbol{B}_p$ with a 4×4 square matrix. The identity matrix and its shifted version are used in this example to perform the lifting of non-zero entries in $\boldsymbol{B}_p$ and the zero matrix is used for 0s.

3b) Minimum Distance: To determine whether or not the minimum distance of typical LDPC codes in a protograph ensemble increases linearly with code length N, the normalized logarithmic asymptotic weight distribution $r(\delta)$ for a given protograph $\mathcal{P}$ can be expressed as [192]

$$\hat{n}_t r(\delta) = \max_{\boldsymbol{\delta}_t:|\boldsymbol{\delta}_t|=\hat{n}_t\delta} \left\{ \sum_{i=1}^{m_p} \phi^{c_i}(\boldsymbol{\delta}_i) - \sum_{j=1}^{n_p} \left(d_{v_j}-1\right)\mathcal{H}(\delta_j) \right\}, \quad (91)$$

where δ denotes the normalized weight of the protograph, $\hat{n}_t$ denotes the transmitted VNs of the protograph and d_{v_j} denotes the degree of the j-th VN. The subvector $\boldsymbol{\delta}_t = (\delta_1, \delta_2, \ldots, \delta_{\hat{n}_t})$ denote the normalized partial weights of the transmitted VNs, and $\mathcal{H}(x) = -x\ln x - (1-x)\ \ln(1-x)$ is the binary entropy function. Moreover, the function

$$\phi^{c_i}(\boldsymbol{\delta}_i) = \limsup_{Z\to\infty} \frac{\ln A^{c_i}_{Z\boldsymbol{\delta}_i}}{Z}$$

denotes the normalized logarithmic asymptotic weight distribution for CN c_i with normalized partial weight vector $\boldsymbol{\delta}_i$, where $A^{c_i}_{Z\boldsymbol{\delta}_i}$ is the weight enumerator of the partial weights of the d_{c_i} VNs connected to check node c_i. The first zero-crossing of the function $r(\delta)$ at $\delta = \delta_{min} > 0$ is called the *typical minimum distance ratio*, which shows a high probability that the minimum distance of most LDPC codes in the ensemble increases linearly with N with proportionality constant δ_{min}.

3c) Types of Protograph Codes: Protograph code properties and design methods were studied in [192], [193], [194], [195], among many other works. Various examples of popular code designs were developed based on the protograph framework. Fig. 36 shows the protograph of several popular code constructions. The protograph for a rate $1/2$ systematic repeat accumulate (RA) code [196] is shown in Fig. 36-a). If an accumulator is cascaded at the input of the systematic RA code, a performance improvement in the waterfall can be achieved, at the expense of a modest increase in the decoding complexity. This type of code is called an accumulate-repeat-accumulate (ARA) code [197] and is shown in Fig. 36-b). Alternatively, the use of another accumulator at the output of an RA code can lead to better performance in the error floor region. This is called a repeat-accumulate-accumulate (RAA) structure and is shown in Fig. 36-c). Fig 36-d) shows another code structure known as the accumulate-repeat-jagged-accumulate (ARJA) which has good performance in the waterfall and error floor regime.

Furthermore, the protograph LDPC ensemble has also been widely adopted in the design of quasi-cyclic (QC) LDPC codes in various communication standards such as WiFi, e.g., [198], WiMAX and 5G New Radio (NR), e.g., [199]. The standardized AR4JA codes [200], which belong to the family of ARJA codes, are also used for next-generation deep-space communications.

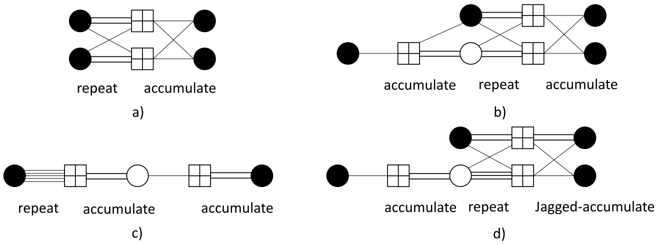


FIGURE 36. Protograph for popular code constructions: a) repeat-accumulate (RA) structure; b) accumulate-repeat-accumulate (ARA) structure; c) repeat-accumulate-accumulate (RAA) structure; d) accumulate-repeat -jagged-accumulate (AR4JA) structure. Puncturing VNs (denoted by a white circle) ensures that all code ensembles have a design rate of 1/2.

The protograph-based raptor-like (PBRL) LDPC codes [201], [202] are a class of LDPC codes that have the basic structure of Raptor codes [203] and LT codes [204], where the raptor-like code structure enables the design of rate-compatible protograph LDPC code families with efficient encoding and decoding. The protograph of a PBRL LDPC code ensemble can be described as

$$\boldsymbol{B}_{\mathbf{PBRL}} = \begin{bmatrix} \boldsymbol{B}_{\mathrm{HRC}} & \mathbf{0} \\ \boldsymbol{B}_{\mathrm{IRC}} & \boldsymbol{I} \end{bmatrix}, \quad (92)$$

where:

- $\boldsymbol{B}_{\mathrm{HRC}}$: protograph of a highest-rate code (HRC),
- $\boldsymbol{B}_{\mathrm{IRC}}$: protograph of an incremental redundancy code (IRC),
- $\mathbf{0}$: all-zeros matrix,
- $\boldsymbol{I}$: identity matrix.

The overall protograph is lifted to produce the derived code. After lifting, the HRC portion of the code structure is identical to the precode in a Raptor code. Similar to Raptor codes, where a precoded $[N, K]$ block code is coded by an additional LT code with a specific degree distribution, the degree one VN in $\boldsymbol{I}$ can be efficiently encoded as modulo-2 sums of the precode symbols by $\boldsymbol{B}_{\mathrm{HRC}}$ in the manner similar to the LT code in Raptor codes. The encoding of PBRL codes can be divided into two steps, 1) encoding of the precode, in this case is the HRC code $\boldsymbol{B}_{\mathrm{HRC}}$, and 2) the encoding of IRC code, which involves XOR operations only.

B. DESIGN AND ANALYSIS TOOLS FOR LDPC CODES

1) DENSITY EVOLUTION

As the code length $N \to \infty$, the *asymptotic decoding threshold* of an LDPC ensemble represents the *capacity* of the ensemble, which distinguishes between *reliable* and *unreliable* communication in the limit of infinite blocklength codes. Similar to the design of turbo codes described in Section V, the asymptotic decoding threshold of an LDPC code can be calculated using *density evolution* (DE) [70], and hence, to obtain the optimal degree distributions $\lambda(x)$ and $\rho(x)$ of an LDPC code ensemble. For a given degree distribution pair, DE calculates the decoding threshold by tracking the evolution of the messages passed between VNs and CNs during the iterative BP decoding process. For the binary erasure channel (BEC), the erasure probability ϵ is used as the metric to measure the reliability of the decoded messages. For the binary-input additive white Gaussian noise (BI-AWGN) channel, the probability density of the messages is used. The optimal degree distribution of an LDPC ensemble can be determined such that the probability of error converges to zero for the largest value of the channel parameter, e.g., the erasure probability ϵ for the BEC channel and the noise standard deviation σ for the BI-AWGN channel.

1a) DE on the BEC: For a given degree distribution pair $(\lambda(x), \rho(x))$, the recursion formula of DE for iterative BP decoding over the BEC is given by

$$p_e^{(\ell)} = p_e^{(0)}\lambda\left(1-\rho\left(1-p_e^{(\ell-1)}\right)\right), \tag{93}$$

where $p_e^{(0)} = \epsilon$ and $p_e^{(\ell)}$ denotes the probability of a variable to check node message in decoding iteration ℓ is erased. The DE threshold ϵ^* of an LDPC ensemble over the BEC is defined as $\epsilon^* \triangleq \sup\{\epsilon > 0 | \lim_{\ell\to\infty} p_e^{(\ell)} = 0\}$

1b) DE on the BI-AWGN channel: The DE analysis of an LDPC ensemble defined by $\lambda(x)$ and $\rho(x)$ on the BI-AWGN channel is similar to that on the BEC. The difference is that, during the recursion of the DE process, *real-valued* space messages, known as the *log-likelihood ratio* (LLR) of the probability of a given bit being 0 or 1, are passed among the edges of a Tanner graph. Therefore, instead of a single probability, the probability densities of LLRs (viewed random variables) need to be tracked. Let $P^{(\ell)}$ be the probability density function (PDF) of the LLR passed from VN to CN at iteration ℓ. The DE analysis over the BI-AWGN channel is given by the recursion formula

$$P^{(\ell)} = P^{(0)} \circledast \lambda^{\circledast}\left(\Gamma^{-1}\left(\rho^{\circledast}\left(\Gamma\left(P^{(\ell-1)}\right)\right)\right)\right), \tag{94}$$

where $\circledast$ denotes convolution and

$$\lambda^{\circledast}(x) = \sum_{d=1}^{d_v} \lambda_d x^{\circledast(d-1)}, \tag{95}$$

$$\rho^{\circledast}(x) = \sum_{d=1}^{d_c} \rho_d x^{\circledast(d-1)}. \tag{96}$$

The operator Γ corresponds to the density change processed at the CNs. An example of such a process is to let $\Gamma = \Phi(x) = -\log(\tanh(x/2))$, which is the CN processing function adopted in the sum-product algorithm (SPA). The density $P^{(0)}$ is the initial PDF of the LLR received from the channel. Assume that the all-zero codeword is transmitted, or the all $+1$ signal vector is transmitted according to the BPSK modulation mapping, i.e., $0 \to +1$ and $1 \to -1$. The probability of bit error after iteration l is equal to the probability that a VN LLR is negative-valued, which is given by

$$p_{\text{err}}^{(\ell)} = \int_{-\infty}^{0} P^{(\ell)}(x)dx. \tag{97}$$

The DE threshold of an LDPC ensemble over a BI-AWGN channel is defined as $\sigma^* \triangleq \sup\{\sigma > 0 | \lim_{\ell\to\infty} p_{\text{err}}^{(\ell)} = 0\}$.

2) GAUSSIAN APPROXIMATION

Although DE provides an exact decoding threshold for a given degree distribution pair, the computational complexity is very high. To simplify the calculation process of DE, Gaussian approximation (GA) [205] reduces the complexity of DE by assuming all messages are Gaussian distributed and that the mean of any message is equal to one-half of its variance. The approach greatly simplifies DE since only the mean value of the messages need to be tracked.

3) EXTRINSIC INFORMATION TRANSFER (EXIT) CHART

Another approach for approximate decoding threshold is that of the EXIT chart [206]. The EXIT chart was first introduced for turbo codes [76], where the details can be found in Section V-C3. Then, the EXIT chart analysis was extended to design the degree distributions of LDPC codes [78]. Unlike DE tracks the density of messages, the EXIT chart tracks the evolution of the average *extrinsic mutual information* $I_{E,V}$ and $I_{E,C}$ passed from VNs to CNs and CNs to VNs, respectively. Compared to DE, the EXIT chart analysis is simple to implement while only an approximation of the true decoding threshold of an ensemble is produced.

For an irregular LDPC code ensemble with degree distribution pair $(\lambda(x), \rho(x))$, the evolution of the average extrinsic mutual information is obtained by recursively averaging over the different node degrees, that is

$$I_{E,V}^{(\ell)} = \sum_{d=1}^{d_v} \lambda_d I_{E,V}^{(\ell-1)}(d), \tag{98}$$

and

$$I_{E,C}^{(\ell)} = \sum_{d=1}^{d_c} \rho_d I_{E,C}^{(\ell-1)}(d), \tag{99}$$

where [207]

$$I_{E,V}^{(\ell)}(d) = J\left(\sqrt{(d-1)\left[J^{-1}\left(1-I_{E,C}^{(\ell-1)}\right)\right]^2 + \sigma_{\text{ch}}^2}\right), \tag{100}$$

and

$$I_{E,C}^{(\ell)}(d) = 1 - J\left(\sqrt{(d-1)\left[J^{-1}\left(1-I_{E,V}^{(\ell-1)}\right)\right]^2}\right), \tag{101}$$

Here, the variance of the LLR at the output of the channel $\sigma_{\text{ch}}^2 = 8RE_b/N_0$ for a rate-R code. The function $J(\cdot)$ is defined by [207]

$$J(\sigma) = 1 - \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{\left(\eta - \sigma^2/2\right)^2}{2\sigma^2}} \log_2\left(1 + e^{-\eta}\right) d\eta \tag{102}$$

with $I_{E,V}^{(0)} = J(\sigma_{\text{ch}}^2)$ is the initialization of the recursion.

4) PROTOGRAPH-BASED EXIT CHART

To identify good codes in the ensemble defined by $\lambda(x)$ and $\rho(x)$, an extension of the EXIT approach, namely the protograph-based EXIT (PEXIT) chart [208], has been proposed for LDPC ensembles defined by protographs, which takes into account the different edge connection properties. The PEXIT chart analysis is facilitated by the relatively small size of protographs and permits the analysis of protograph code ensembles characterized by the presence of multiple parallel edges, degree-1 VNs, and punctured VNs.

The PEXIT chart analysis eliminates the average in (98) and (99) and considers the propagation of the messages on specific edges that are specified by the protograph of the ensemble. Let $I_{E,V}(i,j)$ be the extrinsic mutual information between code bits associated with type-j VNs and the LLRs sent from these VNs to type-i CNs. Similarly, let $I_{E,C}(i,j)$ be the extrinsic mutual information between code bits associated with type-j VNs and the LLRs sent from type-i CNs to these VNs. Let $b_{i,j}$ be the element on the i-th row and the j-th element of the photograph base matrix. Then we have recursive formulas given by [208]

$$I_{E,V}^{(\ell)} = J\left(\sqrt{\sum_{s\neq i} b_{s,j}\left[J^{-1}\left(I_{E,C}^{(\ell-1)}(s,j)\right)\right]^2 + (b_{i,j}-1)\left[J^{-1}\left(I_{E,C}^{(\ell-1)}(i,j)\right)\right]^2 + \sigma_{\text{ch}}^2(j)}\right).$$

and

$$I_{E,C}^{(\ell)} = 1 - J\left(\sqrt{\sum_{s\neq j} b_{i,s}\left[J^{-1}\left(1 - I_{E,V}^{(\ell-1)}(i,s)\right)\right]^2 + (b_{i,j}-1)\left[J^{-1}\left(1 - I_{E,V}^{(\ell-1)}(i,j)\right)\right]^2}\right).$$

Note that if $b_{i,j} = 0$, $I_{E,V} = 0$ and $I_{E,C} = 0$ in VN and CN updates, respectively. The recursion continues until either the maximum iteration is reached or the convergence condition $I_{APP}(j) = 1$ for $j = 0, 1, \ldots, N-1$, where

$$I_{APP}^{(\ell)}(j) = J\left(\sqrt{\sum_{s} b_{s,j}\left[J^{-1}\left(I_{E,C}^{(\ell-1)}(i,j)\right)\right]^2 + \sigma_{\text{ch}}^2(j)}\right). \tag{103}$$

Hence, the threshold is the lowest value of E_b/N_0 for which all $I_{APP}(j)$ converge to 1.

5) RECIPROCAL CHANNEL APPROXIMATION

The decoding threshold of Protograph LDPC ensembles can be analyzed and designed via GA, EXIT chart or PEXIT chart. Alternatively, a simpler approach, namely the *reciprocal channel approximation* (RCA) [209], is devised so that the decoding threshold analysis for protograph LDPC ensembles can be carried out on the associated protograph directly. A single real-valued parameter, in this case, signal-to-noise ratio (SNR) $\mathcal{S}$, is used for full DE. For every value of $\mathcal{S}$, a reciprocal of SNR $\bar{\mathcal{S}}$ is defined such that $\mathcal{C}(\mathcal{S}) + \mathcal{C}(\bar{\mathcal{S}}) = 1$, where $\mathcal{C}(x)$ denotes the capacity of the BI-AWGN channel with SNR x. The parameters $\mathcal{S}$ and $\bar{\mathcal{S}}$ are interchangeable via $\mathcal{R}(x) = \mathcal{C}^{-1}(1 - \mathcal{C}(x))$ for $\mathcal{S} = \mathcal{R}(\bar{\mathcal{S}})$ and $\bar{\mathcal{S}} = \mathcal{R}(\mathcal{S})$.

To apply RCA on a protograph, select an initial channel SNR $\mathcal{S}_{\text{ch}}$ and identify all transmitted variable nodes first. The transformation $\mathcal{S}_{v\rightarrow c} = \mathcal{R}(\bar{\mathcal{S}}_{c\rightarrow v})$ and $\bar{\mathcal{S}}_{c\rightarrow v} = \mathcal{R}(\mathcal{S}_{v\leftarrow c})$ is applied and it refers to the message going out from a VN and CN, respectively. At VN and CN nodes, the extrinsic messages is determined as $\mathcal{S}_{v\rightarrow c} = \mathcal{S}_{\text{ch}} + \sum_{c'\in\mathcal{C}\backslash c} \bar{\mathcal{S}}_{c'\rightarrow v}$ and $\bar{\mathcal{S}}_{c\rightarrow v} = \sum_{v'\in\mathcal{V}\backslash v} \mathcal{S}_{v'\rightarrow c}$, respectively. The process continues and a threshold is determined by the smallest value of $\mathcal{S}_{\text{ch}}$ for which unbounded growth of all messages $\mathcal{S}_{v\rightarrow c}$ and $\bar{\mathcal{S}}_{c\rightarrow v}$ can be achieved.

6) FINITE BLOCKLENGTH SCALING LAW

In [210], the behavior of finite blocklength LDPC codes over the BEC in terms of the waterfall performance were studied. It was observed that if an iterative decoding process goes through a phase transition as a channel parameter ϵ crosses the decoding threshold ϵ^*, then around this transition point the decoding process obeys a very specific scaling law. Let $P_{\text{BLER}}(N, \epsilon)$ be the block error probability as a function of the blocklength N and the channel erasure probability ϵ. According to DE, $P_{\text{BLER}}(N, \epsilon)$ exhibits a phase transition at the iterative decoding threshold ϵ^* as $N \rightarrow \infty$. Then, the estimation of $P_{\text{BLER}}(N, \epsilon)$ for a finite length LDPC code is obtained by the method of the covariance evolution, also known as the *scaling law* [210]

$$P_{\text{BLER}}(N, \epsilon) = Q\left(\frac{\sqrt{N}(\epsilon^* - \epsilon)}{\alpha}\right), \tag{104}$$

where α is the *scaling parameter* that only depends on the degree distributions $\lambda(x)$ and $\rho(x)$, and $Q(\cdot)$ is the Q-function. The scaling behavior has been conjectured to more general settings and channels, and empirical evidence was shown to support the conjecture in [210].

C. FINITE-LENGTH CONSTRUCTION OF QC-LDPC CODES

Once an optimal degree distribution pair is obtained via any of the design tools mentioned above for an LDPC code ensemble, the next step is to construct an LDPC code from the ensemble such that the error rate performance approaches the theoretical threshold of the ensemble. The design of LDPC codes of finite lengths falls into two categories: *pseudorandom* code design and *structured* code design, where *graph-theoretical* based approaches and *algebraic-based or matrix-theoretical* based approaches are commonly adopted, respectively. The well-known graph-theoretical based construction methods are the *progressive edge-growth* (PEG) [211], [212] and the *protograph* methods [92]. The LDPC codes constructed using PEG algorithms are unstructured in the sense that they can only be described by specifying, for each VN, the indices of the CNs to which it is connected. From the practical implementation perspective, this is equivalent to storing the row and column indices of all the non-zero elements of a parity-check matrix in a memory unit. This significantly reduces the practicability of LDPC codes due to their high memory usage when implemented on high-speed hardware platforms such as field-programmable gate arrays (FPGA) or application-specific integrated circuits (ASIC). In addition, unstructured LDPC codes usually are encoded only via the multiplication of the information sequence by a generator matrix of the code, an operation whose complexity is quadratic with the codeword length.

Alternatively, algebraic-based design approaches, which were first introduced in 2000 [213], are commonly adopted in the design of structured LDPC codes. Since then, various algebraic methods for constructing LDPC codes, binary and non-binary, have been developed based on mathematical tools such as finite geometries, finite fields, difference sets and combinatorial designs, e.g., [207], [214], [215], [216], [217], [218], [219], [220], [221], [222], [223], [224], [225], [226], [227], [228], [229]. Moreover, the concepts of protograph can be further adopted in the algebraic-based design approaches, by limiting the permutation matrix to the cyclic shift of a $Z \times Z$ identity matrix. Various early-stage studies on designing protograph-based LDPC codes have been conducted, *e.g.,* [192], [193], [209], [230], [231], [232]. Many of the construction techniques have been adopted in the construction of quantum LDPC codes, *e.g.,* [233], [234], [235], [236]. Furthermore, the PEG algorithm may be effectively used to construct QC-LDPC codes by adding randomness to the code design, e.g., [237], [238], [239]. In practice, a QC-LDPC code is often designed using a hybrid of the two approaches so that the optimized pseudorandom construction performs well in the waterfall region, while code structure can help provide exemplary performance in the error floor region. In what follows, we focus on the construction of QC-LDPC codes.

The parity-check matrix of a QC-LDPC code is commonly in the form of *block-circulant*. Particularly, the parity-check matrix is an $m \times n$ array of $Z \times Z$ square *circulant permutation matrices* (CPMs) defined as

$$\boldsymbol{P} := \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & \cdots & 0 & 0 \end{bmatrix}. \tag{105}$$

Note that $\boldsymbol{P}^i$ represents the i-th cyclic right shift of $\boldsymbol{P}$ for $0 \leqslant i \leqslant Z-1$ and $\boldsymbol{P}^Z = \boldsymbol{P}^0 = \boldsymbol{I}_Z$ is the identity matrix of size Z. Then the parity-check matrix of an $mZ \times nZ$ QC-LDPC code has the structure of

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{P}^{b_{0,0}} & \boldsymbol{P}^{b_{0,1}} & ,\ldots & \boldsymbol{P}^{b_{0,n-1}} \\ \boldsymbol{P}^{b_{1,0}} & \boldsymbol{P}^{b_{1,1}} & ,\ldots & \boldsymbol{P}^{b_{1,n-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{P}^{b_{m-1,0}} & \boldsymbol{P}^{b_{m-1,1}} & ,\ldots & \boldsymbol{P}^{b_{m-1,n-1}} \end{bmatrix}, \tag{106}$$

where $b_{i,j} < Z$ or $b_{i,j} = -1$ if the circulant is a zero-matrix for $0 \leqslant i \leqslant m-1$ and $0 \leqslant j \leqslant n-1$. Such a parity-check matrix may be compactly described by means of its exponent matrix, also known as the *base matrix*,

$$\boldsymbol{H}_b = \begin{bmatrix} b_{0,0} & b_{0,1} & ,\ldots & b_{0,n-1} \\ b_{1,0} & b_{1,1} & ,\ldots & b_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m-1,0} & b_{m-1,1} & ,\ldots & b_{m-1,n-1} \end{bmatrix}. \tag{107}$$

The base matrix not only constitutes a compact description of the parity-check matrix $\boldsymbol{H}$, which yields efficient memory usage to describe the Tanner graph of the QC-LDPC code but also affects the performance of the code by maximizing the girth of the QC-LDPC constructed. For instance, in [222], [240] a necessary and sufficient condition was derived for the Tanner graph to be characterized by a girth not smaller than some value. Hence, constructing a good QC-LDPC code is equivalent to finding a good base matrix with optimized shift values for a given lifting size.

1) QC-LDPC CODES BASED ON PROTOGRAPHS

Unlike the base matrix $\boldsymbol{H}_b$ of QC-LDPC codes where each element represents a cyclic shift of the identity matrix, the protomatrix $\boldsymbol{B}_p$ is a constitute collection of edge types. In this case, a QC-LDPC code can be constructed from its promatrix $\boldsymbol{B}_p$ via a two-step lifting process. Recall that a non-binary entry of $\boldsymbol{B}_p$ represents the multiple edge connections between the same VN and CN types. Thus, the first step is to break this multi-edge type connection. This can be done by replacing element $b_{i,j}$, of $\boldsymbol{B}_p$, $0 \leqslant i \leqslant m_p - 1$ and $0 \leqslant j \leqslant n_p - 1$, by a $Z' \times Z'$ permutation matrix such that the summation of each row and column of each permutation equals to $b_{i,j}$. The value of Z' needs to be greater or equal to the maximum value in the $\boldsymbol{B}_p$ so that the resulting matrix $\boldsymbol{B}'_p$ is a $m_pZ' \times n_pZ'$ protomatrix contains elements of 1s and 0s only. Note that the permutation matrix in the first step of the lifting process does not need to be a CPM. An effective way of placing the 1s' inside each permutation matrix is to use PEG algorithms to add randomness to the code structure [239]. The second step is to obtain the

$Zm_pZ' \times Zn_pZ'$ parity-check matrix of the derived graph by replacing each 1 in $\boldsymbol{B}'_p$ with a $Z \times Z$ CPM with a cyclic shift, and each 0 by a $Z \times Z$ zero matrix. The cyclic shifts can be obtained either by algebraic design approaches or search-based optimization approaches.

2) QC-LDPC CODES FROM FINITE GEOMETRY

The geometric approach to constructing LDPC codes is based on lines and points of finite geometry. Well-known finite geometries are Euclidean and projective geometry over finite fields [216]. It is known that finite geometry LDPC codes have relatively good minimum distances and their Tanner graphs do not contain cycles of length 4. The LDPC codes constructed using finite geometries yield either cyclic or quasi-cyclic depending on the dimension of the Euclidean geometry and therefore possess a simple encoding structure.

Let $EG(m, 2^s)$ be an m-dimensional Euclidean geometry over the Galois Field $GF(2^s)$ where m and s are positive integers. Denote by $\boldsymbol{H}_{EG(m,2^s)}$ an $M \times N$ parity-check matrix over $GF(2)$ composed of $N = (2^{(m-1)s}-1)(2^{ms}-1)/(2^s-1)$ lines in $EG(m, 2^s)$ that pass through $M = 2^{ms} - 1$ non-origin points [216]. If $m > 2$, the N columns of $\boldsymbol{H}_{EG(m,2^s)}$ can be partitioned into $\mathcal{K} = (2^{(m-1)s} - 1)/(2^s - 1)$ cycle classes, and is denoted as $\boldsymbol{h^i}_{(m,2^s)}$, $i = 1, 2, \ldots, \mathcal{K}$. Each of these $\mathcal{K}$ cycle classes consists of $(2^{ms} - 1)$ lines, which are obtained by cyclically shifting (downwards) any line in the class $2^{ms} - 1$ times. Hence,

$$\boldsymbol{H}_{EG(m,2^s)} = \left[\boldsymbol{h^1}_{(m,2^s)}, \boldsymbol{h^2}_{(m,2^s)}, \ldots, \boldsymbol{h^{\mathcal{K}}}_{(m,2^s)}\right].$$

Alternatively, if $m = 2$, $\boldsymbol{H}_{EG(m,2^s)}$ is a square matrix with $\mathcal{K} = 1$. Let $\boldsymbol{g}^i = [g_0^i, g_1^i, \ldots, g_{2^{ms}-2}^i]$ be the first column of a cycle class $\boldsymbol{h}^i_{(m,2^s)}$ for $0 \leqslant i, \leqslant \mathcal{K}$, and $\mathcal{S}_{\boldsymbol{g}^i}$ be the set of indices of the non-zero elements. Note that $\boldsymbol{g}^i$ is also known as the *generator* of the cycle class and the number of non-zero elements of $\boldsymbol{g}^i$ is $|\mathcal{S}_{\boldsymbol{g}^i}| = 2^s$. Then

$$\boldsymbol{h}^i_{(m,2^s)} = \sum_{j}^{2^s} \boldsymbol{P}^{q_j}, \forall q_j \in \mathcal{S}_{\boldsymbol{g}^i}$$

is a weight-2^s circulant block and $\boldsymbol{H}_{EG(m,2^s)}$ is a $1 \times \mathcal{K}$ array of weight-2^s circulant blocks.

One approach of constructing a QC-LDPC code of the form (106) from $\boldsymbol{H}_{EG(m,2^s)}$ is to perform generator splitting [207]. An example of such a method is the $(d_v, d_c) = (4, 32)$ regular $(8176, 7156)$ QC-LDPC code designed from $EG(3, 2^3)$ Euclidean geometry. The code has rate $0.8752 > 1 - d_v/d_c$ because the resulting parity-check matrix $\boldsymbol{H}_{(8176,7156)}$ is not in full rank. The geometry $EG(3, 2^3)$ has total of $M = 2^{ms} - 1 = 511$ non-origin points and $N = (2^{(m-1)s} - 1)(2^{ms} - 1)/2^s - 1 = 4599$ lines, and $\mathcal{K} = 9$ cycle classes, each cycle class is a weight $2^s = 8$ circulant block. The parity-check matrix $\boldsymbol{H}_{(8176,7156)}$ is obtained by splitting the generator $\boldsymbol{g}^i$ of $\mathcal{K} = 8$ cycle classes into smaller subsets $\{\boldsymbol{g}^{i,1}, \boldsymbol{g}^{i,2}, \boldsymbol{g}^{i,3}, \boldsymbol{g}^{i,4}\}$. Each $\boldsymbol{g}^{i,l}, l = 1, 2, 3, 4,$ has 2 elements. The resulting parity-check matrix $\boldsymbol{H}_{(8176,7156)}$ is given by

$$\boldsymbol{H}_{(8176,7156)} = \left[\boldsymbol{h}^1_{(3,2^3)}, \boldsymbol{h}^2_{(3,2^3)}, \ldots, \boldsymbol{h}^8_{(3,2^3)}\right],$$

where each

$$\boldsymbol{h}^i_{(3,2^3)} = \begin{bmatrix} \boldsymbol{h}^{i,1} & \boldsymbol{h}^{i,2} \\ \boldsymbol{h}^{i,3} & \boldsymbol{h}^{i,4} \end{bmatrix}$$

is a 2×2 array of weight-2 circulant blocks, where $\boldsymbol{h}^{i,l} = \boldsymbol{P}^{q_1} + \boldsymbol{P}^{q_2}$, $l = 1, 2, 3, 4$ for $q_1, q_2 \in \boldsymbol{g}^{i,l}$. The Tanner graph of the code has a girth of 6, and at a bit error rate of 10^{-6} the code performs 1dB from the Shannon limit under sum-product decoding with maximum 50 iterations [207]. This code has been selected for use in the NASA Consultative Committee for Space Data Systems (CCSDS) telemetry system [241].

3) QC-LDPC CODES BASED ON FINITE FIELDS

In the early 1960s, finite fields were successfully used to construct linear block codes, especially cyclic codes, with large minimum distances for hard-decision algebraic decoding, such as BCH codes [242], [243] and Reed-Solomon (RS) codes [244]. In the past decades, there have been major developments in using finite fields to construct LDPC codes. LDPC code constructions based on finite fields perform well over the binary-input AWGN channel with iterative decoding based on belief propagation. Most importantly, these finite-field LDPC codes were shown to have low error floors. These codes are more suitable for wireless and optical communication systems and data-storage systems, for which very low bit and/or word error rates are required. Furthermore, most of the LDPC code construction/based on the basis of finite fields are quasi-cyclic and hence they can be efficiently encoded using simple shift registers with linear complexity. The general construction of QC-LDPC codes using finite fields is that the elements of the base matrix $\boldsymbol{H}_b$ belong to the elements of a finite field of a certain size. The position of each element can be determined through different properties of finite fields, such as additive and multiplicative group properties, subgroups, primitive elements of prime fields, and extension fields.

D. SPATIALLY-COUPLED/CONVOLUTIONAL LDPC CODES

In recent years, spatially-coupled LDPC (SC-LDPC) codes have drawn a lot of attention to the research communities and industries. SC-LDPC codes can be viewed as a type of convolutional LDPC codes [245], [246] that have the ability to combine good features of regular and irregular LDPC block codes in a single design [247]. It has been proven, in [159], [248], [249], that SC-LDPC ensembles are capacity-achieving on binary-input memoryless output-symmetric (BMS) channels under BP decoding. To construct SC-LDPC codes from LDPC block codes, the well-known approach called *matrix unwrapping* [246] is commonly adopted. Good convolutional LDPC codes can be constructed from good LDPC codes, e.g., see [250] for guidelines. Consider an $M \times N$ parity-check matrix $\boldsymbol{H}$ of an LDPC block code. Its design code rate is given by $R_{BC} = 1 - M/N$. In practice, SC-LDPC codes are *terminated*, whose parity-check matrix can be represented by

$$\boldsymbol{H}^{SC} = \begin{bmatrix} \overbrace{\begin{matrix} \boldsymbol{H}_0 & & & \\ \boldsymbol{H}_1 & \boldsymbol{H}_0 & & \\ \vdots & \boldsymbol{H}_1 & & \\ \boldsymbol{H}_{m_s-1} & \vdots & \ddots & \\ \boldsymbol{H}_{m_s} & \boldsymbol{H}_{m_s-1} & \ddots & \boldsymbol{H}_0 \\ & \boldsymbol{H}_{m_s} & \ddots & \boldsymbol{H}_1 \\ & & & \vdots \\ & & & \boldsymbol{H}_{m_s-1} \\ & & & \boldsymbol{H}_{m_s} \end{matrix}}^{L} \end{bmatrix}, \tag{108}$$

where m_s is often called the *syndrome former memory* or coupling memory. Each $\boldsymbol{H}_j$ of size $M \times N$, such that $\sum_{j=0}^{m_s} \boldsymbol{H}_j = \boldsymbol{H}$, is a *descendent matrix* of the parity-check matrix $\boldsymbol{H}$. The set of descendent matrices is then repeated L times as shown in (108) to construct the parity-check matrix $\boldsymbol{H}^{SC}$ of the terminated SC-LDPC code, where L is the *termination length* or coupling length of the code. Note that the process of termination results in a parity-check matrix $\boldsymbol{H}^{SC}$ that contains irregular row weights. The code rate of an SC-LDPC code is then a function of L, given by $R_{SC} = 1 - (L + m_s)M/LN$. It is obvious that if the termination length $L \to \infty$, the SC LDPC code has the same code rate as the underlying LDPC block code defined by $\boldsymbol{H}$, that is $R_{SC} \xrightarrow{L\to\infty} R_{BC}$. Different from conventional block codes, SC-LDPC codes possess some important properties over their uncoupled counterparts, including simple code construction approaches and sliding window decoding with high throughput.

1) Threshold Saturation: The *threshold saturation* phenomenon discovered in [159] shows that the decoding threshold of regular SC-LDPC codes under sub-optimal BP decoding can reach that under the optimal maximum a posterior (MAP) decoding on the BEC. Intuitively, this is the result of the termination, which introduces a slight structured irregularity in the graph. The CNs with lower degrees at each end of the terminated graph pass more reliable messages to their neighboring VNs. This effect propagates throughout the graph as decoding iterations increase. On top of that, it is also shown that the MAP decoding threshold of regular-(d_v, d_c, L, m_s) SC-LDPC codes is the same as that of regular (d_v, d_c)-LDPC block codes, when $m_s \to \infty$ and $L \to \infty$. In addition, for regular SC-LDPC codes, the terminated ensembles retain asymptotically good in the sense that their minimum distance grows linearly with blocklength N. Moreover, BP decoding thresholds of regular SC-LDPC codes are extremely close to the BEC capacity, with gaps that diminished for larger (d_v, d_r), in contrast with what happens for regular block LDPC code ensembles. On the other hand, the MAP decoding thresholds of regular-(d_v, d_c) LDPC block code ensembles achieve capacity as (d_v, d_c) become very large. Thanks to threshold saturation, one can simply increase (d_v, d_c) to construct capacity-approaching SC-LDPC codes without tedious optimization steps as in the design of irregular LDPC block codes. The proof of threshold saturation of SC-LDPC codes was generalized to the general BMS channel [249]. A simplified proof was later given in [162], see Section V-K1.

2) Universality: In [249], the universality of SC-LDPC code is investigated and proved that a single SC-LDPC ensemble is universally good for all BMS channels, without the need to customize the ensemble construction to a specific channel. It should be noted that this is not the case for many other types of codes such as irregular LDPC block codes, turbo codes, and the original polar codes. The universality of SC-LDPC codes was extended to other channel settings, e.g., Gaussian multiple access channel [251].

3) Sliding Window Decoding: A terminated SC-LDPC code can be decoded in the same way as decoding a block code. Alternatively, due to the diagonal structure of the parity-check matrix, SC-LDPC codes can be decoded using a much smaller window size, and slide across the matrix diagonally. This type of decoding is known as the sliding window decoder [252] and has the feature that the decoding process may start during the reception of the data frame, and hence, an advantage for streaming type of data transmission, e.g., [246], [252], [253], [254].

Let W be a window of size covering WM CNs and WN VNs. The decoding window slides from time index $t = 0$ to time index $t = L-1$, which associates with different window positions in $\boldsymbol{H}^{SC}$ as shown in Fig. 37. At each time index, an iterative message-passing decoding is performed within the window in the same manner as decoding a block code. The decoding process stops if the syndrome check of the decoding window is satisfied or a predetermined maximum number of iterations is reached. Then the decoding window shifts by M CNs vertically and N VNs horizontally to the next time index t. The first N VNs shifted out of the decoding window are called target symbols.

The performance of window decoding degrades as W becomes smaller. This is because, in a decoding window, there exist VNs that have neighboring CNs outside the decoding window. Thus, the messages sent out from these VNs may not be reliable. These unreliable messages are propagated to the next window and deteriorate the error rate performance of the code. Various research works have been conducted to improve the performance of sliding window decoding for SC-LDPC codes, e.g., [246], [252], [253], [254], [255], [256], [257], [258].

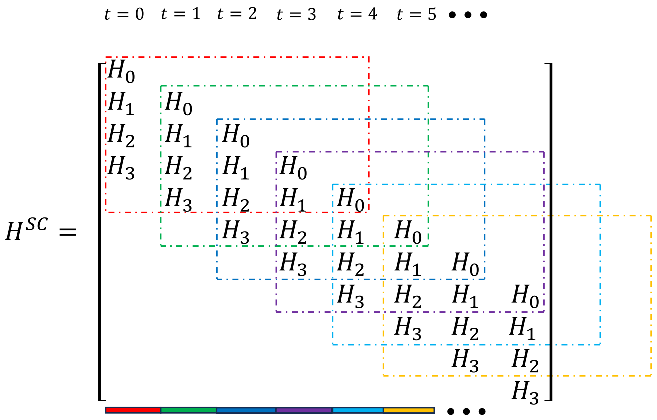


FIGURE 37. An example of sliding window decoder with window size $W = 5$ at time index $t = 0, 1, 2, 3, 4, 5$ (dash regions). The output of each decoding window is the target symbols shown at the bottom of the matrix.

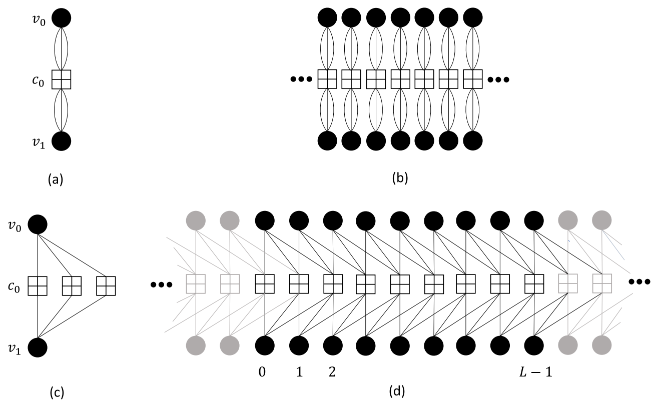


FIGURE 38. (a) Protograph representation of a $(3, 6)$ regular LDPC block code ensemble. (b) sequence of $(3, 6)$ regular LDPC block code ensembles. (c) edge spreading for one $(3, 6)$ regular LDPC block code ensemble with $m_s = 2$.
(d) protograph representation of a terminated SC-LDPC code ensemble with coupling length L and coupling memory $m_s = 2$.

1) PROTOGRAPH SC-LDPC CODES

An insightful way of designing terminated SC-LDPC codes is to use a *protograph* representation of a code ensemble. Let $\boldsymbol{B}_p$ be the $m_p \times n_p$ protomatrix of a protograph LDPC block code ensemble. Then the protograph of a (m_p, n_p, L) ensemble of SC-LDPC codes is obtained by performing *edge-spreading* [180], [259] to split the base matrix $\boldsymbol{B}_p$ into (m_s+1) descendent protomatrices $\boldsymbol{B}_p^{(0)}, \boldsymbol{B}_p^{(1)}, \ldots, \boldsymbol{B}_p^{(m_s)}$. Each of the descendent protomatrices has size $m_p \times n_p$ and $\sum_{i=0}^{m_s} \boldsymbol{B}_p^{(i)} = \boldsymbol{B}_p$. If $\boldsymbol{B}_p^{(i)}$ are identical for $0 \leqslant i \leqslant m_s$, by arranging the set of descendent protomatrices $\boldsymbol{B}_p^{(0)}, \boldsymbol{B}_p^{(1)}, \ldots, \boldsymbol{B}_p^{(m_s)}$ into the form as shown in (108), the resulting terminated protomatrix of an SC-LDPC code is *time-invariant*. Otherwise, it is *time-varying*, where each row of (108) could start with a different $\boldsymbol{B}_p^{(i)}$. The design code rate of the protograph SC-LDPC codes is given by $R_{SC} = 1 - (L + m_s)m_p/Ln_p$.

To construct a practical code from a protograph ensemble, the process of lifting is adopted to derive a large Tanner graph from the protograph of the ensemble. By lifting each element inside $\boldsymbol{B}_p^{(i)}$ by the $Z \times Z$ zero matrix or CPM defined in (105). The derived Tanner graph is a QC protograph SC-LDPC code that has $(m_s + L)Zm_p$ check nodes and LZn_p variable nodes.

Consider the example of $(d_v, d_c) = (3, 6)$ regular protograph LDPC block code ensemble with protomatrix $\boldsymbol{B}_p = [3, 3]$. Let $\boldsymbol{B}_p^{(0)} = \boldsymbol{B}_p^{(1)} = \boldsymbol{B}_p^{(2)} = [1, 1]$. Thus, $m_s = 2$ and $\sum_{i=0}^{m_s} \boldsymbol{B}_p^{(i)} = \boldsymbol{B}_p$. The protomatrix of the corresponding SC-LDPC codes ensemble is

$$\boldsymbol{B}_p^{SC} = \begin{bmatrix} [1\ 1] & & & & \\ [1\ 1] & [1\ 1] & & & \\ [1\ 1] & [1\ 1] & [1\ 1] & & \\ & [1\ 1] & [1\ 1] & [1\ 1] & \\ & & [1\ 1] & [1\ 1] & [1\ 1] \\ & & & [1\ 1] & [1\ 1] \\ & & & & [1\ 1] \end{bmatrix} \tag{109}$$

for $L = 5$. The design rate $R_{SC} = 1 - (5 + 2)/5 \times 2 = 1-7/10 < 1/2$. To construct the parity-check matrix of a QC SC-LDPC, each entry of $\boldsymbol{B}_p^{SC}$ is then replaced with a cyclic shift optimally design, followed by the lifting process with a lifting factor Z. Fig. 38 illustrated the construction of the SC-LDPC code ensemble from the $(3, 6)$ regular protograph LDPC block code ensemble.

The connected chain SC-LDPC codes were introduced in [260], which extends the spatial coupling phenomenon of individual graphs of LDPC block codes in a chained connection of multiple SC-LDPC graphs. By adding additional edges to connect the terminated CN of one sub-chain to the VNs in another sub-chain, reliable information propagates from several directions rather than just from the ends of a single chain, leading to an improved BP decoding threshold when the coupling length L is small. The loop construction of connected chain SC-LDPC codes was proposed in [261] with further improved BP decoding threshold over the BEC. In addition, the work in [262] not only improved the decoding threshold of connected chain SC-LDPC codes by introducing self-connected SC-LDPC code ensembles but also proposed a termination method to reduce the rate loss due to small L. As an example, the designed code rates versus the BEC thresholds of the regular-$(3, 6, L, 2)$ SC-LDPC codes in [180], [260], and [262], denoted by $\mathcal{C}_0$, $\mathcal{L}$, and $\mathcal{M}_1$, respectively, under different chain lengths are shown in Fig. 39. The self-connected chain SC-LDPC code ensembles proposed in [262] achieve the best trade-off between rate loss and gap to capacity.

2) GLOBALLY COUPLED LDPC CODES

The globally-coupled LDPC (GC-LDPC) [263] codes are another class of coupled LDPC codes. Unlike spatially coupled Tanner graphs, where $m_s + 1$ Tanner graphs are coupled by the common check node, the Tanner graph of GC-LDPC codes has L disjoint LDPC block codes coupled by the additional globally-connecting check nodes. As a result, the code length can be scaled up, which

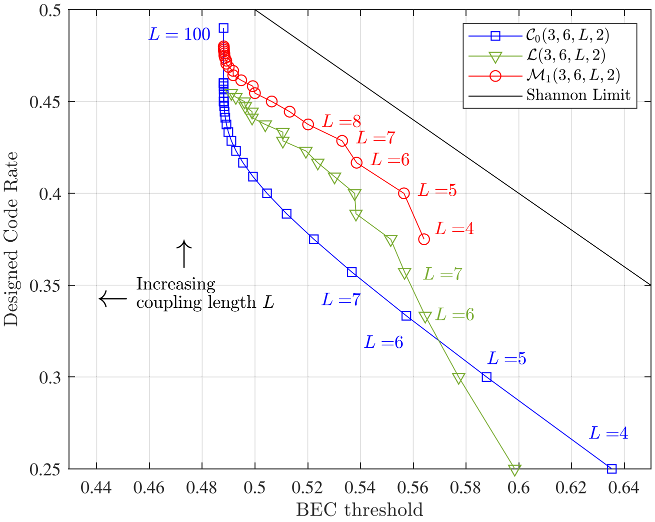


FIGURE 39. Design rate versus BP threshold on the BEC for various coupled ensembles under different chain lengths.

implies promising performance improvement, based on the component LDPC block codes, avoiding construction of a completely new longer LDPC code. An example of the parity-check matrix of a GC-LDPC code is given as

$$\boldsymbol{H}^{GC} = \left[\begin{array}{ccccc} \boldsymbol{H}_0 & & & & \\ & \boldsymbol{H}_1 & & & \\ & & \boldsymbol{H}_2 & & \\ & & & \ddots & \\ & & & & \boldsymbol{H}_{t-1} \\ \hline & & \boldsymbol{H}_{gc} & & \end{array}\right], \quad (110)$$

where the upper submatrix of $\boldsymbol{H}^{GC}$ is a $t \times t$ diagonal array consisting of t LDPC block codes $\boldsymbol{H}_i$, $0 \leqslant i \leqslant t-1$, of size $M \times N$ on the main diagonal. The upper submatrix is known as the *local part* as each Tanner graph of the corresponding LDPC block is disjoint and independent. The lower part, also known as the *global part*, is an $s \times tN$ matrix. If each $M \times N$ submatrix $\boldsymbol{H}_i$ is replaced by a $m_p \times n_p$ protomatrix, the $(m_p t + s) \times n_p t$ protomatrix $\boldsymbol{B}_p^{GC}$ of a GC-LDPC code is obtained. Different constructions of the GC-LDPC codes have been investigated. For instance, the GC-LDPC codes construction based on finite field and finite geometry [263], [264]. A family of non-binary GC-LDPC codes designed from RS codes is proposed in [265] In [266], the construction of rate-compatible GC-LDPC codes is investigated.

To effectively decode GC-LDPC codes, a two-phase local/global iterative decoding scheme for CN-GC-LDPC codes is proposed in [263]. Taking advantage of the cascading structure of the local part, a whole data frame can be split into t independent sections, each section can be simultaneously decoded by an independent decoder. If all sections of the local part are successfully decoded and the locally decoded codeword satisfies the parity-check constraints in the global part, the locally decoded codeword will be delivered to the user. If it does not, the global decoder starts to process the received codeword from the local decoder. To reduce the decoding latency, sliding window decoders are commonly adopted in the BP decoding of SC-LDPC code. The investigation of sliding window decoders for GC-LDPC codes is still ongoing.

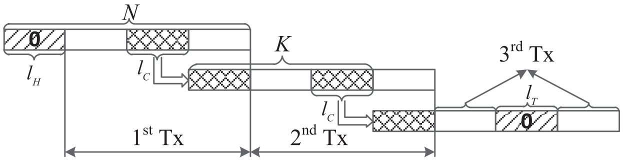


FIGURE 40. The block diagram of encoding a TB with three CBs.

Due to that the rate loss caused by the additional global parity checks cannot be neglected for the finite coupling length L, free-ride coding [267] is proposed to construct coupled LDPC codes without any rate loss. The basic idea is to transmit some extra bits over the original coded link in a superposition (XOR) manner, where the coded length remains unchanged in comparison with the original coded link. At the receiver side, a successive cancellation decoder can be used. It is shown in [267] that the proposed GC-LDPC codes can improve the performance of the component LDPC codes, yielding an extra coding gain of up to 0.8 dB, but without any code rate reduction.

3) PARTIALLY INFORMATION COUPLED LDPC CODES

In the current 5G networks, the effective user data rate is approximately increased by 100 folds compared to 4G, and the maximum transport block (TB) size is over 1.2 million bits. For peak throughput scenarios, the highest number of code blocks (CBs) in a TB reaches 151, and hence, the TB-level HARQ protocol will need more spectrum resources to send feedback information to the transmitter. To improve the spectrum efficiency and the error rate performance of the transport block (TB), partially information-coupled LDPC (PIC-LDPC) codes were proposed [268]. The idea is to share a few information bits between every two adjacent CBs during the encoding. In addition, by adding dummy bits to the first and the last CBs, there is a considerable and consistent SNR gain since the reliable messages from these two CBs spread out across other CBs with the aid of the coupled information bits when performing iterative decoding.

The frame structure of the PIC-LDPC codes with three CBs is illustrated in Fig. 40. The CBs are component codewords of a systematic (N, K) LDPC code of rate $R = K/N$. To obtain the CB at time instant t, the LDPC encoder takes l_c information bits from the information bits at time $t-1$ and $K - l_c$ information bits at time t as its input. The encoder outputs a length-N LDPC component codeword. In other words, the CB at time t shares l_c information bits with the CB at time $t-1$. These shared information bits are called coupled bits and are only transmitted once, such that those l_c bits are punctured from the t-th CB. For the first CB, l_H

information bits are set to zero for initialization. Moreover, l_T information bits in the last CB are set to zero for terminating the coupled code chain. As the coupled bits and the zero bits are not transmitted, the length of the transmitted TB is $N_{TB} = (N - l_H) + (N - l_c) + (N - l_c - l_T)$ and the total information bits in the transmitted TB is $K_{TB} = (K - l_H) + (K - l_c) + (K - l_c - l_T)$. Let $l_H = l_T = l_c = l$. For coupling length L, the code rate of PIC-LDPC codes is

$$R_{\text{PIC}} = \frac{K_{TB}}{N_{TB}} = \frac{L(K-l)-l}{L(N-l)-l}.$$

When $L \to \infty$, the code rate becomes $R_{\text{PIC}} \to \frac{K-l}{N-l}$.

The PIC is performed on the encoder of LDPC codes, which is different from SC-LDPC codes whose coupling is performed based on the parity-check matrix. As a result, PIC-LDPC codes can directly employ conventional encoders and decoders for LDPC block codes as their component code encoding and decoding. The decoding of PIC-LDPC codes can be done at the TB level using the feed-forward and feed-backward (FF-FB) window decoding [268]. The FF-FB decoding exploits the correlation, as well as the coupled bits, between every two consecutive CBs to achieve a good TB error rate. Simulation results show that the PIC-LDPC codes yield at least 0.5 dB gain over the LDPC block codes counterpart from IEEE 802.16e.

E. LDPC CODES IN COMMUNICATIONS STANDARDS

1) IEEE 802 COMMUNICATION STANDARDS

1) 802.3an: The IEEE 802.3an [269] standard adds a physical layer for 10 Gigabit Ethernet over unshielded twisted pair cabling (10GBASE-T) for distances of up to 100 meters. The LDPC code is not quasi-cyclic and is designed from the $[N, K, d_{min}] = [32, 2, 31]$ shortened RS base code over $GF(2^6)$, where d_{min} denotes the minimum distance of the code. The resulting LDPC code is a rate 0.84 binary $[N, K] = [2048, 1723]$ code that guarantees no cycles of length four within the Tanner graph. The matrix has a constant column weight of 6 and a constant row weight of 32. Thus, the LDPC code is a $(6, 32)$-regular LDPC code. Detail of the code designs is described by Djurdjevic et al. in [219].

2) 802.11n Wireless LAN: IEEE 802.11n [270] is an amendment to the previous 802.11 a/b/g standards in the 2.4GHz and 5GHz bands. The amendment adds a high throughput physical layer specification that encodes data fields using either a convolutional code with a memory length of 7 or a QC-LDPC code. There are in total of 12 independent QC-LDPC codes with 3 code lengths and 4 code rates. Each of the 12 codes is derived from a protograph with one of the lifting sizes $27, 54,$ and 81, which corresponds to the three blocklengths $N = 648, 1296$ and 1944. For rates $1/2, 2/3, 3/4,$ and $5/6$, the size of the base matrix is 12×24, 8×24, 6×24, and 4×24, respectively. The exponent matrix [270] for each code is optimized with respect to the error rate performance and the girth property. The parameters of the codes are summarized in Table 15. The parity-check matrix of the code has a dual-diagonal structure. Such a code structure enables efficient encoding [271] to be performed on the parity-check matrix, and hence, no generator matrix is needed.

TABLE 15. IEEE 802.11n LDPC code parameters

Code length N	Lifting size Z	Code rates R
648	27	1/2, 2/3, 3/4, 5/6
1296	54	1/2, 2/3, 3/4, 5/6
1944	81	1/2, 2/3, 3/4, 5/6

3) 802.11ad Wireless LAN at 60GHz: IEEE 802.11ad [272] extend the previous wireless LAN standards into the 60 GHz band, i.e., millimeter wavelength. This version contains a directional multigigabit physical layer specification utilizing four QC-LDPC codes of rates $1/2$, $5/8$, $3/4$, and $13/16$ to send control and data. The base matrix is of size 8×16, 6×16, 4×16, and 3×16, respectively, corresponding to each one of the code rates. In all cases, the length of the codes is 672 and the lifting size is 42. The code has the structure of a lower triangular form, which is different from the one designed for 802.11n. LDPC codes with a lower triangular code structure also yield efficient encoding using the parity-check matrix [273].

4) 802.16e Mobile WiMAX: Mobile WiMAX (Worldwide Interoperability for Microwave Access), IEEE 802.16e [274], was one of the first standards to adopt LDPC codes for forward error correction. The standard added mobility to the metropolitan area network standards. For rates $1/2$, $2/3$,[2] $3/4$, and $5/6$, the base matrix is of size 12×24, 8×24, 6×24 and 4×24, respectively. Moreover, the code lengths vary from 576 to 2304 with a step size of 96 bits, and the corresponding lifting sizes vary from 24 to 96 with a step size of 4.

5) 802.22 Cognitive Wireless: The IEEE 802.22 [275] standard is for cognitive wireless regional area networks that operate in TV bands between 54MHz and 862MHz. The aim of the standard is to bring broadband access to low-population-density areas. The maximum data rate is about 20 Mbit/s. Due to its cognitive radio techniques, it has the potential to be applied in many regions worldwide. The standard adopted the same LDPC codes as in 802.16e with minor modifications. Two shorter code lengths, 384 and 480, were added, and only one base matrix is used for rates $2/3$ and $3/4$.

6) 802.15.3c Millimeter WPAN: IEEE 802.15.3c [270] is a standard for high data rate wireless personal area networks (WPAN). The standard adds a new mmWave PHY layer that operates in the 60 GHz band (57 - 64 GHz) and allows for air throughput of up to 5 Gbit/s. The standard defines 5 LDPC codes with two blocklengths. For rates $1/2$, $5/8$, $3/4$, and $7/8$, the base matrix is of size 16×32, 12×32, 8×32,

[2] Two different base matrix as were designed for rates $2/3$ and $3/4$.

TABLE 16. Submatrix size for nine CCSDS QC-LDPC codes.

	Lifting size Z		
Information length K	rate $1/2$	rate $2/3$	rate $4/5$
1024	512	256	128
4096	2048	1024	512
16384	8192	4096	2048

and 4×32, respectively. Moreover, for these code rates, the lifting size is 21, which yields codes with a length of 672. For a rate of $14/15$, the base matrix is of size 1×15 with a lifting size of 96. The resulting code length is 1440. The code structure for this code is neither a dual-diagonal nor a lower triangular form. Instead, the code structure can be considered as an approximate lower triangular form, which is able to perform efficient encoding in a similar way as for codes in a lower triangular form with an encoding complexity 'gap'. Such a gap is defined as the distance of the given parity-check matrix to a lower triangular matrix [273].

2) CONSULTATIVE COMMITTEE FOR SPACE DATA SYSTEMS (CCSDS)

The CCSDS has included LDPC codes in its recommendation for near-Earth and deep-space telemetry [241]. A total of ten QC-LDPC codes have been included in the recommendation. An $[8160, 7136]$ code with a rate $R = 223/255$ is recommended for near-Earth telemetry applications. The rest nine codes, whose information lengths and code rates are the combination of $K = 1024, 4096, 16384$ and $R = 1/2, 2/3, 4/5$, are recommended for deep-space telemetry.

The $[8160, 7136]$ LDPC code is constructed from the $[8176, 7156]$ QC-LDPC code via shortening and extension. The parity-check matrix of the $[8176, 7156]$ QC-LDPC is a 2×16 array of 511×511 circulant blocks constructed based on Euclidean geometry. Moreover, for rates $1/2, 2/3, 4/5$, the nine QC-LDPCs for deep-space telemetry are represented by a 3×5, 3×7, or 3×11 array of $Z \times Z$ circulant blocks, respectively, where $Z = 2^b, b = 7, 8, \dots, 13$. The parameters of the nine QC-LDPC codes are summarized in Table 16.

The nine QC-LDPC codes for deep-space telemetry are constructed by lifting the AR4JA LDPC code family [200]. Note that for the AR4JA protograph LDPC code family, one variable node with the highest degree is punctured. This refers to the last column of the protomatrix. The lifting procedure is performed in two stages. The first stage uses an expansion factor equal to 4 and the second expansion factor Z' equal to the power of 2 leading to $Z = 4Z'$. In each stage, the lifting procedure employs CPM whose exponents are selected based on the extrinsic message degree (EMD) or approximate cycle EMD (ACE) metrics [276]. The protomatrix for the rate $1/2$ protograph code is given by

$$\boldsymbol{H}_{\frac{1}{2}} = \begin{bmatrix} 0_Z & 0_Z & I_Z & 0_Z & I_Z \oplus \prod_1 \\ I_Z & I_Z & 0_Z & I_Z & \prod_2 \oplus \prod_3 \oplus \prod_4 \\ I_Z & \prod_5 \oplus \prod_6 & 0_Z & \prod_7 \oplus \prod_8 & I_Z \end{bmatrix},$$

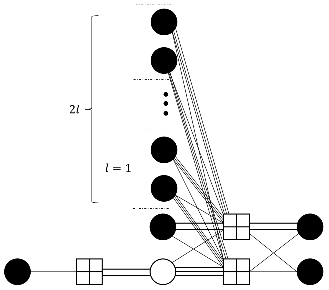


FIGURE 41. Protograph for AR4JA code family.

where $n_p = 5$, $m_p = 3$. The optimized permutation matrices $\prod_i$, $i = 1, 2, \dots, 8$, are in [241]. Since the last column is punctured during the transmission, the code rate $R = (n_p - m_p)/(n_p - \hat{n}_{pu}) = 2/4 = 1/2$. By adding two columns at a time to $\boldsymbol{H}_{\frac{1}{2}}$, the AR4JA protograph LDPC code family, shown in Fig. 41, is obtained with the code rate defined as

$$R_{\frac{1+l}{2+l}} = \frac{2l + (n_p - 3)}{2l + (n_p - 1)}, \tag{111}$$

where $l = 0, 1, 2, 3, \dots,$ is an integer meaning the number of additional double columns added to $\boldsymbol{H}_{\frac{1}{2}}$, and $n_p = 5$ is the number of columns in $\boldsymbol{H}_{\frac{1}{2}}$. The optimized permutation matrices $\prod_i$, $i = 9, 10, \dots$ for rates $2/3$, and $4/5$ can be found in [241].

3) DVB/DVB-S2

In 2005, the 2nd generation of Digital Video Broadcasting (DVB-S2) became the first standard to adopt LDPC codes [277]. The standard defines four different system configurations and applications: broadcasting, interactive services, digital satellite news gathering (DSNG), and professional services. The systematic non-QC-LDPC codes in this standard are constructed via the accumulation of information bits, which the addresses of the parity bits are specified in Annex B in [277]. There are 21 LDPC codes in total included in the standard, 10 supported code rates for the short frame ($N = 16200$) and 11 code rates are supported for the normal frame ($N = 64800$). Moreover, the DVB-S2X [188] is the next-generation satellite transmission standard which is an extended version of its predecessor DVB-S2. Apart from the 21 LDPC codes in DVB-S2, the new specification introduced additional 34 LDPC codes with 3 new codes specified for the new frame length of 32400. The new specifications in DVB-S2X allow for spectral efficiency gains of up to 50% by offering lower roll-off factors, higher modulation orders, and finer code rate granularity compared to DVB-S2. Table 17 summarized the LDPC codes in the DVB-S2/DVB-S2X standards.

4) 3GPP 5G NR

According to the 5G NR specifications, URLLC and mMTC services are sensitive to latency and hence used for high-reliability short data transmissions, while eMBB targets

TABLE 17. DVB-S2/DVB-S2X LDPC code parameters.

Blocklength N	Submatrix size Z	Code rates R
16200	360	1/4, 1/3, 2/5, 1/2, 3/5, 2/3, 3/4, 4/5, 5/6, 8/9 (DVB-S2) 11/45, 4/15, 14/45, 7/15, 8/15, 26/45, 32/45 (DVB-S2X*)
32400	360	1/5, 11/45, 1/3 (DVB-S2X)
64800	360	1/4, 1/3, 2/5, 1/2, 3/5, 2/3, 3/4, 4/5, 5/6, 8/9, 9/10 (DVB-S2) 2/9, 13/45, 9/20, 11/20, 26/45, 28/45, 23/36, 25/36, 13/18, 7/9, 90/180, 96/180, 100/180, 104/180, 116/180, ... 124/180, 128/180, 132/180, 135/180, 140/180, 154/180, 18/30, 20/30, 22/30 (DVB-S2X*)

$*$ denotes additional codes added on top of DVB-S2.

transmitting large blocks of data with high throughput. The 5G deployment scenarios of eMBB require the support of not only a high throughput of up to 20 Gbps, but also a wider range of code rates, code lengths, and modulation formats than 4G LTE. Hence, QC-LDPC codes are recommended as the channel coding scheme for eMBB [278]. In particular, the recommended code lengths for eMBB ranged from 100 bits to 8000 bits, and code rates ranged from $1/5$ to $8/9$. Furthermore, the promising transport block error rate (BLER) performance at 10^{-2} and invisible error floor down to BLER of below 10^{-4} for code blocks are required for the QC-LDPC codes in 5G standard.

Besides QC structure, the 5G NR LDPC codes simultaneously possess rate compatibility and support multiple lifting factors. The rate compatibility of 5G LDPC codes is effectively implemented with the aid of a raptor-like code structure. Moreover, as multiple lifting sizes are supported, a vast range of information lengths and code rates can be easily adapted. Furthermore, 5G LDPC codes support code rate adjustment up to bit-level granularity. This is accomplished by performing puncturing, shortening, and repetition of coded bits after the lifting process of the base matrix. Such a process is called the *rate-matching*, which is one of the important modules in a practical 5G LDPC coding/decoding chain.

1) Code Structure: The LDPC codes for 5G New Radio (NR) are ensembles of PBRL-LDPC codes. The base matrix structure can be represented as

$$\boldsymbol{B}_{\mathbf{5G}} = \begin{bmatrix} \boldsymbol{B}_{\text{core}} & \mathbf{0} \\ \boldsymbol{B}_{\text{ex}} & \boldsymbol{I} \end{bmatrix}, \tag{112}$$

where $\boldsymbol{B}_{\text{core}}$ denotes the dense core matrix with dual-diagonal code structure as LDPC codes in IEEE 802.11n and 802.16e and $\boldsymbol{B}_{\text{ex}}$ denotes the sparse matrix indicating the connection in the single parity-check (SPC) extension of the 5G LDPC codes from high rate to low rate. The $\mathbf{0}$ and $\boldsymbol{I}$ indicate the zeros and identity matrices, respectively.

Two base matrices, namely base graph 1 (BG1) and base graph 2 (BG2), are adopted for 5G LDPC codes. Both BG1 and BG2 have similar code structures as shown in (112), but BG1 supports information blocks up to 8448 bits and code rates from $1/3$ to $8/9$, while BG2 supports information blocks up to 3840 bits and code rates from $1/5$ to $2/3$ [279].

As mentioned earlier, QC-LDPC codes with dual-diagonal structure yield efficient encoding on its parity-check matrix and hence no generator matrix is required. 5G LDPC codes also benefit from this property due to its dual-diagonal structure in the core matrix $\boldsymbol{B}_{\text{core}}$. The base matrix format of both BGs in (112) can be sub-divided and represented as

$$\boldsymbol{B}_{\mathbf{5G}} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} & \mathbf{0} \\ \boldsymbol{C} & \boldsymbol{D} & \boldsymbol{I} \end{bmatrix}, \tag{113}$$

where the size of each sub-divided matrix is given by:

- $\boldsymbol{A}$: $g \times k_b$.
- $\boldsymbol{B}$: $g \times g$.
- $\boldsymbol{C}$: $(m_p - g) \times k_b$.
- $\boldsymbol{D}$: $(m_p - g) \times g$.
- $\boldsymbol{I}$: $(m_p - g) \times (m_p - g)$ identity matrix.
- $\mathbf{0}$: $g \times (m_p - g)$ all-zero matrix.

The parameter $g = 4$ is set for both BGs, whereas $n_p = 68, m_p = 46$ for BG1 and $n_p = 52, m_p = 42$ for BG2. The information block, denoted by k_b, is 22 for BG1 and 10 for BG2. The base matrix $\boldsymbol{B}_{\mathbf{5G}}$ is lifted using CPMs of one of the supported lifting factors Z to obtain a derived Tanner graph of desired length and rate. Hence, the resulting LDPC code is in the QC structure.

2) Encoding of 5G NR LDPC Codes: The encoding of 5G NR LDPC codes is performed on the parity-check matrix by solving the equation

$$\boldsymbol{x} \cdot \boldsymbol{H}^T = \mathbf{0}, \tag{114}$$

where $\boldsymbol{x}$ is a systematic codeword and T denotes the matrix transpose. In the following, the encoding process is performed on a base matrix $\boldsymbol{B}_{\mathbf{5G}}$ rather than on a derived graph $\boldsymbol{H}$. The encoding process on a derived graph will be the same except that the complexity is up-scaled by a factor of Z. In the following, a two-stage encoding process for 5G LDPC codes using the BG1 represented in the form of (113) is introduced: 1) encoding of the core check $\boldsymbol{B}_{\text{core}}$ and 2) encoding of the SPC $\boldsymbol{B}_{\text{ex}}$.

Let $\boldsymbol{x} = [\boldsymbol{s}, \hat{\boldsymbol{p}}, \bar{\boldsymbol{p}}]$ be a codeword, where $\boldsymbol{s} = [s_0, s_1, \ldots, s_{k_b-1}]$ denotes the information block of size k_b. Let $\boldsymbol{p} = [\hat{\boldsymbol{p}}, \bar{\boldsymbol{p}}]$ be the m_p parity blocks, where $\hat{\boldsymbol{p}} = [\hat{p}_0, \hat{p}_1, \ldots, \hat{p}_{g-1}]$ and $\bar{\boldsymbol{p}} = [\bar{p}_0, \bar{p}_1, \ldots, \bar{p}_{m_p-g-1}]$ and $g = 4$. Then, the precise representation of codeword $\boldsymbol{x}$ is

$$\boldsymbol{x} = [s_0, s_1, \ldots, s_{k_b-1}, \hat{p}_0, \hat{p}_1, \ldots, \hat{p}_{g-1}, \bar{p}_0, \bar{p}_1, \ldots, \bar{p}_{m_p-g-1}]. \tag{115}$$

The encoding of 5G NR LDPC codes is carried out by solving

$$\boldsymbol{B_{5G}} \cdot \boldsymbol{x}^T = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} & \boldsymbol{0} \\ \boldsymbol{C} & \boldsymbol{D} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{s}^T \\ \hat{\boldsymbol{p}}^T \\ \bar{\boldsymbol{p}}^T \end{bmatrix} = \boldsymbol{0}. \tag{116}$$

Equation (116) can be naturally split into two equations, as follows:

$$\boldsymbol{A}\boldsymbol{s}^T + \boldsymbol{B}\hat{\boldsymbol{p}}^T = \boldsymbol{0}, \tag{117a}$$

$$\boldsymbol{C}\boldsymbol{s}^T + \boldsymbol{D}\hat{\boldsymbol{p}}^T + \boldsymbol{I}\bar{\boldsymbol{p}}^T = \boldsymbol{0}. \tag{117b}$$

From these two equations, it can be seen that the parity bits can be computed in two steps. The first step is to calculate $\hat{\boldsymbol{p}}$ from equation (117a), and the second step is to compute $\bar{\boldsymbol{p}}$ from (117b) using the computed $\hat{\boldsymbol{p}}$.

Consider the first step, rewrite (117a) into the form

$$\boldsymbol{A} \begin{bmatrix} s_0 \\ s_1 \\ \vdots \\ s_{k_b-1} \end{bmatrix} + \boldsymbol{B} \begin{bmatrix} \hat{p}_0 \\ \hat{p}_1 \\ \hat{p}_2 \\ \hat{p}_3 \end{bmatrix} = \boldsymbol{0}, \tag{118}$$

where $\boldsymbol{A} = [a_{i,j}]^{g \times k_b}, 0 \leqslant i \leqslant g-1, 0 \leqslant j \leqslant k_b - 1$ and

$$\boldsymbol{B} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}. \tag{119}$$

By expanding (118) and perform matrix multiplication, the following is obtained due to modulo-2 addition:

$$\sum_{i=0}^{3} \sum_{j=0}^{k_b-1} a_{i,j} s_j + \hat{p}_0 = 0. \tag{120}$$

Hence, the first parity block $\hat{p}_0$ of $\hat{\boldsymbol{p}}$ is computed by accumulating all the results from $\boldsymbol{A}\boldsymbol{s}^T$. Let

$$\theta_i = \sum_{j=0}^{k_b-1} a_{i,j} s_j \text{ for } i = 0, 1, 2, 3. \tag{121}$$

The individual parity blocks in $\hat{\boldsymbol{p}} = [\hat{p}_0, \hat{p}_1, \hat{p}_2, \hat{p}_3]$ can be expressed as the following set of equations:

$$\hat{p}_0 = \sum_{i=0}^{3} \theta_i, \tag{122a}$$

$$\hat{p}_1 = \theta_0 + \hat{p}_0, \tag{122b}$$

$$\hat{p}_2 = \theta_2 + \hat{p}_3, \tag{122c}$$

$$\hat{p}_3 = \theta_3 + \hat{p}_0. \tag{122d}$$

The above equations compute each θ_i value by accumulating all the $a_{i,j}s_j$ values. In modulo-2 operation, θ_i is obtained by carrying out XOR operations on all the elements of $a_{i,j}s_j$. Furthermore, it is worth to mention that the submatrix $\boldsymbol{B}$ in (118) is different for BG2.[3] Hence, the equations in (122a) - (122d) are different when performing encoding for BG2.

In the second step, the $\bar{\boldsymbol{p}}$ portion can be easily determined based on (117b). By performing rearrange and expansion, each of the parity blocks in $\bar{\boldsymbol{p}} = [\bar{p}_0, \bar{p}_1, \ldots, \bar{p}_{m_p-5}]$ can be computed using the following equations:

$$\bar{p}_i = \sum_{j=0}^{k_b-1} c_{i,j} s_j + \sum_{l=0}^{3} \hat{p}_l d_{i,k_b+l}, i = 0, 1, \ldots, m_p - 5. \tag{123}$$

In the actual encoding process, the entries $a_{i,j}$, $c_{i,j}$, and $d_{i,j}$ represent a CPM with a circular shift value. Thus, the multiplication of $a_{i,j}s_j$, $c_{i,j}s_j$, and $\hat{p}_l d_{i,k_b+l}$ in the above calculations are multiplication between a CPM matrix and a vectors sequence. Finally, combining the information block $\boldsymbol{s}$ and the parity block $\boldsymbol{p} = [\hat{\boldsymbol{p}}, \bar{\boldsymbol{p}}]$, the encoded systematic codeword $\boldsymbol{x}$ is obtained.

F. ITERATIVE MESSAGE PASSING DECODING OF LDPC CODES

Decoding of LDPC codes happens on Tanner graph using iterative message passing decoding algorithms. There are two types of LDPC decoders have been investigated and researched in the past decades: hard-decision decoders and soft-decision decoders. The difference between the two types of decoders is the input to the decoder and the format of the message passed along each edge during the iterative decoding process. The input to a hard-decision decoder is a binary sequence representing the sign of each bit and is passed along edges for the iterative decoding process. A majority-based decision rule is made after each decoding iteration to determine the sign of each bit. On the other hand, the input to a soft-decision decoder is a sequence of log-likelihood ratio (LLR), which is in the range of $[+\infty, -\infty]$. The magnitude of an LLR represents the reliability of the bit, and the sign represents the polarity of the bit. The sequence of LLRs is passed along the edges of a Tanner graph to update the reliability of each bit iteratively. The final decision is made according to the sign of the LLR of each bit. Alternatively, there are decoders proposed in the literature which combine the features of both hard-decision and soft-decision decoders. In this case, the input of a decoder is a sequence of LLR, whereas only the sign of each variable node is passed along the edges to perform the iterative decoding process. A majority-based decision rule together with weighting is used to determine the sign of each bit at the end of each iteration. One of the well-known examples of such a type is the *weighted bit-flipping* (WBF) decoder [207], [216].

For the sake of simplicity, consider that binary codeword $\boldsymbol{x} = (x_0, x_1, \ldots, x_{N-1})$ is transmitted over a binary-input memoryless channel, and denoted by $\boldsymbol{y} = (y_0, y_1, \ldots, y_{N-1})$ the received sequence at the output of the channel. The nature of $\boldsymbol{y}$ depends on the channel:

- For the Binary Symmetric Channel (BSC), $\boldsymbol{y} \in \{0, 1\}^N$ is a binary sequence of length N, obtained by flipping each bit of $\boldsymbol{x}$ with crossover probability p.

[3] For BG2, the first column of $\boldsymbol{B}$ is $[1\ 0\ 1\ 1]^T$.

- For the Binary Erasure Channel (BEC), $\boldsymbol{y} \in \{0, 1, \mathcal{X}\}^N$, where $\mathcal{X}$ denotes an erasure. Each bit of $\boldsymbol{x}$ is either erased $(y_j = \mathcal{X})$ with probability of ϵ or perfectly received $(y_j = x_j)$ with probability of $1 - \epsilon$.
- For the Binary-input Additive White Gaussian Noise (BI-AWGN) channel, $\boldsymbol{y} \in \mathbb{R}^N$ is a length N real vector, obtained by $y_j = (1-2x_j)+w_j$, where $(1-2x_j) \in \{\pm 1\}$ is the BPSK modulation of the bit x_j, and w_j is the white Gaussian noise with zero mean and variance σ^2.

The commonly adopted message format in BP decoding is the LLR, which is represented as a ratio of the *a posterior probability* (APP) of the transmitted bits and of the channel output, that is,

$$r_j = \log\left(\frac{\Pr(x_j = 0|y_j)}{\Pr(x_j = 1|y_j)}\right). \tag{124}$$

For the BSC channel with crossover probability p, the LLR of the j-th variable node is computed as

$$r_j = (1 - 2y_j)\log\left(\frac{1-p}{p}\right), \text{for } y_j = 0, 1. \tag{125}$$

For the BEC channel with erasure probability ϵ, the LLR of the j-th variable node is computed as

$$r_j = \begin{cases} +\infty & \text{if } y_j = 0, \\ -\infty & \text{if } y_j = 1, \\ 0 & \text{if } y_j = \mathcal{X} \end{cases} \tag{126}$$

For the BI-AWGN channel with noise variance σ^2, the LLR of the j-th bit is computed as

$$r_j = \frac{2y_j}{\sigma^2}. \tag{127}$$

Denoted by $\mathcal{A}(v)$ the set of indices with the corresponding CN connected to the VN v. Similarly, let $\mathcal{B}(c)$ be the set of indices with the corresponding VN connected to the CN c. Furthermore, let $V_{i,j}$ be the variable-to-check (V2C) message sent from the j-th VN to the i-th CN, and $E_{i,j}$ be the check-to-variable (C2V) message sent from the i-th CN to the j-th VN. Then the iterative decoding process is summarized in the following steps:

- The decoder takes $\boldsymbol{r} = \{r_0, r_1, \ldots, r_{N-1}\}$ as the input *a priori* information of the VNs, and it is computed from $\boldsymbol{y}$ depends on the nature of the channel.
- For each VN, the new V2C message $V_{i,j}^{(u)}$ sent out at iteration u is computed as a function $\mathcal{F}_1(*)$ of all the incoming C2V messages $E_{i',j}^{(u-1)}$ of the previous iteration and the initial channel LLR r_j, that is,

  $$V_{i,j}^{(u)} = \mathcal{F}_1\left(r_j, E_{i',j}^{(u-1)}\right), i' \in \mathcal{A}(v_j), i' \neq i. \tag{128}$$

  When $u = 0$, $E_{i,j}^{(u-1)} = 0$ for $0 \leqslant i \leqslant M - 1$ and $0 \leqslant j \leqslant N - 1$.
- For each CN, the new C2V message $E_{i,j}^{(u)}$ sent out at iteration u is computed as a function $\mathcal{F}_2(*)$ of the incoming V2C messages $V_{i,j'}^{(u-1)}$ of the previous iteration, that is,

  $$E_{i,j}^{(u)} = \mathcal{F}_2\left(V_{i,j'}^{(u-1)}\right), j' \in \mathcal{B}(c_i), j' \neq j. \tag{129}$$

- For each VN, the updated APP $\hat{r}_j$ at iteration u is computed as a function of r_j and all the incoming C2V messages $E_{i,j}^{(u)}$, that is,

  $$\hat{r}_j^{(u)} = \mathcal{F}_1\left(r_j, E_{i,j}^{(u)}\right), i \in \mathcal{A}(v_j). \tag{130}$$

  The tentative decision of the bit $\hat{x}_j, 0 \leqslant j \leqslant N-1$, is decoded to 0 if $\hat{r}_j^{(u)} \geqslant 0$ and is decoded to 1 otherwise.
- The iterative decoding process terminates and outputs the decoded sequence $\hat{\boldsymbol{x}} = (\hat{x}_0, \hat{x}_1, \ldots, \hat{x}_{N-1})$ if

  $$\boldsymbol{H}\hat{\boldsymbol{x}}^T = \boldsymbol{0} \mod 2. \tag{131}$$

  Otherwise, the iterative decoding process continuous until the maximum iteration number $I_{\max}$ is reached.

In the following, various iterative message-passing decoding algorithms are introduced.

G. LDPC DECODING ALGORITHMS

1) SUM-PRODUCT ALGORITHM

Along with the introduction of LDPC codes in Gallager's seminal work in 1960, a near-optimal probabilistic decoding algorithm for LDPC codes was introduced that is now called the *Sum-Product algorithm* [57], [91].

In the SPA decoding, the function to compute the C2V messages in (129) is explicitly given by

$$E_{i,j} = \prod_{j' \in \mathcal{B}(c_i), j' \neq j} \alpha_{i,j'} \cdot \Phi\left(\sum_{j' \in \mathcal{B}(c_i), j' \neq j} \Phi\left(\beta_{i,j'}\right)\right), \tag{132}$$

where $\alpha_{i,j'} = \text{sign}(V_{i,j'})$, $\beta_{i,j'} = |V_{i,j'}|$ and $\Phi(x) = -\log(\tanh(x/2))$. Next, the function to compute the V2C messages in (128) is explicitly given by

$$V_{i,j} = r_j + \sum_{i' \in \mathcal{A}(v_j), i' \neq i} E_{i',j}. \tag{133}$$

The updated APP of each VN in (130) is given by

$$\hat{r}_j = r_j + \sum_{i \in \mathcal{A}(v_j)} E_{i,j}. \tag{134}$$

2) MIN-SUM ALGORITHM (MSA)

While SPA has near-optimal error rate performance, it also has some drawbacks which may limit its use in practical applications. One of the drawbacks is the use of the computationally expensive $\Phi(*)$ function. The second drawback is the sensitivity of the BP decoding to the accuracy of the channel parameter estimate used in the initialization step to compute the a priori information $\boldsymbol{r}$.

Both drawbacks can be addressed by using an approximation to compute C2V message [280], [281]. It can be easily seen that the function $\Phi(*)$ in (132) is a decreasing function satisfying $\Phi(\Phi(x)) = x$. Therefore, for any set of b real values $(a_0, a_1, a_2 \ldots, a_{b-1})$, we have $\Phi(\sum_{i=0}^{b-1} \Phi(a_i)) \leqslant \Phi(\Phi(a_{i'})) = a_{i'}$ for $0 \leqslant i' \leqslant b-1$. Thus,

$$\Phi\left(\sum_{i=0}^{b-1} \Phi(a_i)\right) \leqslant \min_{0\leqslant i\leqslant b-1}(a_i). \tag{135}$$

Based on (135), the min-sum decoding was proposed to simplify the calculation (132) in SPA. More specifically, the computation of C2V messages $E_{i,j}$ is approximated as

$$E_{i,j} = \left(\prod_{j'\in\mathcal{B}(c_i), j'\neq j} \operatorname{sign}\left(V_{i,j'}\right)\right) \cdot \left(\min_{j\in\mathcal{B}(c_i), j'\neq j}\left(|V_{i,j'}|\right)\right). \tag{136}$$

Compared to the $\Phi(*)$ function in the SPA, MSA is commonly adopted in practice as it has low implementation cost because the check node operation is simplified to compare operations.

3) NORMALIZED AND OFFSET MSA

The MSA approximates SPA by assigning the upper bound value in (135) in each decoding iteration. Hence, the computed C2V messages are an overestimation of the true value computed via SPA. Due to this overestimation, the MS algorithm suffers from performance degradation compared to SPA. To reduce this performance gap, various improved MS-based decoding algorithms have been studied, e.g., [282], [283], [284]. The normalized min-sum (NMS) [283], [284] and the offset min-sum (OMS) [283] are probably the most popular ones, due to their simplicity. The NMS and OMS decoding algorithms rely on a scaling factor or an offset factor to compensate for the overestimation of the C2V messages. Hence, the computation of C2V messages $E_{i,j}$ is modified to

$$E_{i,j} = \left(\prod_{j'\in\mathcal{B}(c_i), j'\neq j} \operatorname{sign}\left(V_{i,j'}\right)\right) \cdot \left(\alpha \min_{j\in\mathcal{B}(c_i), j'\neq j}\left(|V_{i,j'}|\right)\right), \tag{137}$$

or

$$\begin{aligned} E_{i,j} = &\left(\prod_{j'\in\mathcal{B}(c_i), j'\neq j} \operatorname{sign}\left(V_{i,j'}\right)\right) \\ &\cdot \max\left(\min_{j\in\mathcal{B}(c_i), j'\neq j}\left(|V_{i,j'}|\right) - \beta, 0\right), \end{aligned} \tag{138}$$

where α is the scale factor adopted in the NMS algorithm, and β is the offset factor used in the OMS algorithm. For a range of SNR, the optimal values of α and β can be determined by Monte-Carlo simulation for regular LDPC codes, or through DE analysis for irregular LDPC codes [285], [286].

Although the performance of MS decoding can be improved by using a normalization factor or an offset factor, it is also important to properly tune these factors to avoid creating artificial error floors, particularly, in the case where only a limited room for optimization is provided such as fixed-point decoders implemented on a small number of bits. Consequently, the performance of the OMS and NMS may exhibit high error floors [287]. To overcome these drawbacks, two-dimensional (2-D) NMS and OMS decoding algorithms have been provided [288] [289], [290]. The 2D correction schemes rely on normalization factors (resp. offset factors) used to normalize (resp. offset) both V2C and C2V messages and whose values can be optimized as a function of the variable node or the check node degree.

4) SELF-CORRECTED MS

Another MS-based decoding algorithm, referred to as self-corrected MS (SCMS) decoding, was proposed in [291]. The main idea is to detect unreliable V2C messages during the iterative decoding process, and to erase them by setting their values to zero. More specifically, a V2C message $V_{i,j}^{(u)}$ in iteration u is assigned 0 if its sign changed with respect to the V2C message $V_{i,j}^{(u-1)}$ in iteration $u-1$. Hence, the computation of V2C messages $V_{i,j}$ is modified to

$$\begin{aligned} V_{tmp}^{(u)} &= r_j + \sum_{i'\in\mathcal{A}(v_j), i'\neq i} E_{i',j}^{(u-1)}; \\ V_{i,j}^{(u)} &= \begin{cases} 0 & \text{if} \operatorname{sign}\left(V_{i,j}^{(u-1)}\right) \neq \operatorname{sign}\left(V_{tmp}^{(u)}\right), V_{i,j}^{(u-1)} \neq 0, \\ V_{tmp}^{(u)} & \text{else}. \end{cases} \end{aligned} \tag{139}$$

The performance of SCMS decoding is very close to BP in the error floor region [291], [292]. Moreover, its built-in feature of erasing unreliable messages can also be advantageously exploited for energy-efficient implementations [293], [294].

5) APPROXIMATE-MIN

Approximate-min (A-min*) [295] decoding algorithm is an approximation of the SPA decoder by applying the Jacobian logarithmic identity to change the way of computing the C2V messages $E_{i,j}$ using a recursive method. Let Λ^* be the recursive function

$$\Lambda^*(a, b) = \operatorname{sign}(a) \operatorname{sign}(b) \cdot \mathcal{J}(a, b),$$

where

$$\begin{aligned} \mathcal{J}(|a|, |b|) = &\left(\min(|a|, |b|) + \ln\left(1 + e^{-||a|+|b||}\right)\right. \\ &\left. - \ln\left(1 + e^{-||a|-|b||}\right)\right). \end{aligned}$$

The C2V message of a CN c_i to VN v_j is computed recursively as

$$E_{i,j} = \Lambda^*\left(V_{i,j'_{|\mathcal{B}(c_i)|}}, \ldots, \Lambda^*\left(V_{i,j'_4}, \Lambda^*\left(V_{i,j'_3}, \Lambda^*\left(V_{i,j'_1}, V_{i,j'_2}\right)\right)\right)\right), \tag{140}$$

where $\{j'_1, j'_2, \ldots, j'_{|\mathcal{B}(c_i)|}\} \subset \mathcal{B}(c_i)$ which excludes j. Let

$$E_{i,j} = \boldsymbol{\Lambda}^*{}_{j'\in\mathcal{B}(c_i), j'\neq j} V_{i,j'} \tag{141}$$

be the recursive operation given in (140). Then the check node computation is performed in the following steps:

- For each CN c_i, find the incoming V2C message with the minimum magnitude and label its source VN as $v_{j_{min}}$ and the V2C message as $V_{i,j_{min}}$.
- The C2V message $E_{i,j_{min}}$ send to the VN $v_{j_{min}}$ is the calculation (140) with $j = j_{min}$.
- For all other variable node $j \in \mathcal{B}(c_i), j \neq j_{min}$,

$$E_{i,j} = \left(\prod_{j' \in \mathcal{B}(c_i), j' \neq j} \operatorname{sign}\left(V_{i,j'}\right) \right) \cdot \Lambda^*\left(E_{i,j_{min}}, V_{i,j_{min}}\right)$$

Observe that in the A-min* approximation of SPA, only two magnitudes are computed at each check node, requiring only $d_c - 1$ computation of Λ^* function to compute both, where d_c is the check node degree. Moreover, compared to MS-based decoding algorithms, the message sent to the least reliable variable node $v_{j_{min}}$, in the A-min* decoder is exactly that of the SPA decoder. This explains the performance improvement of the A-min* decoder over the MS decoder and its negligible loss relative to the SPA decoder.

6) ADJUSTED MS

The Adjusted min-sum (AdjMS) algorithm of LDPC codes is proposed in [296], and the C2V approximation function $\hat{\Lambda}^*(a, b)$ is given by

$$\hat{\Lambda}^*(a, b) \approx \min(a, b) - f(|a - b| + h(M)), \tag{142}$$

where $f(x) = \log(1 + \exp^{-x})$ and $h(x)$ may be defined as $\log(\coth(x))$ or $-\log(1 - \exp^{-2x})$.

Unlike the conventional implementation of MS-based algorithms, where the incoming V2C messages with minimum two magnitudes are found to update the outgoing C2V messages, the C2V messages in the AdjMS decoding rely on the maximum and minimum value of the incoming $V_{i,j}$. Similar to A-min*, AdjMS applies the approximation function $\hat{\Lambda}^*$ to recursively calculate two values assigned to $E_{i,j}$. Let $j_{\max}$ and j_{min} be the index of the incoming V2C messages with the maximum and the minimum magnitude. Then the C2V messages computed for check node i are given by

$$E_{tmp} = \hat{\boldsymbol{\Lambda}}^*{}_{j' \in \mathcal{B}(c_i), j' \neq j_{min}, j' \neq j_{\max}} V_{i,j'} \tag{143}$$

and

$$\begin{aligned} E_{i,j \neq j_{min}} &= \hat{\Lambda}^*(V_{i,j_{min}}, E_{tmp}), \\ E_{i,j_{min}} &= \hat{\Lambda}^*(V_{i,j_{\max}}, E_{tmp}). \end{aligned} \tag{144}$$

In [297], the error rate performance of AdjMS over a range of code lengths and code rates is shown. The results show that the AdjMS algorithm with layered scheduling[4] can achieve SPA decoding performance with flood scheduling using only half of the required iterations.

H. QUASI-MAXIMUM LIKELIHOOD (QML) DECODING OF LDPC CODES

It is shown in [298] that the SPA is sub-optimal and has a considerable performance gap to the maximum likelihood (ML) decoding with short blocklength codes, which is due to the existence of small cycles in their associated Tanner graphs. To further improve code performance, the concept of QML decoding has been further investigated in recent years. One of the first QML decodings is the ordered statistic decoding (OSD) [299], and later the idea was adopted in the decoding of LDPC codes by the same author in [300]. Various research has been conducted in this direction to reduce the complexity of QML decoders *e.g.,* [290], [301], [302], [303], [304]. The most common strategy adopted in these works are to introduce multiple rounds of *decoding tests*, so-called *reprocessing*, after the failure of the conventional BP decoding. More specifically, the decoder is reinitialized with a list of different decoder inputs during the reprocessing, where each input sequence is generated by substituting the channel outputs of the selected unreliable VNs with the maximum or minimum values. The conventional BP decoding is conducted with each input sequence, and the decoding output is stored if it generates a valid codeword. The 'best' codeword is chosen from the list of valid codewords as the decoder output according to a certain decision metric. This type of decoder is referred to as the *list* decoder.

I. MACHINE LEARNING-BASED LDPC DECODERS

In a communication system, a large number of signal-processing tasks, such as detection and decoding, can be formulated as optimization problems. Conventionally, these optimization problems are typically solved using numerical algorithms that iteratively refine the solution. However, in practice, the iterative process can only be afforded with only a small number of iterations. To accurately find the solution with a small number of iterations, the numerical solver requires parameter tuning. The most straightforward way to tune the algorithm parameters is by using heuristic approaches based on simulation results. However, such conventional approaches are prone to result in suboptimal performance and may cause stability issues if the system conditions change. Following the idea of introducing intelligence to the future B5G or 6G wireless networks, various machine learning-aided approaches have been developed to reinforce the design of next-generation wireless communication systems, such as signal detection and channel encoding and decoding [305], [306], [307], [308], [309].

Among all the deep learning-based data detection or channel decoding algorithms, "deep unfolding" [310] is an efficient method to improve an existing algorithm's performance. More specifically, deep unfolding takes an iterative algorithm with a fixed number of iterations, unfolds its structure onto several hidden layers of a *neural network* system, and introduces a number of trainable parameters such

[4] Decoder scheduling will be further discussed in the next section.

as multiplicative weights and bias. Hence, deep learning has become one of the promising optimization tools for iterative decoding of linear block codes, and much research has been dedicated to this direction in recent years. For instance, in [311], [312], [313], [314], [315], an off-line training model for BP decoding of linear block codes with high-density (also sometimes referred to as high-density parity-check codes) codes were investigated. Moreover, research on machine learning-aided LDPC decoding has also attracted a lot of attention. In [316], unfolding MS algorithm to decode LDPC codes is proposed by introducing additional parameters tuning the scales and offset of the standard NMS and OMS algorithms for 5G LDPC codes, respectively. The work of [317] proposes the idea of unfolding to learn finite-alphabet (FA) decoding of LDPC codes. Simulation results show that by unfolding and learning FA decoders, gains of up to 0.3 dB can be achieved for a $(1296, 972)$ QC-LDPC code over conventional MS decoder when using 3 quantization bits. The authors in [318] proposed a neural 2D normalized MS decoder, together with various weight-sharing techniques to reduce the number of parameters that must be trained. Furthermore, the machine learning-aided decoding for photograph LDPC codes has been investigated in [319], along with a trajectory-based extrinsic information transfer (T-EXIT) chart developed for various decoders. Various other machine learning-related research on LDPC decoding have also been conducted. For instance, in [320] a syndrome-based neural network is used to estimate the LLR for flash memory controller with LDPC codes. In [321], the optimal quantization bits are estimated using machine learning, and the works in [322] proposed a machine learning-based quantization decoding for 5G LDPC codes. In [323], a pruning-based neural BP decoder is proposed, resulting in a different parity-check matrix in each decoding iteration.

J. DECODING OF NON-BINARY LDPC CODES

The non-binary (NB) formulation of LDPC codes over $GF(q)$, $q > 2$, was proposed by Davey and MacKay in the late 90s [324]. Let $GF(q)$ be a Galois field with q elements, where q is a power of a prime. A q-ary LDPC code of length N is given by the null space over $GF(q)$ of the sparse parity-check matrix $\boldsymbol{H}_{NB}$ over $GF(q)$. The non-zero elements of $\boldsymbol{H}_{NB}$ are represented by symbols contained in the respective $GF(q)$ or the binary m-tuple in the extension field $GF(2^m)$, such that for any valid codeword $\boldsymbol{x} \in GF(q)$

$$\boldsymbol{H}_{NB} \otimes \boldsymbol{x}^T = \boldsymbol{0}, \tag{145}$$

where $\otimes$ denotes the multiplication over $GF(q)$.

The BP decoding of non-binary LDPC codes, known as the q-ary SPA, is introduced in [324], and later on improved by using fast Fourier transform (FFT) [325], [326], [327]. Denoted by $\boldsymbol{z} = \{z_0, z_1, \ldots, z_{N-1}\}$ the hard-decision of the received sequence $\boldsymbol{y}$. In binary cases, only two prior probabilities of the j-th received symbol are needed, $\Pr(z_j = 0)$ and $\Pr(z_j = 1)$, and the two probabilities can be compactly represented in the form of LLR when passing along edges in a Tanner graph. In q-ary SPA decoding of a non-binary LDPC code, a set of q prior probabilities represent the j-th received symbol, that is $P_j^{\alpha_k} = \Pr(z_j = \alpha_k)$ for all elements α_k, $1 \leqslant k \leqslant q$, in the $GF(q)$. Denoted by $\boldsymbol{\omega^{\alpha}}_{i,j} = (\omega_{i,j}^{\alpha_1}, \omega_{i,j}^{\alpha_2}, \ldots, \omega_{i,j}^{\alpha_q})$ and $\boldsymbol{\theta^{\alpha}}_{i,j} = (\theta_{i,j}^{\alpha_1}, \theta_{i,j}^{\alpha_2}, \ldots, \theta_{i,j}^{\alpha_q})$ the set of probability message sent from VN v_j to CN c_i and CN c_i to VN v_j, respectively. Upon receiving these incoming prior probabilities, the CN computes the outgoing C2V messages as

$$\theta_{i,j}^{\alpha_k} = \sum_{\boldsymbol{z}:z_j=\alpha_k} \Pr(s_i = 0|\boldsymbol{z}, z_j = \alpha_k) \cdot \prod_{j' \in \mathcal{B}(c_i), j' \neq j} \omega_{i,j'}^{z_{j'}}, \tag{146}$$

where $\boldsymbol{s} = \{s_0, s_1, \ldots, s_{M-1}\}$ is the syndrome vector obtained from

$$\boldsymbol{H}_{NB} \otimes \boldsymbol{z}^T = \boldsymbol{s}.$$

The probability $\Pr(s_i = 0|\boldsymbol{z}, z_j = \alpha_k) = 1$ when $z_j = \alpha_k$ and $\boldsymbol{z}$ satisfies the i-th checksum, i.e., $s_i = 0$; otherwise the probability equals the zero. The computed C2V messages $\boldsymbol{\theta^{\alpha}}_{i,j}$ are then used to update the V2C probability $\boldsymbol{\omega^{\alpha}}_{i,j}$ for the next decoding iteration

$$\omega_{i,j}^{\alpha_k} = f_{i,j} \cdot P_j^{\alpha_k} \prod_{i' \in \mathcal{A}(v_j), i' \neq i} \theta_{i',j}^{\alpha_k}, \tag{147}$$

where $f_{i,j}$ is the normalization term to ensure that $\sum_{k=1}^{q} \omega_{i,j}^{\alpha_k} = 1$. The tentative decision of $\boldsymbol{z} = \{\hat{z}_1, \hat{z}_2, \ldots, \hat{z}_N\}$ is estimated based on

$$\hat{z}_j = \arg\max_{\alpha_k} P_j^{\alpha_k} \prod_{i \in \mathcal{A}(v_j)} \theta_{i,j}^{\alpha_k}. \tag{148}$$

The decoding process continues if $\boldsymbol{H}_{NB} \otimes \hat{\boldsymbol{z}}^T \neq 0$ or until the maximum decoding iteration is reached.

For each nonzero entry in $\boldsymbol{H}_{NB}$, the number of computations required to compute the probability messages passing between CN c_i to VN v_j in each decoding iteration is on the order of $\mathcal{O}(q^2)$. For large q, the computational complexity may become prohibitively large. The FFT-based $q-$ary SPA has complexity reduced to $\mathcal{O}(q \log q)$, or $\mathcal{O}(pq)$ for $q = 2^p$. Furthermore, the work in [327] proposed the *extended min-sum* (EMS) algorithms to simplify message update at the check-node output by computing suboptimal reliability measures with low computational complexity rather than the extract reliability in q-ary SPA. To make this happen, the concept of the *configuration set* is introduced such that only a subset n_m of the most significant values of each vector at a check node is used to compute the output reliability, resulting in a complexity order of $n_m q$. Since then, low-complexity decoding of non-binary LDPC code become one of the toughest problems in the field for many years, and many related works to non-binary LDPC codes have been investigated. In [328], the 'Min-max' algorithm was introduced with the same complexity order as q-ary SPA, but using only addition and comparison operations. The symbol-flipping decoding (SFD) [329], [330] for non-binary LDPC

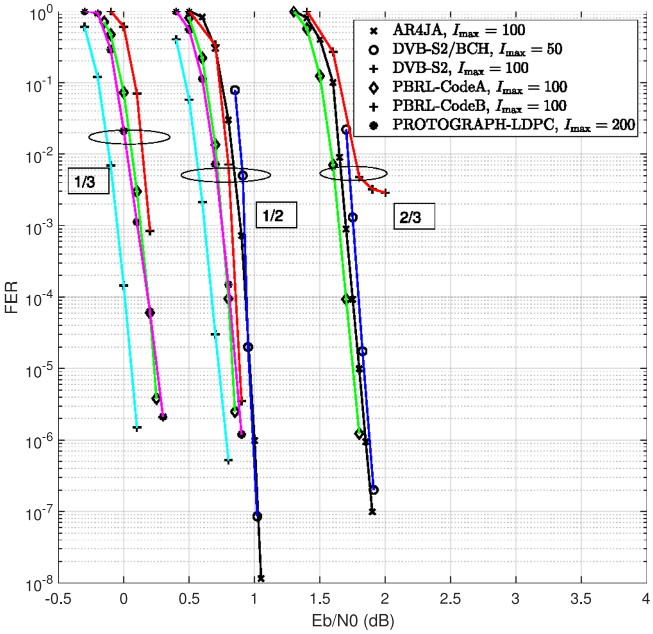


FIGURE 42. FER performance of rate-compatible protograph-based LDPC codes of different structures.

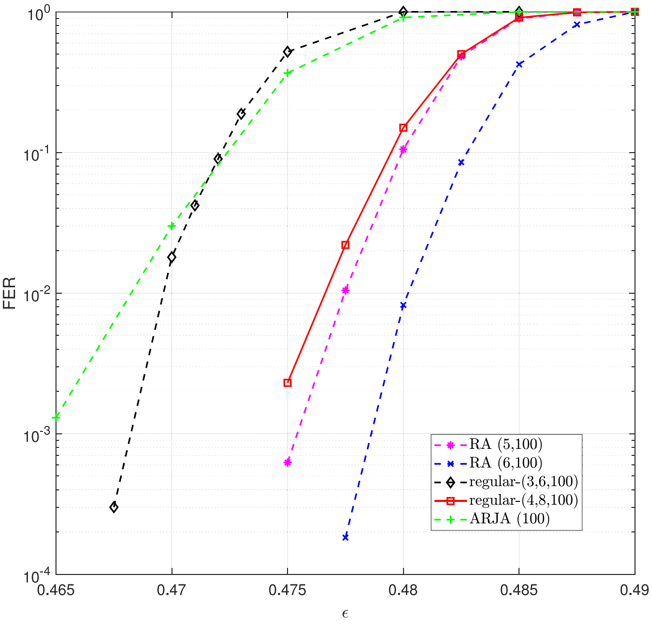


FIGURE 43. FER for different protograph SC-LDPC constructions with $L = 100$ and $N = 2000$ bits.

codes was explored intensively by using flipping metrics to determine which unreliable symbols should be flipped during the iterations. Improved SFD based on prediction (SFDP) has been investigated in [331], [332]. In [333], the early termination architecture of a modified Trellis Min-Max (T-MM) decoding algorithm for non-binary LDPC codes is presented, and the maximum achievable throughput is 4.68 Gb/s. The work in [334] investigated the decoding of non-binary LDPC codes over subfields by adopting the method of expanding a non-binary Tanner graph over a finite field into a graph over a subfield, resulting in a reduction of decoding complexity. Moreover, the analysis of the trapping set and absorbing set for non-binary protograph LDPC codes is investigated in [335].

K. PERFORMANCE COMPARISON

1) CODE CONSTRUCTION COMPARISON

Fig. 42 compares the FER performance of LDPC codes of different structures at rates 1/3, 1/2, and 2/3. More specifically, the PBRL LDPC codes, denoted as 'PBRL-CodeA' and 'PBRL-CodeB' [201], and the rate-compatible protograph LDPC code, denoted as 'PROTOGRAPH-LDPC' [336], is compared to the AR4JA code from the CCSDS standard [241] and the LDPC code from the DVB-S2 standard (with and without an outer BCH code) [277]. The block lengths of the DVB-S2 codes are fixed to 64800 bits, whereas the PBRL and AR4JA codes have a fixed information length of 16368 bits and block lengths of 32736 bits and 24552 bits for rate 1/2 and rate 2/3, respectively. Moreover, the block lengths of the rate-compatible protograph LDPC code PROTOGRAPH-LDPC are 32736 bits and 49104 bits for rate 1/2 and rate 1/3, respectively.

From the figure, in the waterfall region, both of the PBRL codes outperform both the AR4JA codes and the DVB-S2 codes even though the DVB-S2 codes have longer code lengths and benefit from concatenation with a BCH code. Further, at a FER around 10^{-6}, the PBRL code PBRL-CodeB is 0.2 dB outperforms the PBRL-CodeA and the rate-compatible PROTOGRAPH-LDPC code at rate $1/3$, and about 0.1 dB better at rate $1/2$.

Fig. 43 shows the frame error rate performance for different protograph SC-LDPC codes constructed from the following component codes: 1) regular $(3, 6)$ and $(4, 8)$ LDPC block codes [180] with coupling memory $m_s = 2$ and $m_s = 3$, respectively; 2) rate-1/2 ARJA code [337] with $m_s = 1$; 3) regular $\mathrm{RA}(q, L)$ codes [338] with coupling memory $m_s = q$ and repetition factor $q = 5$ and $q = 6$, respectively, over the BEC channel. The coupling length of each code is $L = 100$, while the protograph lifting factor $M = 2000$. As shown in the figure, the ARJA(L) code has a poor finite-length performance, due to the high structure of the ARJA protograph with 5 variable nodes in the uncoupled protograph and the small block length N compared to the rest of the codes. It is also interesting to note that both spatially coupled RA codes outperform the regular $(4, 8, 100)$ SC-LDPC code in finite blocklength performance, although they have worse BP decoding threshold than SC-LDPC codes [339, Table III].

2) DECODING ALGORITHM COMPARISON

Fig. 44 and Fig. 45 show the frame error rate (FER) performance of 5G NR LDPC codes with information length $K = 120$ and $K = 8448$ simulated over an AWGN channel with 4-QAM modulation and maximum iteration number $I_{\max} = 50$. The codes are decoded using the SPA, the MS,

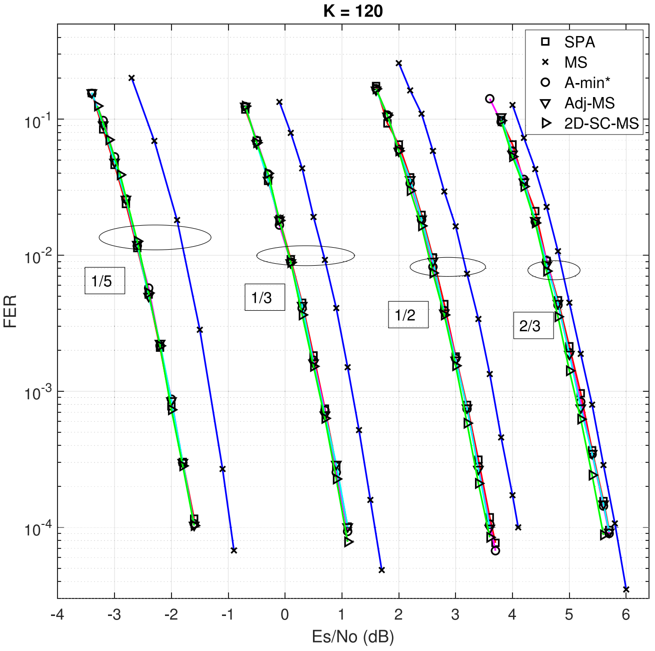


FIGURE 44. FER performance of 5G NR LDPC with $K = 120$, $R = 1/5, 1/3, 1/2, 2/3$.

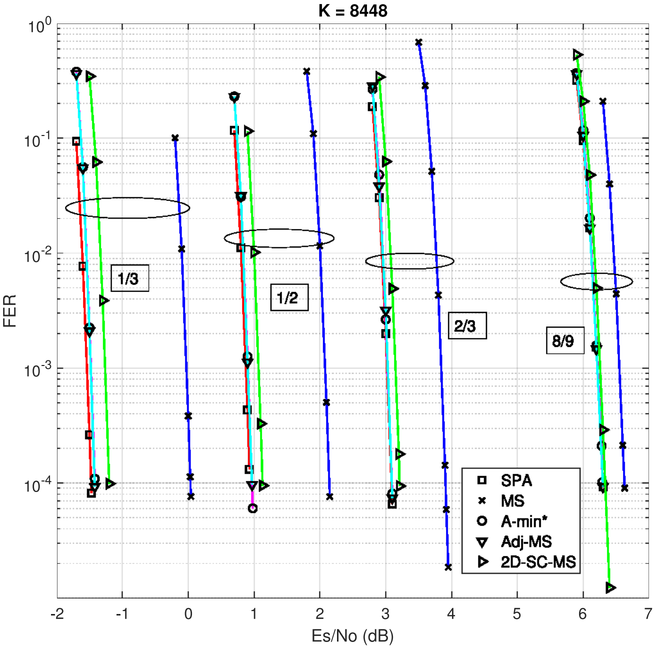


FIGURE 45. FER performance of 5G NR LDPC with $K = 8448$, $R = 1/3, 1/2, 2/3, 8/9$.

the A-min*, the AdjMS, and the 2D-SCMS [290] algorithms. From Fig. 44, the performance of A-min*, AdjMS, and 2D-SCMS are similar to SPA when the information block size K is small. The performance gap of MS decoding from SPA reduces as the code rate increases. However, when K is very large, the performance of AdjMS and A-min* is similar and has a negligible performance gap from the performance of SPA when the code rate reduces. On the other hand, the performance of 2D-SCMS decoding shows a slightly larger gap from SPA compared to AdjMS and A-min*, and the performance gap of MS decoding from SPA is much larger compared to AdjMS, A-min*, and 2D-SCMS.

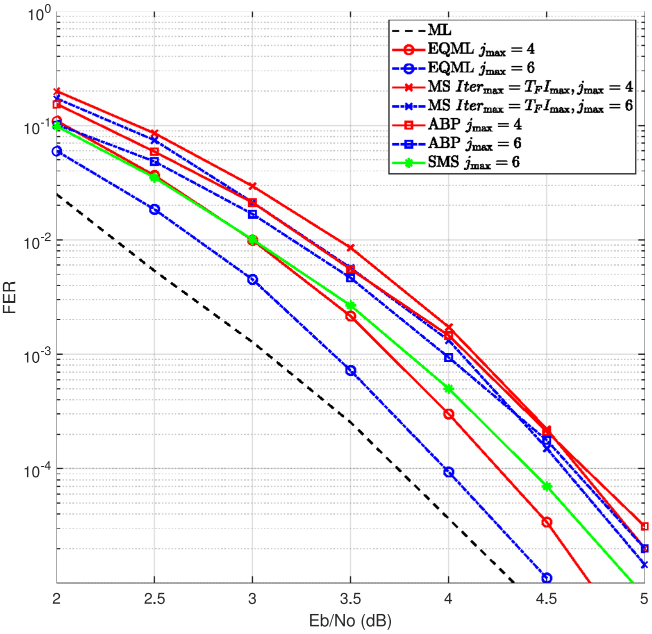


FIGURE 46. FER performance of the regular $(4, 7)$ LDPC code.

3) QUASI ML DECODING COMPARISON

Fig. 46 illustrated the FER performance of the regular $(4, 7)$ LDPC code with $K = 48$ and $R = 1/2$ [340] decoded over the AWGN channel with BPSK modulation and maximum decoding iteration of $I_{\max} = 30$. The parameter $j_{\max}$ is defined such that $T_F = 2^{j_{\max}+1} - 1$ is the number of the decoding tests that a QML decoder runs. Moreover, the conventional MS decoder is performed with the same number of decoding iterations as the QML decoders run, that is $Iter_{\max} = T_F I_{\max}$. From the figure, by performing reprocessing, the enhanced QML (EQML) [289], [290] decoder outperforms the MS decoder by about 0.5 dB and 0.6 dB for $j_{\max} = 4$ and $j_{\max} = 6$, respectively. The ML performance of the code [340], the FER performance of the augmented belief propagation (ABP) decoder [301] and the Saturated MS decoder (SMS) [302] are also shown in the figure. For $j_{\max} = 6$, the EQML decoder outperforms the ABP decoder and SMS decoder by about 0.6 dB and 0.4 dB, respectively, and it can approach the performance of the ML decoder within 0.3 dB at FER=10^{-4}.

Fig. 47 shows the FER performance of the 5G LDPC code with $K = 120$ and $R = 1/5$ decoded over AWGN channel with 4-QAM modulation and $I_{max} = 50$. By performing reprocessing, the enhanced QML (EQML) decoder outperforms the SPA decoder with the same number of decoding iterations as the EQML decoder by about 0.2 dB and 0.3 dB for $j_{\max} = 4$ and $j_{\max} = 6$, respectively. In addition, the normal approximation (NA) [23] bound for blocklength $N = 600$ is shown. The performance gap between the EQML

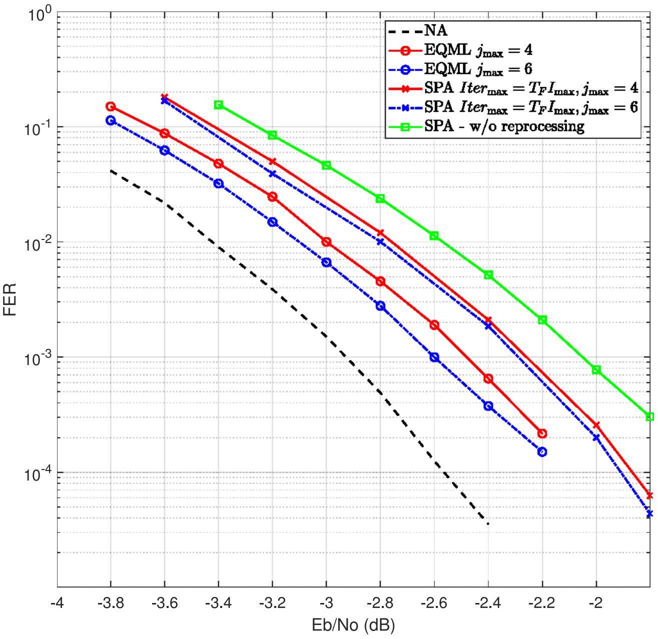


FIGURE 47. FER performance of 5G NR LDPC with $K = 120$, $R = 1/5$.

decoder with $j_{\max} = 6$ and the NA is within 0.6 dB at FER=10^{-4}.

L. IMPLEMENTATION OF LDPC DECODERS

1) DECODER SCHEDULING

The scheduling of the LDPC decoding process determines the order in which VNs and CNs are processed. The three most common scheduling methods, namely flooding [341], layered belief propagation (LBP) [342] and informed dynamic scheduling (IDS) [343], are described in the following.

1) Flooding: In a flooded LDPC decoder, all check nodes update the C2V messages simultaneously in the first half of the iteration, followed by all variable nodes updating the V2C messages simultaneously in the second half of the iteration. During each half of the iteration, only one side of the Tanner graph is activated and performs message calculation. An example of the flooded decoder is illustrated in Fig. 48. It can be seen in Fig. 48-a) that in the first half of the iteration, the CNs c_1, c_2, c_3, c_4 update their C2V messages, which are then sent to its neighboring VNs, while all the VNs are not computing. Fig. 48-b) illustrates the second half of the iteration, where VNs $v_1, v_2, \ldots, v_8$ update their V2C messages and send them to their neighboring CNs.

2) Layered: Layered BP (LBP) decoder, on the other hand, is operated in a sequential manner within each iteration. It sequentially processes each CN in turn. Once a CN has sent the updated C2V message to its neighboring VNs, these VNs perform V2C message calculations before moving on to the next CN. The iteration is complete when all CNs have been processed. Using Fig. 49 as an example, a layered LDPC decoder may commence each decoding iteration by activating CN c_1 first, sending the updated C2V messages to VNs v_1, v_4, v_5 and v_8 as shown in Fig. 49-a). Each of these VNs then activated, sending updated V2C messages to its neighboring CNs except c_1, as illustrated in Fig. 49-b). Fig. 49-c) and Fig. 49-d) show the similar process for CN c_2. The iteration completes until the rest of CNs c_3 and c_4 complete the process above.

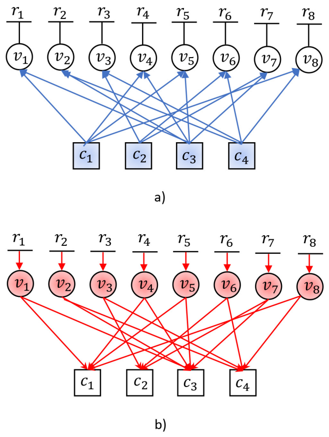


FIGURE 48. An example of the flooding schedule.

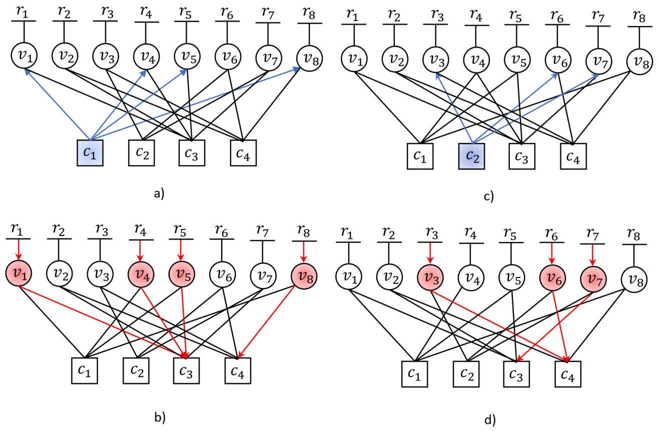


FIGURE 49. An example of the layered schedule.

The advantage of the layered LDPC decoder is that the information obtained during an iteration is available to aid the remainder of the iteration. In addition, the layered decoder tends to have a faster convergence speed than the flooded decoder, and hence less number of iterations are needed to converge to the correct codeword. However, the drawback of the layered decoder is that it does not have the same high level of parallelism as the flooded decoder, possibly resulting in low processing throughput and higher processing latency due to its sequential processing.

3) Informed Dynamic Scheduling: Informed dynamic scheduling (IDS) inspects the messages that are passed between the nodes, selecting to activate whichever node is expected to offer the greatest improvement in message belief. During the iterative decoding process, the inspection of messages requires additional calculations to determine

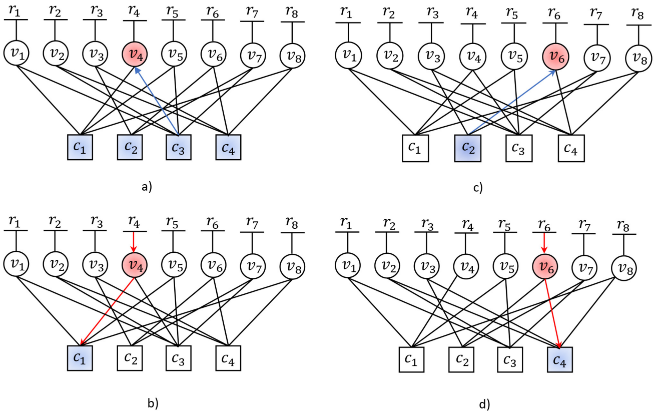


FIGURE 50. An example of informed dynamic scheduling.

which node to activate. The additional calculation takes place at the CNs, where the difference between the previous C2V message $E_{i,j}^{(u-1)}$ sent over an edge and the C2V $E_{i,j}^{(u)}$ that is obtained using recently-updated information in the current iteration. This difference is termed the *residual* and represents the improvement in the belief of the new C2V message $E_{i,j}$. At the start of the decoding iteration, the residual of each outgoing C2V message of each CN is calculated. As $E_{i,j}^{(-1)} = 0$, the residual equals the magnitude of the C2V message to be sent over that edge. The message with the greatest residual is identified and the receiving VN is then activated, sending updated V2C messages $V_{i,j}$ to each of its neighboring CNs. These CNs are then activated and calculate the residual for each of their edges. The new maximum residual is then obtained among all the residuals in the graph before the process is repeated.

Using the example code in Fig. 50, at the start of the decoding iteration, the C2V message $E_{3,4}$ from CN c_3 to VN v_4 is identified as having the highest magnitude of all the C2V messages in the Tanner graph. The VN v_4, after receiving the message from CN c_3, is then activated and calculates the updated V2C $V_{1,4}$, which is then passed to CN c_1. The CN c_1 can then be activated to calculate new residuals for its other three edges. The new residual is compared with others from the previous step, allowing a new maximum to be identified. As shown in Fig. 50-c), the next highest residual identified is the message from CN c_2 to VN v_6. Thus, VN v_6 is activated to calculate the updated V2C message sent to its neighboring CN c_4 as illustrated in Fig. 50-d). This implies that in the decoding of IDS, the next highest residual does not necessarily have to originate from the most recently updated CN c_1. Hence, a particular CN can be updated several times before another one is updated once.

2) DECODER ARCHITECTURE

The implementation of a practical LDPC decoder can be varied. The well-known architectures are fully parallel, unrolled fully parallel, and partially parallel. The requirements of a decoder are measured in several aspects, including area efficiency, energy efficiency, error performance, and throughput. A simple and effective model to estimate the throughput of a decoder architecture for LDPC block codes is based on the average number of edges the decoder architecture processes in one clock cycle, denoted as $Proc(\mathcal{E}(\boldsymbol{H}))$. Let $\mathcal{E}(\boldsymbol{H}) = Z\mathcal{E}(\boldsymbol{H}_b)$ be the total number of edges in a Tanner graph, where Z is the lifting size, and $\boldsymbol{H}_b$ is the protomatrix. The information throughput of an architecture for one iteration is estimated by

$$\hat{\mathcal{T}}(\boldsymbol{H}) = \frac{Proc(\mathcal{E}(\boldsymbol{H}))}{\mathcal{E}(\boldsymbol{H})} \cdot n_p \cdot Z \cdot f_{\max} \cdot R, \tag{149}$$

where n_p is the column number of a protomatrix, $f_{\max}$ is the maximum operating frequency of the decoder and R is the code rate. In practice, decoding of an LDPC code is an iterative process and hence the information throughput is estimated as $\mathcal{T}(\boldsymbol{H}) = \hat{\mathcal{T}}(\boldsymbol{H})/I_{\max}$, where $I_{\max}$ denotes the preset iteration number decoder performs.[5] Based on the model in (149), the three well-known architectures can be defined:

- Fully parallel: $\mathbf{Proc}(\mathcal{E}(\boldsymbol{H})) = \mathcal{E}(\boldsymbol{H})$,
- Unrolled fully parallel: $\mathbf{Proc}(\mathcal{E}(\boldsymbol{H})) > \mathcal{E}(\boldsymbol{H})$, and
- Partially parallel: $\mathbf{Proc}(\mathcal{E}(\boldsymbol{H})) < \mathcal{E}(\boldsymbol{H})$.

An illustration of three types of decoder architecture is given in Fig. 51, where rows that are processed in parallel are marked in yellow.

1) Fully Parallel: A fully parallel LDPC decoder is a realization of a flooded LDPC decoder described in the previous section. A fully parallel decoder consists of:

- Node processors (NP): the number of VN processors (VNP) is equal to the number of columns, $N = Zn_p$, in the parity-check matrix $\boldsymbol{H}$ and the number of CN processors (CNP) is equal to the number of rows $M = Zn_p$ in $\boldsymbol{H}$.
- Routing network (RNW): the routing network is represented by wires connecting the VNPs and CNPs according to $\boldsymbol{H}$.

As all VNs or CNs are activated simultaneously to update messages, only a small number of clock cycles is required for an iteration, resulting in high processing throughput, which turns out to be the biggest advantage for a fully parallel decoder. However, routing congestion, especially for large block sizes, is a major challenge in implementing the RNW due to the large number of wires needed to describe the connections between VNs and CNs. For LDPC codes that have thousands of VNs and CNs, the routing network involves tens of thousands of connections between VNPs and CNPs. Moreover, if $\boldsymbol{H}$ is an irregular structure, the interconnections of the RNW are highly irregular, which will further contribute to the increase in cost, as well as the reduction in the maximum operating frequency due to a high routing delay. Another drawback of a fully parallel LDPC

[5] If early stop option is adopted in the decoder, then the average iteration number $I_{avg\_iter} \leqslant I_{\max}$ is often considered.

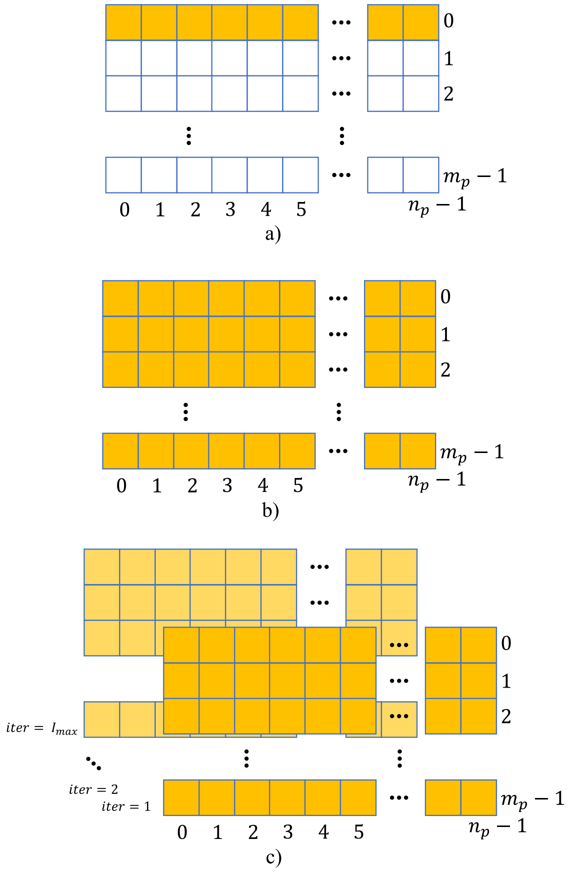


FIGURE 51. Decoder architectures for LDPC codes: a) partially parallel processing; b) fully parallel processing; c) unrolled fully parallel processing.

decoder is its low flexibility. Although the architecture has no limitations on the structure of $\boldsymbol{H}$, a decoder is often specific to an LDPC with fixed interconnections in the RNW. A redesign of the entire decoder is needed if the LDPC code is modified. Hence, this type of architecture cannot easily accommodate features such as reconfigurable decoders.

To reduce the complexity of these fully parallel decoders, the straightforward way is to reduce the wires in the RNW unit. The *bit-serial decoder* [344] uses a single wire to send the message from a VN to a CN or vice versa. Thus, the connection between a VNP and a CNP consists of only two wires, instead of $\mathcal{Q}(V_{i,j})$ wires and $\mathcal{Q}(E_{i,j})$ wires for the V2C and C2V messages, respectively, where $\mathcal{Q}(*)$ represents quantization function. Such a decoder trades throughput for cost reduction since each VNP or CNP only becomes activated when all $\mathcal{Q}(E_{i,j})$ or $\mathcal{Q}(V_{i,j})$ bits of the message are received. Hence, the throughput reduction is related to the quantization level. The reduced quantization of messages leads to a reduced number of wire interconnects between VNPs and CNPs, however at a cost of degradation of the error correction capability.

2) Unrolled Fully Parallel: Unrolled fully parallel architecture is applicable for extremely high throughputs (*e.g.,* hundreds to thousands of Gb/s), which would be considered as the desired architecture for broadband data transmission in 6G. The basic idea behind this type of architecture is to introduce another level of parallelism by unrolling the decoding iterations. Consider the Tanner graph in Fig. 48 as an example. There are in total $N = 8$ VNs and $M = 4$ CNs, which can be referred to as 8 VNPs and 4 CNPs in a fully parallel decoder. In an unrolled decoder, the total number of VNPs and CNPs is $8I_{\max}$ and $4I_{\max}$, respectively. Fig. 52 shows an example of the unrolled LDPC decoder. The latency of the decoder is determined by the number of iterations, and hence, the number of pipeline stages (referring to the 'Pipe Reg' in Fig. 52), but the throughput is fixed by the cycle duration. While such a fully unrolled decoder architecture requires significant hardware resources, it also has very high throughput since one codeword can be decoded in each clock cycle. Thus, the hardware efficiency (*i.e.,* throughput per unit area) of the fully unrolled decoder often turns out to be significantly better than the hardware efficiency of the fully parallel (non-unrolled) decoders and partially parallel decoders [345]. However, the flexibility is limited similar to the fully parallel architecture. However, the unrolled architecture implies mainly local wires, which reduces the routing congestions.

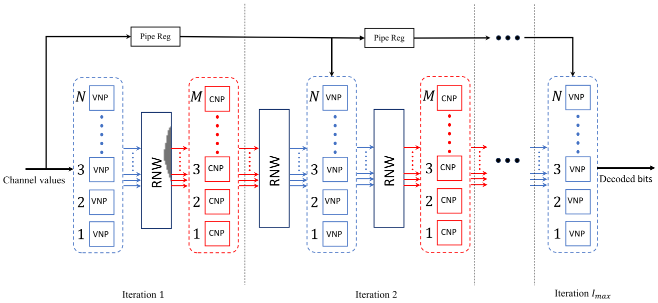


FIGURE 52. Unrolled LDPC decoder.

3) Partially Parallel: Another type of decoder architecture, namely *partially parallel* [346], [347], [348], [349], [350], has been considered to reduce the complexity and cost of the (unrolled) fully paralleled decoder. The main feature of such an architecture is to introduce some level of serialization of the CN and VN operations by processing only a subset of edges in parallel. This can be done in either a row-based or a column-based manner. The sequential processing of rows or columns allows layered decoding, i.e., taking advantage of intermediate node updates, which accelerates convergence and thus reduces the number of iterations. Fig. 53 shows an example of a partially parallel architecture for layered decoding of QC-LDPC codes. The key components included are:

Processing unit (PU): The PUs associated with each VN consist of the computations of:

- V2C messages of the current iteration: this is obtained by subtracting the C2V message of the previous iteration from the stored APP value, where

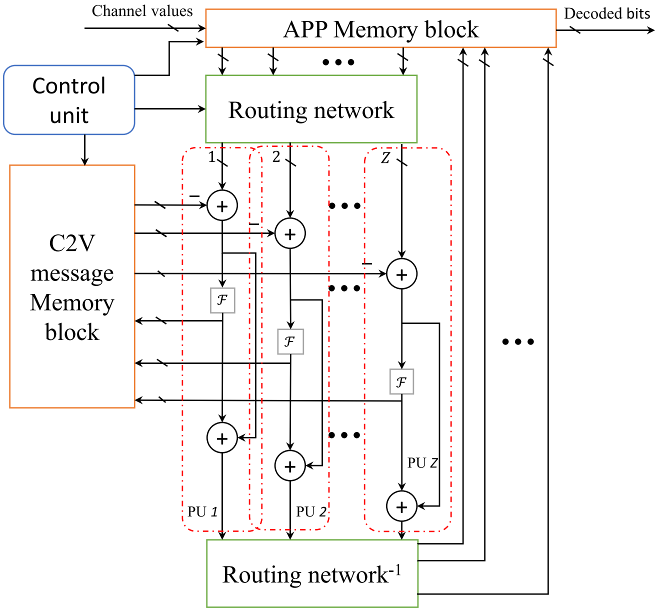


FIGURE 53. Pipelined QC-LDPC decoder.

$$\hat{r}_j^{-1} = r_j,$$

$$V_{i,j}^{(u)} = \hat{r}_j^{(u-1)} - E_{i,j}^{(u-1)}. \tag{150}$$

- C2V messages of the current iteration: this is obtained via the function $\mathcal{F}_2(*)$ with certain decoding algorithms described in Section VI-G,

$$E_{i,j}^{(u)} = \mathcal{F}_2\left(V_{i,j'}^{(u)}\right), j' \in \mathcal{B}(c_i), j' \neq j. \tag{151}$$

- Updated APP message of the current iteration by adding the C2V and V2C messages of the current iteration,

$$\hat{r}_j^{(u)} = E_{i,j}^{(u)} + V_{i,j}^{(u)}. \tag{152}$$

The number of PU is equal to the circulant size Z, and hence, for each layer (row) of the base matrix $\boldsymbol{H}_b$, d_c clock cycles are needed to load all d_c circulants of APP values to the PUs for pipeline processing. Let δ_{pipe} be the pipeline depth of PU, that is, the number of pipeline stages for an input APP message to complete all the calculation steps in the PU. Then the total number of cycles required to complete a layer of decoding is $d_c + \delta_{pipe}$. Let $\mathcal{E}(\boldsymbol{H}_b)$ be the total number of non-zero elements in the base matrix. The total number of clock cycles to complete a decoding iteration is $\mathcal{E}(\boldsymbol{H}_b) + \delta_{pipe}$.

Block RAM: Three blocks of memory are needed in this kind of architecture. The initial channel LLR values need to be stored in a block RAM, and this memory block will be constantly updated by new APP messages computed during decoding iterations. This is illustrated as the 'APP Memory block' in the Fig. 53. The size of this BRAM is $n_p \times Z\mathcal{Q}(r)$, where n_p and $Z\mathcal{Q}(r)$ are the depth and width of the memory block, respectively, $\mathcal{Q}(r)$ denotes the number of quantization bits to represent the initial channel LLR r. The second block RAM unit of size $\mathcal{E}(\boldsymbol{H}_b) \times Z\mathcal{Q}(E_{i,j})$ is needed to store the computed C2V messages as this message needs to be used in the computation of V2C message in the next iteration according to (150). An additional code description memory (usually a ROM) is needed to store the information of the parity-check matrix $\boldsymbol{H}$. This information includes the column indices of each row, the shift positions for each circulant block, as well as other useful information determined during the design stage.

Routing network: The routing network is needed for this type of architecture because each circulant block is loaded in the decoder sequentially, and the shifting value corresponding to each circulant block varies. Furthermore, the updated APP message in one decoding iteration needs to be routed in the reverse direction before being stored back in the APP memory for use in the next iteration. In this case, the routing network must be reconfigurable to handle all possible shifting values smaller than the lifting size Z, while also being capable of both forward and reverse routing networks. Examples of well-known routing networks are Benes network [351], Oh-Parhi network (OPN) [352] and QC-LDPC shift network (QSN) [353].

3) COMPARISON BETWEEN DECODERS

The channel codec implementation is critical in terms of power consumption and silicon area, particularly at data rates approaching 1 Tb/s, which is one of the important KPIs foreseen in 6G. The recent results of the implementation of the LDPC decoder are briefly reviewed in Table 18. Note that for a comparison the results collected in the table have a similar code rate. Results of other code rates can be found in the referenced works. In addition, more results in LDPC decoder implementations can be found in [354].

To achieve ultra-high throughput of 1 TB/s, there are many possible optimization directions: increasing the decoding parallelism, reducing the decoding latency, increasing the clock speed, and increasing the code length. From the table, it can be seen that iteration unrolled fully parallel architecture is promising, however at the expense of I_{max} copies of the circuit. Alternatively, row-layered scheduling with multi-core instantiated architecture could also approach a throughput of 1 TB/s. For instance, the 1 Tb/s throughput achieved in [355] instantiated 12 single core and is configured for maximum 4 iterations. Moreover, extended pipelined design, by inserting additional flip-flops (or register units) in between each operation during the decoding, significantly boosts the operating frequency, so that the overall throughput is improved. For instance, the implementations in [356] and [357] add a pipeline with seven stages for each iteration to increase the frequency up to 1.88 GHz for the (648, 540) LDPC code in IEEE 802.11n, and up to 1.511GHz for the (1944, 1620) LDPC code in IEEE 802.11n. The resulting peak throughput is 1.218Gb/s and 2.937Gb/s, respectively. The throughput of the (1944, 1620) code is 3 times the throughput of the (648, 540) code due to the fact that the information length is 3 times different. Hence,

TABLE 18. Comparison of ASIC implementation of LDPC decoders.

Ref.	[345]	[358]	[359]	[355]	[356]	[357]	[360]
Technology	65nm	28nm	28nm	16nm	28nm	28nm	65nm
Algorithm	Min-sum	Adaptive degeneration	Finite alphabet	Min-sum	Min-sum	Min-sum	Offset min-sum
Architecture	Unrolled fully parallel	Fully parallel	Unrolled fully parallel	Layered	Unrolled fully parallel	Unrolled fully parallel	Layered
LDPC code (N, K)	$(672, 546)$	$(60000, 53570)$	$(2048, 1723)$	$(1027, 856)$	IEEE 802,11n $(648, 540)$	IEEE 802.11n $(1944, 1620)$	3GPP 5G NR $(10368, 8448)$
Code rate R	0.8125	0.88	0.84	0.833	0.833	0.833	0.88
Iterations	9	49	5	4	5	5	3
SNR @BER=10^{-7} (dB)	–	4.55	4.95	5.3	5	5.05	4.25***
Throughput* (Gb/s)	131(161)	400(455)	494(588)	833(1000)	1015(1218)	2446(2937)	19.2(21.8)
Clock speed f_{max}	257MHz	373MHz	862MHz	1GHz	1.88GHz	1.511GHz	500MHz
Latency	105ns	134ns	69.6ns	38ns	19.68ns	27.8ns	–
Core Area mm^2	12.09	7.46	16.2	2.24	5.49	16.46	5.74
Area efficiency** (Gb/s/mm^2)	13Gb/s/mm^2	61Gb/s/mm^2	36Gb/s/mm^2	446Gb/s/mm^2	222Gb/s/mm^2	178Gb/s/mm^2	3.8Gb/s/mm^2
Core power (mW)	–	624	13350	3.19	–	–	413
Energy efficiency (pJ/bit)	3.61	1.56	22.7	3.82@6dB	12.74@4dB	10.31@4dB	–

∗: Here, the information throughput is shown. The corresponding coded throughput is given inside the bracket.

∗∗: Area efficiency is calculated based on the coded throughput.

∗ ∗ ∗: Estimated value at BER = 10^{-6}.

increasing the information block size would also increase the throughput for the same decoding architecture and configurations.

M. FUTURE DIRECTIONS

1) NEW CODE STRUCTURES

For future-generation wireless communication systems, such as B5G and 6G, the required transmission data rate will be extremely high due to demanding, data-hungry applications and technologies such as streaming multimedia, augmented reality (AR), virtual reality (VR), the metaverse and more. For LDPC codes, it may be necessary to introduce a lifting size larger than 384, which is currently the largest lifting size used in the 5G NR, while the protograph remains unchanged because the 5G NR LDPC code's performance has already been pushed to the limit using 16nm and 7nm technologies [361]. As shown in (149) the throughput of an LDPC decoder is directly proportional to the size of the information block; increasing the lifting size will enhance the decoder's throughput. Hence, new base matrices with optimized shift values need to be designed based on the new or existing 5G NR PBRL LDPC code structure. Furthermore, as the performance of 5G NR LDPC codes has been pushed to the limit, trade-offs need to be made between parallelization, pipelining, iterations, and unrolling, while linking them with the decoder architecture.

2) SPECTRUM EFFICIENT LDPC CODING SCHEMES

Recent standards exploit the collaborative use of re-transmission protocol and channel coding schemes for lowering end-to-end delay essential for high-speed applications. However, this requires the need for additional bits in sending the acknowledgment of the received ensemble data. Further, each transportation data block consists of several code blocks. If any of the code blocks get corrupted, the retransmissions can improve the spectrum efficiency. As 5G trends focus on increasing the data rate a hundred times, there are more code blocks in one transport block. Several techniques have been implemented to overcome this challenge that re-transmit selective code blocks of corrupted data but at the cost of increased overhead due to the additional cyclic redundancy check (CRC) bits needed for each code block. The work [268] proposed the PIC method, which involves the use of some interlinked CBs of encoded LDPC codewords which improves the problem of spectrum efficiency and reduces delay too. Also, Non-Binary-LDPC Codes (NBLC) [362], [363], [364] have been proposed to escalate spectrum efficiency. Furthermore, due to the superior error rate performance and simple code construction approach, SC-LDPC codes with windowed decoder would be an option for a spectrum-efficient channel coding scheme. It is known that the maximum size of a transportation block in the 5G NR standard is over 1.2 million bits, and is expected to be larger in future communication proposals. The rate-loss of SC-LDPC codes due to the termination length L and coupling memory m_s can be minimized with parameters of SC-LDPC codes, such as L and m_s, properly chosen. However, one of the key challenges for SC-LDPC codes to be considered in practice is the efficient encoding methods, which is one of the research directions that has very limited reports.

3) UNIFIED RECONFIGURABLE DECODERS

As one of the key technologies for 6G, the unified design of channel coding schemes at the circuit level of different types of codes is important [365], such as Turbo/LDPC decoders [366] and LDPC/Polar decoders [367]. The unified decoder architecture would significantly benefit the decoder implementation in chipset design. In [368], a deep-learning-based unified polar-LDPC decoder for 5G communication systems is proposed. Moreover, the authors in [367] proposed a joint LDPC/polar decoding algorithm based on the BP decoding algorithm. Based on the proposed decoding algorithm, a reconfigurable decoding architecture is proposed for standard-compatible decoding of LDPC codes and polar codes. Furthermore, it is known that the biggest challenge in the design of a unified BP decoder for LDPC and Polar codes is the significant performance disparity between Polar codes decoded using belief propagation (BP) and those decoded under the successive cancellation list (SCL) and CRC-aided SCL decoders. The performance of Polar codes under BP decoding is notably worse which is one of the potential barriers that need to be overcome in the design of a unified Polar-LDPC decoder.

VII. POLAR CODES

Since their introduction in 2008, polar codes have garnered significant attention. They were the first family of binary linear codes with explicit construction that provably achieve symmetric capacity of arbitrary binary-input discrete memoryless channels (B-DMCs) [369]. Their construction is explicit as implied by the transformations that lead to channel polarization (CP). Due to their structural similarity with binary Reed-Muller codes, many concepts and decoding schemes can be borrowed from RM codes and adopted for use with polar codes.

In this section, while briefly reviewing the polar coding, we discuss promising advances that could potentially reduce the polar code decoding latency or improve their reliability. Toward this goal, we first describe the basis of polar coding, which is channel polarization, the origin of the idea, the properties of polar codes, and their relation with Reed-Muller codes. Then, we review the code construction methods and commonly used code concatenation schemes, such as CRC-polar coding. This is followed by the known decoding algorithms used for polar codes and their variants along with their hardware implementations. This section is completed by discussing the puncturing and shortening techniques used for rate-compatible polar codes for practical applications and polar-coded modulation schemes for high-order modulation. We end this section with a comparison of the performance and complexity of various polar coding and decoding schemes.

A. CHANNEL POLARIZATION EFFECT

The polarization effect is realized by the transformation of $N = 2^n, n \geqslant 1$ identical and independent copies of a physical/raw B-DMC $W{:}\mathcal{X} \to \mathcal{Y}$ into a set of N correlated virtual channels or a vector channel with N sub-channels as

$$W_N^{(i)} : \mathcal{X} \to \mathcal{Y}^N \times \mathcal{X}^{i-1}, \quad i \in [1, N], \tag{153}$$

which are either better or worse than the original channel W. As N grows large, the channels perceived by individual bits start to polarize, that is, they approach the status of either a perfect channel or an unreliable channel. Note that the polarization is not restricted to a particular transformation, but is considered a general phenomenon. A single-step (local) channel transform is performed for two raw channels based on the linear map $\boldsymbol{G}_2 \triangleq \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$, which is equivalently represented by two channel transformations of $W \boxed{\ast} W$ and $W \circledast W$ resulting in $W \mapsto (W^-, W^+)$. Hence, we denote the basic channel transformation by

$$W^0 = W^- = W \boxed{\ast} W, \quad W^1 = W^+ = W \circledast W. \tag{154}$$

The output alphabet of $W \boxed{\ast} W$ is $\mathcal{Y}^2$, the output alphabet of $W \circledast W$ is $\mathcal{Y}^2 \times \mathcal{X}$, and their transition probabilities are given by

$$W^0(y_1, y_2 \mid u_1) \triangleq \frac{1}{2} \sum_{u_2 \in \mathcal{X}} W(y_1 \mid u_1 \oplus u_2) W(y_2 \mid u_2) \tag{155}$$

and

$$W^1(y_1, y_2, u_1 \mid u_2) \triangleq \frac{1}{2} W(y_1 \mid u_1 \oplus u_2) W(y_2 \mid u_2). \tag{156}$$

The channel transformation can then be recursively expanded to any power of two $N = 2^n$ channels for an integer $n \geqslant 1$. The corresponding transformation matrix can be obtained by the Kronecker power of $\boldsymbol{G}_2$ as $\boldsymbol{G}_N = \boldsymbol{G}_2^{\otimes n}$. Starting from $N = 2^n$ raw channels $W = W_1^{(1)}$, then in each channel transformation stage $j \in [1, n]$, every two sub-channels $W_{2j-1}^{(i)}$ are transformed into two child sub-channels as

$$\begin{aligned} W_{2j}^{(2i-1)} &= W_{2j-1}^{(i)} \boxed{\ast} W_{2j-1}^{(i)}, \\ W_{2j}^{(2i)} &= W_{2j-1}^{(i)} \circledast W_{2j-1}^{(i)}. \end{aligned} \tag{157}$$

Fig. 54 demonstrates the transformation of eight raw channels W to sub-channels $\{W_8^{(i)}{:}i \in [1, 8]\}$ from right to left. Let us denote the binary expansion of integer $i - 1$, for $i \in [1, N]$ by $(b_{n-1}, \ldots, b_1, b_0)$, for $b_k \in \{0, 1\}$ with the most significant bit on the left. Then, the sub-channels can be defined as

$$W_N^{(i)} = \left(\left(W^{b_0} \right)^{b_1} \ldots \right)^{b_{n-1}}, i \in [1, N] \tag{158}$$

where $(\cdot)^{b_k}$ is obtained from equations (155) and (156).

To proceed further, let us define the notions of symmetric capacity and the Bhattacharyya parameter as rate and reliability parameters. For a B-DMC $W : \{0, 1\} \to \mathcal{Y}$, the channel transition probabilities are denoted by $W(y \mid x)$, where $y \in \mathcal{Y}, x \in \mathcal{X} = \{0, 1\}$. The channel W is said to be symmetric if for every $y \in \mathcal{Y}$ and a permutation π where $\pi = \pi^{-1}$, we have $W(y \mid 1) = W(\pi(y) \mid 0)$. Then, the

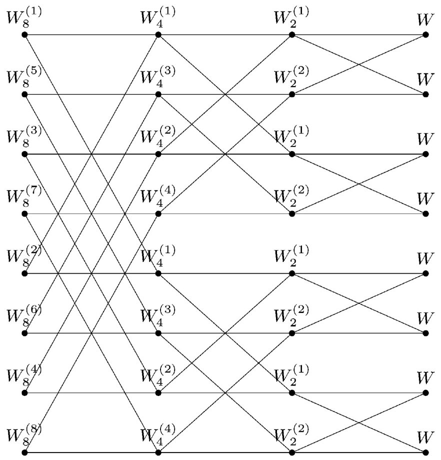


FIGURE 54. The channel transformation for $N = 8$ [369].

symmetric capacity and the Bhattacharyya parameter of W are defined as

$$I(W) \triangleq \sum_{y\in\mathcal{Y}}\sum_{x\in\mathcal{X}} \frac{1}{2} W(y \mid x) \ \log \frac{W(y \mid x)}{\frac{1}{2}W(y \mid 0) + \frac{1}{2}W(y \mid 1)},$$

and

$$Z(W) \triangleq \sum_{y\in\mathcal{Y}} \sqrt{W(y \mid 0) \ W(y \mid 1)}.$$

For the basic transformation where there exist only two channels, as in (155) and (156), the capacities of these symmetric channels are related by

$$I\left(W^{-}\right) + I\left(W^{+}\right) = 2I(W). \tag{159}$$

As can be seen, the capacities of the raw channels are preserved in the channel transformation. This can be expanded to any N channels of power of two as follows. Given the input vector u_1^N, the transformed vector $x_1^N = u_1^N \boldsymbol{G}_N$ to be transmitted through the raw channel, and the corresponding vector y_1^N representing the output of the raw channel through N uses, the transition probabilities from the input of the synthesized vector channel W_N to the output of the underlying N raw channels W^N are related by $W_N(y_1^N|u_1^N) = W^N(y_1^N|u_1^N\boldsymbol{G}_N)$. From $W_N(y_1^N|u_1^N)$, the transition probability of the bit-channel (a.k.a sub-channel or synthetic channel) $i \in [1, N]$ is implicitly defined as

$$W_N^{(i)}\left(y_1^N, u_1^{i-1}|u_i\right) = \sum_{u_{i+1}^N\in\{0,1\}^{N-i}} \frac{1}{2^{N-1}} W_N\left(y_1^N|u_1^N\right). \tag{160}$$

Let us now investigate how the rate (the sub-channel capacity) and the reliability change as a result of transformation. For any $i \in [1, N]$, $N = 2^n, n \geqslant 0$ and B-DMC W, the transformation $(W_N^{(i)}, W_N^{(i)}) \mapsto (W_{2N}^{(2i-1)}, W_{2N}^{(2i)})$ is rate-preserving and reliability-improving as [369, Proposition 7]

$$I\left(W_{2N}^{(2i-1)}\right) + I\left(W_{2N}^{(2i)}\right) = 2I\left(W_N^{(i)}\right),$$
$$Z\left(W_{2N}^{(2i-1)}\right) + Z\left(W_{2N}^{(2i)}\right) \leqslant 2Z\left(W_N^{(i)}\right),$$

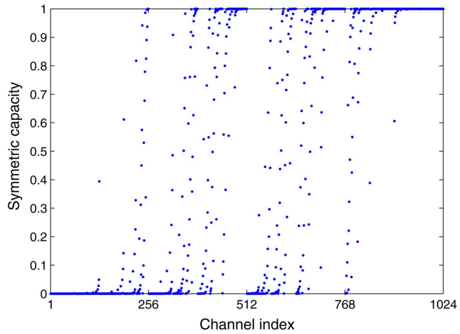


FIGURE 55. The channel transformation for $N = 2^{10}$ [369].

where the Bhattachariya parameter for sub-channel $W_N^{(i)}$ given the channel observation y_1^N is

$$Z\left(W_N^{(i)}\right) \triangleq \sum_{y_1^N, u_1^{i-1}} \sqrt{W_N^{(i)}\left(y_1^N, u_1^{i-1} \mid 0\right) W_N^{(i)}\left(y_1^N, u_1^{i-1} \mid 1\right)}. \tag{161}$$

If $I(W_N^{(i)}) > I(W_N^{(j)})$, or equivalently $Z(W_N^{(i)}) < Z(W_N^{(j)})$, it is said that the sub-channel $W_N^{(i)}$ is more reliable than $W_N^{(j)}$, denoted as $W_N^{(i)} \succ W_N^{(j)}$, or simply $i \succ j$. The channel polarization theorem [369] states that the mutual information of the i-th sub-channel, $I(W_N^{(i)})$ for every $i \in [1, N]$, converges to 0 or 1 as N approaches infinity. That is, the channels obtained after n levels of transformation by (154) are either almost perfect, $I(W_{2^n}^{(i)}) \geqslant 1 - \delta$ where $i \in [0, 2^n - 1]$ and $\delta > 0$, or almost unreliable, $I(W_{2^n}^{(i)}) \leqslant \delta$. Note that the fraction of channels with $I(W_{2^n}^{(i)}) \in (\delta, 1-\delta)$ diminishes:

$$\lim_{n\to\infty} \frac{\left|\left\{i : I\left(W_{2^n}^{(i)}\right) \in (\delta, 1-\delta)\right\}\right|}{2^n} = 0.$$

Given (159), by induction to

$$\sum_{i\in[0,2^n-1]} I\left(W_{2^n}^{(i)}\right) = 2^n I(W),$$

we can conclude that the fraction of almost perfect channels approaches the symmetric capacity. Therefore, we would have $N \cdot I(W)$ perfect sub-channels to use for K information bits. Conversely, the fraction of indices $i \in [1, N]$ for which the sub-channels become extremely bad sub-channels approaches $(1 - I(W))$. Fig. 55 illustrates the polarization effect for the binary erasure channel (BEC) W where the erasure probability is 0.5. In the case of BEC W, the rate $I(W_N^{(i)})$ can be computed using the following recursive relations:

$$I\left(W_N^{(2i-1)}\right) = I\left(W_{N/2}^{(i)}\right)^2$$
$$I\left(W_N^{(2i)}\right) = 2I\left(W_{N/2}^{(i)}\right) - I\left(W_{N/2}^{(i)}\right)^2$$

Polar codes with rate $R = K/N$ are constructed by selecting K indices with the highest $I(W_N^{(i)})$ for $i \in [1, N]$.

These are dedicated to information bits and called the *non-frozen set*, $\mathcal{A}$. The input bits corresponding to *frozen set* $\mathcal{A}^c$ are usually set to zero. We further discuss the construction of polar codes in the next section. Note that this construction method is optimal for the original decoding algorithm proposed for polar codes called successive cancellation (SC) decoding.

B. ORIGIN OF POLAR CODES

The idea of building synthetic channels [370] originated from the concatenated schemes for convolutional codes under sequential decoding by Massey [371] and Pinsker [372] in order to boost the cutoff rate. The cutoff rate is said to be "boosted" when the sum of the cutoff rates of the synthetic channels is greater than the sum of the cutoff rates of the raw channels. The key idea to achieve that was to correlate the independent copies of raw channels through concatenation. In Pinsker's scheme, the identical outer convolutional transforms were employed while the inner block code (with length N) was suggested to be chosen at random. This requires maximum likelihood (ML) decoding with prohibitive complexity. Different from Pinsker's scheme, in multi-level coding and multi-stage decoding (MLC/MSD), originally proposed in [373] as an efficient coded-modulation technique, N convolutional codes at different rates $\{R_i\}$ are used, which consequently require a chain of N outer convolutional decoders. In contrast to the aforementioned schemes, polar coding was originally designed as a low-complexity recursive channel combining and splitting operations, where the polarization effect constrains the rates R_i to 0 or 1. This method of building synthetic channels turned out to be so effective that no outer code was employed to achieve the original aim of boosting the cutoff rate.

C. PROPERTIES OF POLAR CODES

The generator matrix of the polar code (K, N) is a $K \times N$ submatrix of polar transform $\boldsymbol{G}_N = \boldsymbol{B}\boldsymbol{G}_2^{\otimes n} = [\boldsymbol{g}_1 \dots \boldsymbol{g}_N]^T$ consisting of rows $\boldsymbol{g}_i$ with indices $i \in \mathcal{A}$, denoted by $\boldsymbol{G}$, where $\boldsymbol{B}$ is a bit-reversal permutation matrix defined in [369]. The matrix $\boldsymbol{B}$ is symmetric, that is, the (i, j)th entry is equal to the (j, i)th entry for all i and j. Hence, $\boldsymbol{B}^T = \boldsymbol{B}$. As the inverse of a symmetric matrix is the matrix that reverses the permutation that the original matrix performs, we have $\boldsymbol{B}^{-1} = \boldsymbol{B}$. On the other hand, $\boldsymbol{G}_2^{\otimes n}$ is invariant under bit-reversal, we have $\boldsymbol{G}_N = \boldsymbol{B}^T\boldsymbol{G}_2^{\otimes n}\boldsymbol{B}$. Since $\boldsymbol{B}^T = \boldsymbol{B}^{-1}$, hence $\boldsymbol{B}_N$ commutes with the bit-reversal operator, that is, $\boldsymbol{B}\boldsymbol{G}_2^{\otimes n} = \boldsymbol{G}_2^{\otimes n}\boldsymbol{B}$.

As $\boldsymbol{G}_2^{\otimes n}$ is a lower-triangular matrix with 1s on the diagonal, it is invertible. In fact, the inverse of $\boldsymbol{G}_2^{\otimes n}$ is itself [369]. Given $\boldsymbol{G}_2^{-1} = \boldsymbol{G}_2$ and $\boldsymbol{B}$ commutes $\boldsymbol{G}_2^{\otimes n}$, we have

$$\boldsymbol{G}_N^{-1} = \left(\boldsymbol{G}_2^{\otimes n}\right)^{-1}\boldsymbol{B}^{-1} = \left(\boldsymbol{G}_2^{-1}\right)^{\otimes n}\boldsymbol{B} = \boldsymbol{B}\boldsymbol{G}_2 = \boldsymbol{G}_N. \quad (162)$$

Let us denote the elements of the row $\boldsymbol{g}_i$ of the matrix $\boldsymbol{G}_N$ by $\{g_{i,j}\}, j \in [N]$. Then, the submatrix $\boldsymbol{G}_{\mathcal{A}}$ includes the rows $\boldsymbol{g}_i, i \in \mathcal{A}$ while the submatrix $\boldsymbol{G}_{\mathcal{AB}}$ consists of $g_{i,j}, i \in \mathcal{A}, j \in \mathcal{B}$. Also, let $\boldsymbol{u}_{\mathcal{A}}$ denote the subvector of $\boldsymbol{u}$ consists of $u_i, i \in \mathcal{A}$.

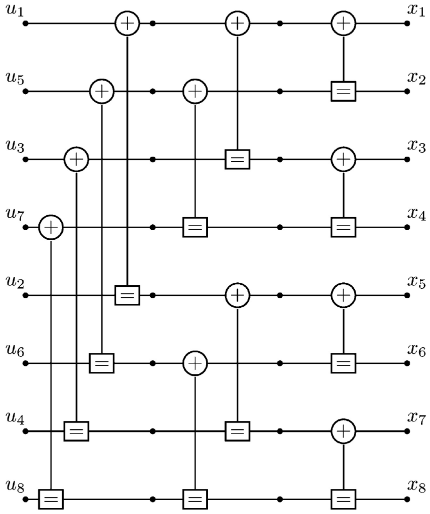


FIGURE 56. Factor graph for $N = 8$ [369].

For encoding, the information bits $\boldsymbol{d} = [d_1 \dots d_K]$ are inserted into the input vector $\boldsymbol{u} = [u_1 \dots u_N]$ at the coordinates $i \in \mathcal{A}$, while $u_i = 0$ for $i \in \mathcal{A}^c$, that is, the bad subchannels are used for the transmission of known values (by default 0). Therefore, we can encode an information sequence $\boldsymbol{d} = \boldsymbol{u}_{\mathcal{A}}$ using $\boldsymbol{d}\boldsymbol{G}_{\mathcal{A}} = \boldsymbol{u}\boldsymbol{G}_N = \boldsymbol{x}$.

The parity check matrix $\boldsymbol{H}$ of a polar code is characterized as follows [374]: Given $\boldsymbol{G}_N^{-1} = \boldsymbol{G}_N$ and $\boldsymbol{x} = \boldsymbol{u}\boldsymbol{G}_N$, we have $\boldsymbol{u} = \boldsymbol{x}\boldsymbol{G}_N$. To impose parity check constraints $\boldsymbol{x} \cdot \boldsymbol{H}^T = \boldsymbol{0}$ on $\boldsymbol{x}$, it suffices to select the columns of $\boldsymbol{G}_N$ corresponding to $u_j = 0$. Therefore, the parity check matrix $\boldsymbol{H}$ of polar codes is a submatrix of $\boldsymbol{G}_N$ consisting of columns $j \in \mathcal{A}^c$.

1) SYSTEMATIC POLAR CODES

A systematic code allows the original message to be recovered directly from the received codeword. That is, the systematic coding results in $\boldsymbol{x}_{\mathcal{A}} = \boldsymbol{u}_{\mathcal{A}}$. To achieve this, we obtain coded bits $\boldsymbol{x} = \{\boldsymbol{x}_{\mathcal{A}}, \boldsymbol{x}_{\mathcal{A}^c}\}$ using submatrix $\boldsymbol{G}_{\mathcal{AA}}$ as

$$\boldsymbol{x}_{\mathcal{A}} = \boldsymbol{u}_{\mathcal{A}}\boldsymbol{G}_{\mathcal{AA}}^{-1}\boldsymbol{G}_{\mathcal{AA}} = \boldsymbol{u}_{\mathcal{A}}\boldsymbol{I}_K = \boldsymbol{u}_{\mathcal{A}}, \quad (163)$$

while the parity bits $\boldsymbol{x}_{\mathcal{A}^c}$ are obtained by

$$\boldsymbol{x}_{\mathcal{A}^c} = \boldsymbol{u}_{\mathcal{A}}\boldsymbol{G}_{\mathcal{AA}}^{-1}\boldsymbol{G}_{\mathcal{AA}^c} = \boldsymbol{u}_{\mathcal{A}}\boldsymbol{P}, \quad (164)$$

where $\boldsymbol{P} = \boldsymbol{G}_{\mathcal{AA}}^{-1}\boldsymbol{G}_{\mathcal{AA}^c}$.

2) MINIMUM DISTANCE OF POLAR CODES

The minimum distance d of polar codes is equal to the minimum weight of the rows of the generator matrix, that is, $d = \min\{w(\boldsymbol{g}_i), i \in \mathcal{A}\}$. Fig. 56 illustrates the factor graph representation of $\boldsymbol{G}_2^{\otimes 3}$ proposed in [375] for the BP decoding of Reed-Muller codes. Observe that if we pass the input vector $\boldsymbol{u} = [u_1 \dots u_8]$ from the left-hand side through the factor graph consisting of an addition operator

$\oplus$ in the Galois field F_2 (we call it f-node) and the passing operator $=$ (we call it g-node), we get the coded sequence $\boldsymbol{x} = [x_1 \; \dots \; x_8]$ on the right-hand side.

We can divide all codewords of a polar code $\mathcal{C}(\mathcal{A})$ (excluding the all-zero codeword) into cosets $\mathcal{C}_i(\mathcal{A})$ as

$$\mathcal{C}_i(\mathcal{A}) \triangleq \left\{ \boldsymbol{g}_i \oplus \bigoplus_{h \in \mathcal{H}_i} \boldsymbol{g}_h \;:\; \mathcal{H}_i \subseteq \mathcal{A} \backslash [0, i] \right\} \subseteq \mathcal{C}(\mathcal{A}). \quad (165)$$

Then, the minimum distance of the code $\mathcal{C}(\mathcal{A})$, is $d_{\min} = \min_{i \in \mathcal{A}} \mathrm{w}(\boldsymbol{g}_i)$ [376], [377], [378, Lemma 3].

3) FORMATION OF MIN-WEIGHT CODEWORDS

The weight of any codeword in the coset $\mathcal{C}_i(\mathcal{A})$ follows [377, Corollary 3]

$$\mathrm{w}\left(\boldsymbol{g}_i \oplus \bigoplus_{j \in \mathcal{H}_i} \boldsymbol{g}_h \right) \geqslant \mathrm{w}(\boldsymbol{g}_i), \quad (166)$$

where $\mathcal{H}_i \subseteq [i+1, N-1]$. Then, the minimum weight codewords in each coset $\mathcal{C}_i$ is decomposed into $\boldsymbol{G}_N$-rows as [377, Lemma 6]

$$\mathrm{w}\left(\boldsymbol{g}_i \oplus \bigoplus_{j \in \mathcal{J}} \boldsymbol{g}_j \oplus \bigoplus_{m \in \mathcal{M}(\mathcal{J})} \boldsymbol{g}_m \right) = w_{\min}, \quad (167)$$

where $\boldsymbol{g}_i$ is the leading row, $\boldsymbol{g}_j, j \in \mathcal{J}$ are the core rows, and $\boldsymbol{g}_m, m \in \mathcal{M}(\mathcal{J})$ are the balancing rows. The indices of the core rows in $\mathcal{J}$ are a subset of the set $\mathcal{K}_i \triangleq \{j \in \mathcal{I} \backslash [0, i] \colon |\mathrm{supp}(j) \backslash \mathrm{supp}(i)| = 1\}$ [377, Defnition 4, Lemma 5]. As a result, every subset of $\mathcal{K}_i$ along with the other rows in (167) forms a codeword of minimum weight. The number of subsets of $\mathcal{K}_i$ is given by $2^{|\mathcal{K}_i|}$. Given $\mathcal{B} \triangleq \{i \in \mathcal{I} : \mathrm{w}(\boldsymbol{g}_i) = d_{\min}\}$, the total number of minimum weight codewords of the polar code will be $\sum_{i \in \mathcal{B}} 2^{|\mathcal{K}_i|}$. The set $M(\mathcal{J})$ is a function of the set $\mathcal{J}$ and every $m \in M(\mathcal{J})$ has the property $|\mathrm{supp}(m) \backslash \mathrm{supp}(i)| > 1$ (see [377, eqs. (33), (34)] for a detailed definition of $M(\mathcal{J})$). The number of codewords with minimum weight generated by the leading row $\boldsymbol{g}_i$ is denoted by $A_{i, d_{\min}}$.

The elements of the set $\mathcal{A}$ can be represented by monomials depending on the binary representation of the indices. For example, given the index $i = i_0, i_1, \dots, i_{n-1} = 10101$, the corresponding monomial f is a product of variables $f = \prod_{j=0}^{m-1} x_j^{i_j} = x_0 x_2 x_4$ of degree 3. Now, we collect all these monomials associated with the elements of the set $\mathcal{A}$ of a certain polar code and form the set $\mathcal{I}$, then we can call it the monomial code.

In [379] it was shown that there is a decreasing order relation between the elements of the set $\mathcal{I}$, as described in [380, Definition 3], equivalent to the partial order. The decreasing property induces new algebraic properties that can give a slightly better understanding of the algebraic structure of polar codes.

4) RATE OF POLARIZATION

For matrix $\boldsymbol{G}_2$, block-length $N = 2^n$, and rate $R < I(W)$, the probability of block error $P_e(N, R)$ for polar coding and successive cancellation decoding can be bounded as $P_e(N, R) \leqslant 2^{-N^{\beta}}$, or $P_e(N, R) = O(2^{-N^{\beta}})$, for any $\beta < \frac{1}{2}$ [381]. The parameter $\frac{1}{2}$ is called the *exponent* of G_2, denoted by $E(\boldsymbol{G}_2) = \frac{1}{2}$ and is a performance measure. This implies that when n is sufficiently large, there exists a set $\mathcal{A}$ of size $N \cdot R$ such that $\sum_{i \in \mathcal{A}} Z(W_{2^n}^{(i)}) \leqslant 2^{-N^{\beta}}$. Let us define the *partial distances* $D_i, i = 1, \dots, \ell$ of an $\ell \times \ell$ matrix $G = [g_1 \dots, g_\ell]^T$ as [382]

$$\begin{aligned} D_i &\triangleq d_H(g_i, \langle g_{i+1}, \dots, g_\ell \rangle), \quad i = 1, \dots, \ell - 1 \\ D_\ell &\triangleq d_H(g_\ell, 0), \end{aligned}$$

where d_H and $\langle \cdot \rangle$ denote the minimum Hamming distance and row span, then the *rate of polarization* $\mathrm{E}(G)$ is

$$\mathrm{E}(G) = \frac{1}{\ell} \sum_{i=1}^{\ell} \log_\ell D_i.$$

The partial distances of a polarizing matrix constructed from the Kronecker product can be expressed as a product of those of its component matrices [383]. As a result, the exponent of the polarizing matrix is a weighted sum of the exponents of its component matrices.

The rate of polarization of an $\ell \times \ell$ kernel matrix $\boldsymbol{G}$ indicates how fast the error probability P_e decays in the code length N assuming that the rate R is fixed. Therefore, this rate can be used to characterize asymptotically the performance of polar codes based on underlying kernels. It was shown in [382] that by constructing polar codes based on larger matrices $\boldsymbol{G}$, the exponent can improve, i.e., becomes larger. We will discuss this in Section I.

D. REED-MULLER CODES VERSUS POLAR CODES

A binary Reed-Muller (RM) code of length $N = 2^n$, order r, dimension $K = \sum_{i=0}^{r} \binom{n}{i}$ and the minimum distance $d = 2^{n-r}$ can be represented based on the transformation matrix $\boldsymbol{G}_N$ where the generator matrix is a $K \times N$ submatrix of $\boldsymbol{G}_N = [\boldsymbol{g}_1 \; \dots \; \boldsymbol{g}_N]^T$ consisting of rows $\boldsymbol{g}_i$ with indices $i \in \{j \in [N] \mid w(\boldsymbol{g}_j) \geqslant 2^{n-r}\}$. We refer to this approach of selecting rows based on their weights as *RM rule*. As can be seen, the differences between RM codes and polar codes are mainly in the choice of rows of the transformation matrix $\boldsymbol{G}_N$ and the flexibility in the choice of the code dimension K. Furthermore, the decreasing property described in Section VII-C also applies to RM codes. Therefore, this family of monomial codes is called decreasing monomial codes.

From a decoding perspective, successive cancellation decoding (see Section VII-K1) closely resembles recursive decoding of RM codes [384].

E. CODE CONSTRUCTION

In this section, we discuss the main approaches to designing polar codes. We focus mainly on the reliability-based construction, which is optimal for SC decoding. We also consider error-coefficient, the number approach for ML and near-ML decoding algorithms.

The construction (a.k.a. rate profile) defines the set $\mathcal{A} \subset \{1, 2, \ldots, N\}$ of rows of $\boldsymbol{G}_N$ corresponding to the K best sub-channels $W_N^{(i)}$ whose mutual information $\{I(W_N^{(i)}), i \in \mathcal{A}\}$ is the largest among $\{I(W_N^{(i)}), i \in [N]\}$. We call this approach as *Arıkan rule*, which is different from the RM rule used in the construction of RM codes. While the construction of polar codes is explicitly defined, the exact evaluation of $I(W_N^{(i)})$ for an additive white Gaussian noise channel is intractable, as it depends on the calculation of the parameters of channels whose output alphabets grow exponentially in code length.

As discussed in Section VII-A, the seminal work on polar codes [369] proposed using the Bhattacharyya parameter $Z(W_N^{(i)})$ as a measure of the probability of error (a reliability metric) over the binary erasure channel (BEC) and then choosing K sub-channels with the smallest $Z(W_N^{(i)}), i \in [N]$. The block error event, $\mathcal{E}$, of the code resulting from the set $\mathcal{A}$ under SC decoding is a union over $\mathcal{A}$ of the events $\mathcal{B}_i$ in which the first bit error occurs at the i-th bit, expressed as $\mathcal{E} = \bigcup_{i\in\mathcal{A}} \mathcal{B}_i$. The set is defined as [369]

$$\begin{aligned}\mathcal{B}_i &= \left\{\left(u_1^N, y_1^N\right) : \hat{u}_1^{i-1} = u_1^{i-1}, \hat{u}_i\left(y_1^N, u_1^{i-1}\right) \neq u_i\right\}\\ &\subseteq \left\{\left(u_1^N, y_1^N\right) : \hat{U}_i\left(y_1^N, u_1^{i-1}\right) \neq u_i\right\} \triangleq \mathcal{A}_i. \qquad (168)\end{aligned}$$

Then, the block error probability can be upper bounded by

$$P(\mathcal{E}) = \sum_{i\in\mathcal{A}} P(\mathcal{B}_i) \leqslant \sum_{i\in\mathcal{A}} P(\mathcal{A}_i) \leqslant \sum_{i\in\mathcal{A}} Z\left(W_N^{(i)}\right). \qquad (169)$$

1) DENSITY EVOLUTION

In [385], $P(\mathcal{A}_i)$ was considered as decoding error probabilities in belief propagation (BP) decoding on the tree graph corresponding to the ith bit while eliminating other edges of the Tanner graph. The root and leaves of the tree correspond to the variables nodes of u_i and y_1^N, respectively. This helps to evaluate the probability of error at the root node using *density evolution* as a known approach for LDPC codes [386].

The log-likelihood ratio (LLR) of the i-th bit, defined as

$$L_N^{(i)}\left(y_1^N, \hat{u}_i^{i-1}\right) \triangleq \log \frac{W_N^{(i)}\left(y_1^N, \hat{u}_1^{i-1} \mid 0\right)}{W_N^{(i)}\left(y_1^N, \hat{u}_1^{i-1} \mid 1\right)}, \qquad (170)$$

is calculated recursively the intermediate LLRs from leaves to the root of the tree by the following updating rules [369], [375], [385], [387], [388], [389], [390]:

$$\begin{aligned}&L_N^{(2i-1)}\left(y_1^N, \hat{u}_1^{2i-2}\right)\\ &\quad = 2\tanh^{-1}\left(\tanh\left(L_{N/2}^{(i)}\left(y_1^{N/2}, \hat{u}_{1,e}^{2i-2} \oplus \hat{u}_{1,o}^{2i-2}\right)/2\right)\right.\\ &\qquad \left.\times \tanh\left(L_{N/2}^{(i)}\left(y_{N/2+1}^N, \hat{u}_{1,e}^{2i-2}\right)/2\right)\right) \qquad (171)\\ &L_N^{(2i)}\left(y_1^N, \hat{u}_1^{2i-1}\right) = L_{N/2}^{(i)}\left(y_{N/2+1}^N, \hat{u}_{1,e}^{2i-2}\right)\\ &\qquad + (-1)^{\hat{u}_{2i-1}} L_{N/2}^{(i)}\left(y_1^{N/2}, \hat{u}_{1,e}^{2i-2} \oplus \hat{u}_{1,o}^{2i-2}\right) \qquad (172)\end{aligned}$$

where $\hat{u}_{1,e}^i$ and $\hat{u}_{1,o}^i$ are subvectors which consist of elements of $\hat{u}_1^i$ with even and odd indices, respectively, and channel LLR:

$$L_1^{(i)}(y_i) = \log \frac{W(y_i \mid 0)}{W(y_i \mid 1)}. \qquad (173)$$

According to [385], [386], for a symmetric B-DMC with the probability density functions (PDFs) $a_w(x)$ of the channel LLRs in (173) for all-zero codeword, we have

$$P(\mathcal{A}_i) = \mathfrak{E}\left(\mathrm{a}_N^i\right)$$

where

$$\mathfrak{E}(\mathrm{a}) := \lim_{\epsilon\to+0}\left(\int_{-\infty}^{-\epsilon} \mathrm{a}(x)\,\mathrm{d}\,x + \frac{1}{2}\int_{-\epsilon}^{+\epsilon} \mathrm{a}(x)\,\mathrm{d}\,x\right), \quad (174)$$

and $\mathrm{a}_N^i, i \in [1, N]$ are recursively obtained using

$$\mathrm{a}_{2N}^{2i} = \mathrm{a}_N^i \circledast \mathrm{a}_N^i, \quad \mathrm{a}_{2N}^{2i-1} = \mathrm{a}_N^i \boxed{\ast}\, \mathrm{a}_N^i, \quad \mathrm{a}_1^1 = \mathrm{a}_w, \quad (175)$$

and $\circledast$ and $\boxed{\ast}$ denote the convolutions of LLR density functions corresponding to (171) and (172), respectively. Recall that the addition of independent random variables, here LLRs, implies the convolution of their densities. Then, a polar code can be constructed by choosing a set $\mathcal{A}$, where $|\mathcal{A}| = N \cdot R$, which minimizes

$$\sum_{i\in\mathcal{A}} P(\mathcal{A}_i)$$

2) APPROXIMATE DENSITY EVOLUTION

To reduce the complexity of DE, several approximate methods have been proposed. In [391], an approach was proposed based on the upper bound and the lower bound on the error probability of the sub-channels.

The success of the Gaussian approximation [205] for sparse graph codes suggests the use of a similar approach for polar codes to approximate the density evolution of LLRs throughout the tree graph [392], [393]. The intuition behind this approximation is that the channels $W_N^{(i)}$ behave like binary AWGN channels with varying noise levels when N is sufficiently large. Therefore, we only need to monitor the noise variances of these channels. For AWGN channels, the distribution of the channel LLRs in (173) is Gaussian, that is, $L_1^{(i)}(y_i) \sim \mathcal{N}(m, 2m)$ with mean $m = \frac{2}{\sigma^2}$ and variance $\frac{4}{\sigma^2}$, where $\frac{1}{\sigma^2}$ is the SNR of the channel. In [205], it was suggested that the intermediate LLRs in (172) and (171) be approximated by Gaussian random variables where the relationship between the mean (expected value) E and the variance V is $V[L_N^{(i)}] = 2E[L_N^{(i)}]$. As a result, considering the check and variable nodes of degree $d_c = d_v = 3$ in the factor graph of polar codes (Fig. 56), the convolution operations in (175) are reduced to [205, eqs. (6), (4)]:

$$E\left[L_N^{(2i-1)}\right] = \phi^{-1}\left(1 - \left(1 - \phi\left(E\left[L_{N/2}^{(i)}\right]\right)\right)^2\right) \quad (176)$$

$$E\left[L_N^{(2i)}\right] = 2E\left[L_{N/2}^{(i)}\right] \qquad (177)$$

where function $\phi(x)$ for $x \in [0, \infty)$ is defined as [205, Definition 1]

$$\phi(x) = \begin{cases} 1 - \frac{1}{\sqrt{4\pi x}} \int_{\mathbb{R}} \tanh \frac{u}{2} e^{-\frac{(u-x)^2}{4x}} du, & \text{if } x > 0 \\ 1, & \text{if } x = 0 \end{cases} \quad (178)$$

To reduce the complexity of calculating $\phi(x)$, an approximation was suggested with acceptable accuracy in [205]. For $x < 10$, one can use the following curve fitting:

$$\phi(x) \sim e^{\alpha x^{\gamma} + \beta}$$

where $\alpha = -0.4527$, $\beta = 0.0218$, and $\gamma = 0.86$. For $x \geqslant 10$, we can use the average of the upper and lower bounds for $\phi(x)$ as [205, Lemma 1]:

$$\sqrt{\frac{\pi}{x}} e^{-\frac{x}{4}} \left(1 - \frac{3}{x}\right) < \phi(x) < \sqrt{\frac{\pi}{x}} e^{-\frac{x}{4}} \left(1 + \frac{1}{7x}\right), x > 0. \quad (179)$$

3) PARTIAL ORDER & POLARIZATION WEIGHT

There are also SNR-independent low-complexity methods for reliability evaluation. It was shown in [379], [380], [394] that there exists a certain partial order relation between the reliabilities of the sub-channels in polar coding. They are called "partial" because the available relations cannot form a fully ordered integer sequence corresponding to the indices of all N sub-channels. The partial orders are deterministic and universal; that is, the relation holds for any transmission channel. Hence, they are called universal partial order (UPO).

Given $i, j \in [0, 2^n - 1]$, we denote the partial order $i \preceq j$ if they satisfy one of the following conditions [377], [378, Definition 1]: a) $\text{supp}(i) \subseteq \text{supp}(j)$, b) $\text{supp}(j) = (\text{supp}(i) \backslash \{a\}) \cup \{b\}$ for some $a \in \text{supp}(i)$, $b \notin \text{supp}(i)$ and $a < b$, and c) there exists $k \in [0, 2^n - 1]$ that satisfy $i \preceq k$ and $k \preceq j$. In [385, Sec. V], it was shown that for all $i \in \mathcal{A}$ and $j \in [0, N-1]$, if $j \succeq i$ then $j \in \mathcal{A}$.

The universal partial order of polar codes has two main properties [395, Proposition 1]: (a) The orders determined for a code of block-length N remain unchanged for block-length of $2N$ (nested property), (b) the order of $x \prec y$ and the order of $(N- 1 - x) \succ (N - 1 - y)$ are twin pairs for a polar code with block-length N.

Later, a closed-form algorithm based on binary expansion known as *polarization weight* (PW) was proposed in [395], [396] to characterize the reliability order of the subchannels for AWGN channels. The polarization weight of the sub-channel x with binary expansion $(b_{n-1}, \ldots, b_1, b_0)$ is defined as [395, Definition 3]

$$f^{\text{PW}} : x \mapsto \sum_{i=0}^{n-1} b_i \beta^i \quad (180)$$

where β is an optimization parameter (the suggested value is $\beta = 2^{1/4}$). Computing the PW for all sub-channels gives a reliability measure, and subsequently a reliability order. The PW algorithm preserves the nested structure (or nested frozen sets) for polar codes [395, Proposition 2]. This property helps in using one long reliability sequence that is found offline, for shorter block-lengths. Due to the low complexity, this method was adopted in the 5G standard [397] in the form of a reliability sequence, which gives the indices of sub-channels in ascending reliability order.

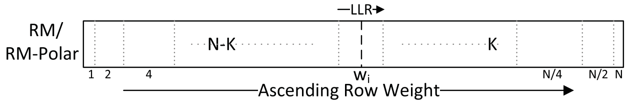


FIGURE 57. RM-Polar Construction [398], a construction based on RM rule and Arıkan rule.

4) RM-POLAR CONSTRUCTION

Alternatively, we can employ both the weight-based RM rule and the reliability-based Arıkan rule to construct a polar-like code as demonstrated in [398, Sec. VI.1]. That is, the set $\mathcal{A}$ is divided into two;

$$\mathcal{A} = \mathcal{A}^{\text{RM}} \cup \mathcal{A}^{\text{Polar}}. \quad (181)$$

Given the blocklength $N = 2^n$ and code dimension K, we find $r' = \arg\max_r \sum_{j=0}^{r} \binom{n}{j} \leqslant K$ and denote $K' = \sum_{j=0}^{r'} \binom{n}{j}$. Then, we find the indices of K' rows of $\boldsymbol{G}_N$ with weights $\text{w}(\boldsymbol{g}_i) \geqslant 2^{n-r'}$. That is,

$$\mathcal{A}^{\text{RM}} = \left\{ i \in [0, N-1] \, : \, 2^{\text{w}(\text{bin}(i))} \geqslant 2^{n-r'} \right\},$$

where $K' = |\mathcal{A}^{\text{RM}}|$. To find $\mathcal{A}^{\text{Polar}}$, we exclude the indices in $\mathcal{A}^{\text{RM}}$ from the reliability sequence of length N and select the remaining $K - K'$ indices from the most reliable indices remaining in the sequence with row weight $2^{n-(r'+1)}$. That is,

$$\mathcal{A}^{\text{Polar}} = \underset{\substack{\mathcal{S} \subseteq ([0,N-1] \backslash \mathcal{A}^{\text{RM}}) \\ |\mathcal{S}| = K - |\mathcal{A}^{\text{RM}}|}}{\arg\max} \sum_{\substack{i \in \mathcal{S} \\ \text{w}(\text{bin}(i)) = n - r' - 1}} L_N^{(i)},$$

where $L_N^{(i)}$ is the average LLR of sub-channel i which can be found from the methods discussed in the previous sections. Observe that this approach gives r-th order RM code (r, n) when $K = \sum_{j=0}^{r} \binom{n}{j}$ and therefore $\mathcal{A}^{\text{Polar}} = \varnothing$. If we use the 5G reliability sequence in this process, we call it *5G-RM* construction [399], otherwise (crossed) *RM-polar* construction, can be denoted by RMxPolar codes, due to intersection with RM codes. Fig. 57 illustrates the selection of K indices with this approach.

A different approach for constructing RM-Polar codes is to constrain the selection of most reliable sub-channels with row weights of $\text{w}(\boldsymbol{g}_i) \geqslant 2^{n-r'-1}$ [400] where set $\mathcal{A}$ is defined as below:

$$\mathcal{A} = \underset{\substack{\mathcal{S} \subset [0,N-1] \\ |\mathcal{S}| = K}}{\arg\max} \sum_{\substack{i \in \mathcal{S} \\ \text{w}(\text{bin}(i)) \geqslant n - r' - 1}} L_N^{(i)},$$

To differentiate from the RMxPolar codes, we refer to this code as *constrained RM-Polar* codes. Note that the set $\mathcal{A}$ in this approach is most likely different from (181), because there might be sub-channels with row weight(s) larger than the minimum weight w_{min} excluded from $\mathcal{A}$ and instead include more sub-channels with row weight w_{min}. This results in a code with a higher number of cosets $\mathcal{C}_i(\mathcal{A})$, where $\mathrm{w}(\mathrm{bin}(i)) = \log_2(\mathrm{w}_{min})$ and therefore consequently containing more minimum weight codewords. Due to this improvement, relative to the constrained RM-polar codes in [400], crossed RM-polar codes are also called improved RM-polar codes [401].

According to the Union bound, the block error rate is a function of the minimum distance d_{min} and the number of small-weight codewords, in particular, the number of minimum weight $A_{w_{\min}}$, a.k.a. the error coefficient. Hence, a code design approach that considers both the sub-channel reliability and the error coefficient is of interest; however, optimizing both would be highly complex and requires performing a search for every code. An approach in [378], [402] was suggested that simply improves the error coefficient of an available reliability-based construction, which was discussed above. Also, as shown in [403], we can modify the reliability-based construction by bit-swapping to reduce the possibility of eliminating the correct path in the list decoding. This shows that one can tailor the construction of polar code for a specific decoding algorithm. This insight was also used in [404] to partition the code block and to use different list sizes for each partition (depending on the possibility of correct path elimination in order to reduce computational complexity.

F. CODE CONCATENATION AND PRE-TRANSFORMATION

A polar code can be used as an inner code, while an outer code such as a cyclic code can be used for its detection capability. In this case, the systematic cyclic coded bits are called cyclic redundancy check (CRC) bits, which are appended to the information bit sequence $\boldsymbol{d}$. Observe that the CRC bits as additional parity bits are placed on sub-channels in $\mathcal{A}^c$. As the probability of false detection increases with the length of the sequence $\boldsymbol{d}$, so-called distributed CRCs are suggested to be used. In the distributed CRCs scheme, the information sequence is divided into segments, and each segment is outer encoded separately.

Another pre-transformation scheme considers parity bits u_i resulting from a linear combination of bits in the sequence $\boldsymbol{d}$ such that $u_i = \sum_{j\in\mathcal{J}} u_j$ where $\mathcal{J}$ is a subset of $\mathcal{A}$ and for every $j \in \mathcal{J}$, we have $j < i$. These parity bits are carried similarly by sub-channels in $\mathcal{A}^c$, hence sometimes called dynamic frozen bits [405]. The choice of the linear combination of bits has not been explicitly structured except in a recently introduced variant of polar codes called polarization-adjusted convolutional (PAC) codes. We discuss them in detail in Section VII-H.

G. INFORMATION (OR SPATIALLY) COUPLED POLAR CODES

To improve the efficiency of the transport block (TB) in communication standards, consecutive systematic code blocks (CBs) in a TB are coupled by sharing a portion of the information bits, hence called *partially information coupled* (PIC) polar codes [406], [407], [408]. In this scheme, a $\boldsymbol{u}$ corresponding to the l-th CB, consists of the coupled bits $\boldsymbol{u}^c_{l-1}$ with the previous CB, the code bits $\boldsymbol{u}_l$, the coupled bits $\boldsymbol{u}^c_l$ with the next CB, and the CRC bits $\boldsymbol{c}_l$ of the CB. Fig. 58 illustrates the coding process that includes permutation Π, systematic encoding $\boldsymbol{x} = [\bar{\boldsymbol{u}}_{\mathcal{A}}, \bar{\boldsymbol{u}}_{\mathcal{A}}\boldsymbol{G}^{-1}_{\mathcal{A}\mathcal{A}}\boldsymbol{G}_{\mathcal{A}\mathcal{A}^c}]$ according to (163) and (164), followed by puncturing of the coded bits corresponding to $\boldsymbol{u}^c_{l-1}$. The partially coupled information bits are punctured in the next CB. Given that the coupled bits are correctly decoded in the previous CB, the error propagation resulting from these bits is avoided in successive cancellation-based decoding algorithms. The inter-CB decoding scheme, which realizes a windowed decoder with variable window size, achieves a trade-off between the decoding performance and complexity. In [409], the coupled portions of the information bits are transmitted as frozen bits or added to the information bits, both in the next CB in a row.

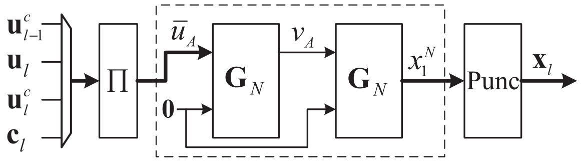


FIGURE 58. Encoding scheme for a single CB [407].

H. PAC CODES

Polarization-adjusted convolutional (PAC) codes are pre-transformed polar codes suggested initially in the Shannon lecture at the International Symposium on Information Theory (ISIT), 2019. As mentioned in Section VII-B, although the motivation behind polar codes was to build a synthetic channel to boost the cutoff rate inspired by classical concatenated schemes, the concatenation was not employed at that stage due to the performance of polar codes and the availability of the low complexity decoding procedure. Nevertheless, it was later demonstrated that a rate-1 convolutional pre-transformation can improve the block error rate (BLER) for short codes under sequential decoding [410].

In PAC coding, the information bits $\boldsymbol{d} = (d_0, d_1, \ldots, d_{K-1})$ are first mapped to a vector $\boldsymbol{v} = (v_0, v_1, \ldots, v_{N-1})$ using a rate-profile defined by set $\mathcal{A}$. This set includes the indices of the positions where the information bits are placed in the input vector to the convolutional transform (CT). The rule to form this set does not have to be the same as polar codes. The bit values in the remaining positions in $\boldsymbol{v}$ are set to 0. The constraint $v_{\mathcal{A}^c} = 0$ simply leads to an irregular decoding tree. Note that 1) the outputs of CT for indices in $\mathcal{A}^c$, which enter the polar transform, are no longer fixed, i.e., known a priori - unlike in polar coding. 2) In contrast to convolutional coding, in which usually $R_c < 1$, here we use a one-to-one

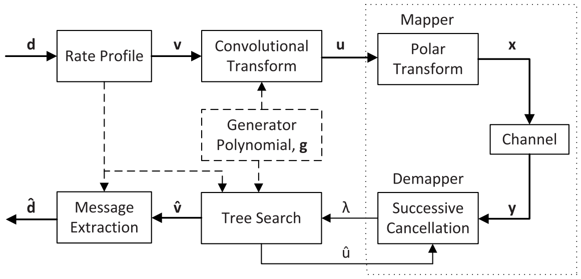


FIGURE 59. PAC Coding Scheme [398].

transform $\boldsymbol{G}$, hence the vectors $\boldsymbol{v}$ and $\boldsymbol{u}$ in $\boldsymbol{u} = \boldsymbol{v}\boldsymbol{G}$ have the same dimension.

The relation between the input bits $d_{i-m,i}$ and the output bit u_i, at time-step i, is obtained as a binary convolution by

$$u_i = \sum_{j=0}^{m} g_j d_{i-j}, \tag{182}$$

where $g_i \in \{0, 1\}$ and m is the number of previous input bits stored in a shift register (m is also known as *memory size*). By representing bit sequences as polynomials in the delay variable D representing a time step in the encoder, an output sequence $x(D)$ is obtained as $g(D)d(D)$, where $g(D) = \sum_{j=0}^{m} g_j D^j$ is the *generator polynomial*. The coefficients of the generator polynomial in the context of convolutional codes are represented in octal notation. For example, the commonly used $\boldsymbol{g} = [1\ 0\ 1\ 1\ 0\ 1\ 1]$ is represented by 133.

After the convolutional transform, the vector $\boldsymbol{u}$ is mapped to $\boldsymbol{x}$, as Fig. 59 shows, employing the polar transform $\boldsymbol{G}_N$; therefore, $\boldsymbol{x} = \boldsymbol{u}\boldsymbol{G}_N$.

The readers can refer to Section V-A on Convolutional codes, a component code used in Turbo coding. In [411], [412], [413] it was shown that the precoding stage can reduce the number of codewords of minimum weight as a result of the inclusion of frozen rows in $\boldsymbol{G}_N$. Hence, precoding can potentially improve the block error rate of polar codes. Various decoding algorithms have been adapted to PAC codes, such as sequential decoding [410], [414], [415], [416], list decoding [398], [417], list Viterbi decoding [418], belief propagation decoding [419], GRAND [420], [421], [422], stack decoding [398], [423], [424], [425], fast and simplified list decoding [426], [427], and other decoding algorithms [428], [429].

Furthermore, reverse PAC coding has recently been proposed [430], [431] that can overcome the limitations of forward precoding to further reduce the number of minimum weight codewords. Among other precoding schemes and variants of PAC codes, we can name tail-biting PAC codes [432], row-merged polar codes [433], [434], spatially coupled PAC codes [435], modified polar codes [401], parity check PAC codes [436], and other schemes [437], [438], [439], [440]. Numerous construction methods for PAC codes, predominantly based on search, have also been suggested in [441], [442], [443], [444]. It was proposed in [378] to use the reliability-based rate-profile of polar codes as it is easy to construct, and instead to modify it to drastically reduce the number of minimum weight codewords. Furthermore, rate-matching techniques, such as puncturing and shortening, for systematic and non-systematic PAC codes have also been investigated in [399], [445], [446], [447], [448], [449], [450]. Finally, PAC coding has been considered for source coding and joint source-channel coding in [451], [452], [453].

I. LARGE KERNEL POLAR CODES (BINARY AND NON-BINARY)

Arıkan's pioneering work [369] was based on the linear binary *kernel matrix* $\boldsymbol{G}_2$ of dimension two and exponent $\frac{1}{2}$ (see Section VII-C). To improve the exponent and, consequently, the performance of polar codes under SC decoding, large non-singular kernel matrices $\boldsymbol{G}$ were suggested in [382]. The polarization effect over BI-DMC still holds for such an $\ell \times \ell$ kernel matrix $\boldsymbol{G}$ provided that none of its column permutations is an upper triangle matrix. The block error probability is found to be $O(2^{-\ell^{n\beta}})$ if β is less than the exponent of the kernel matrix.

The upper and lower bounds of the achievable exponents for the kernels showed that there are no matrices of size smaller than 15×15 with exponents $\mathrm{E}(\boldsymbol{G})$ exceeding $\frac{1}{2}$. Furthermore, it was shown that a general construction based on BCH codes for large ℓ can achieve exponents arbitrarily close to 1. For example, at size 16×16, this construction yields an exponent $\mathrm{E}(\boldsymbol{G}) > \frac{1}{2}$.

Different linear non-binary kernels have also been proposed. In [454], non-binary kernel matrices were constructed based on Reed-Solomon codes and Hermitian codes that were later expanded to other algebraic geometry codes, including Suzuki codes, in [455], concatenated algebraic geometry codes, due to their large minimum distance and often nested structure, in [456]. These kernels have a larger exponent than any linear binary kernel of the same dimension. Furthermore, non-linear binary kernels were studied in [457], [458] which provide exponents superior to any linear binary kernel of the same number of dimensions.

The SC decoding algorithm for large kernel polar codes has the complexity of order $O(2^{\ell} N \log N)$ operations, where $O(2^{\ell})$ is the complexity of kernel processing [459] based on trivial implementation. Although this complexity can be slightly reduced, direct calculation remains prohibitive for kernels with relatively large sizes, such as $\ell = 16, 32$. More efficient algorithms, such as window processing [460], [461] and recursive trellis processing [460] have been proposed. The complexity of these algorithms is still exponential with respect to kernel size ℓ. Moreover, some of these approaches, such as window processing which is based on the idea of expressing LLRs for a large kernel via LLRs for the dimension-2 kernel $\boldsymbol{G}_2$, are not admitted by all kernels in an efficient way.

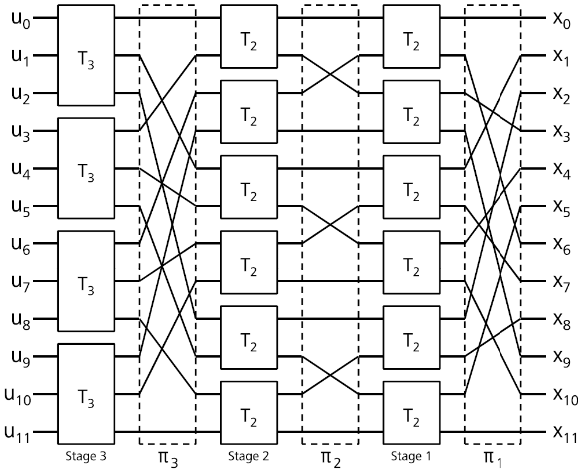


FIGURE 60. Tanner graph of the multi-kernel polar code with the transformation matrix $T_{12} = T_2 \times T_2 \times T_3$ [463].

J. MIXED/MULTI-KERNEL POLAR CODES

As we observed in Section VII-A, the code construction based on the n-fold Kronecker product of $\boldsymbol{G}_2$ restricts the length N of polar codes to be powers of two, i.e., $N = 2^n$. In the case of larger kernels, the code length becomes powers of integers, i.e., $N = l^n$, where l is the dimension of the kernel. One can employ puncturing, shortening, and repetition techniques to adjust the blocklength and have an arbitrary blocklength. However, these techniques come with disadvantages. In addition to degradation in performance, as the punctured and shortened codes are decoded based on their mother polar codes, the decoding latency is proportional to the actual code length. Furthermore, the coordinates of dummy bits representing shortened or punctured bits affect the polarization of the codes and improper selection of these coordinates may result in catastrophic error correction performance.

In this section, we consider the multi-kernel polar codes [462] in which different kernel sizes over the binary alphabet [463] are mixed, while the polarization effect is preserved [464]. Encoding of multi-kernel polar codes follows the general structure of polar codes, and decoding can be performed by any SC-based decoding discussed earlier.

An (N, K) multi-kernel polar code is defined by an $N \times N$ transformation matrix T_N as

$$T_N = T_{p_1} \otimes T_{p_2} \otimes \cdots \otimes T_{p_s}, \tag{183}$$

where the code length is $N = p_1 p_2 \ldots p_s$ and the binary polarizing matrices T_{p_i} of size $p_i \times p_i$ for $i = 1, \ldots, s$, called kernels of dimension p_i. Note that the Kronecker product in (183) is not commutative. Fig. 60 illustrates the Tanner graph of a polar code with length $N = 12 = 2 \times 2 \times 3$ composed of two kernels of dimension 2 and one kernel of dimension 3. Observe that when all kernels are composed of kernel $T_2 = \boldsymbol{G}_2$, we will have the polar codes discussed in Section VII-A. The permutations of edges between kernels are implicitly defined by the Kronecker product, similarly to polar codes [369]. Permutation π_i connects the output $j, j = 0, 1, \ldots, p_{i-1}$ of stages $i-1, i = 2, 3, \ldots, s$ to the input $\pi_i(j)$ of stage i.

A multi-kernel polar code can be designed based on three principles of reliability, Hamming distance, and a combination of both. Let us review these code design approaches: 1) The reliability approach is based on the polarization effect, with the aim of minimizing the BLER under SC decoding.

K. DECODING ALGORITHMS

Since Arıkan's seminal work, many decoding algorithms have been adapted for decoding polar codes and their variants. In this section, we review the major decoders and their recent improvements and compare them in terms of complexity and block error rate performance. We also consider the realization of these decoders as hardware architectures and provide an assessment of their compatibility with the KPIs of the potential scenarios in 6G.

1) SUCCESSIVE CANCELLATION (SC) DECODING

As we observed, channel combining introduces a correlation between the source bits. As a result, each coded bit with a given index relies on all its preceding source bits. This correlation can be conceptually treated as *interference* in the source-bit domain and can be exploited in the decoding process. Therefore, the bits are decoded one at a time in a specific order. The bit decision $\hat{u}_i$ is made before the calculations start to find the next bit $\hat{u}_{i+1}$, and the already decided bits influence the decision of the following bit decisions. The successive cancelation of the "interference" caused by the previous bits improves the reliability of retrieving the source bits. Apparently, the name of this decoding process has been borrowed from the successive interference cancellation (SIC) decoding technique used by a receiver in wide-band wireless communications where two (or more) packets arrive simultaneously (which otherwise cause a collision). SIC is an iterative algorithm in which received data are decoded on the order of decreasing power levels. That is, in the case of two signals, the stronger signal is first decoded and then subtracted from the combined signal, and then the weaker signal is extracted from the residue.

In each use of the system, a codeword is transmitted and a channel output vector $y \in \mathcal{Y}^N$ is received. In SC decoding, the source bits corresponding to the frozen bits are set to zero, $\hat{u}_i = 0, i \in \mathcal{A}^c$. The information bits $\hat{u}_i, i \in \mathcal{A}$ are decoded sequentially through maximum likelihood (ML) decoding of the channel $W_N^{(i)}$ as

$$\hat{u}_i\left(y_1^N, \hat{u}_1^{i-1}\right) = \underset{u_i=0,1}{\operatorname{argmax}}\, W_N^{(i)}\left(y_1^N, \hat{u}_1^{i-1} \mid u_i\right). \tag{184}$$

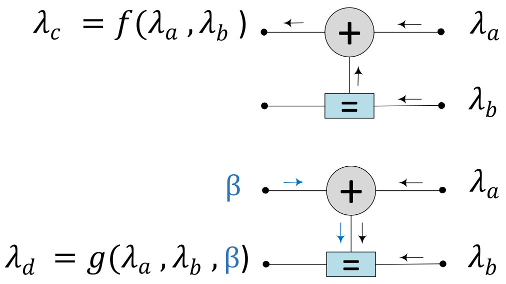

$\lambda_c = f(\lambda_a, \lambda_b)$
λ_a
λ_b
β
λ_a
$\lambda_d = g(\lambda_a, \lambda_b, \beta)$
λ_b

FIGURE 61. Internal LLR (λ) calculations [415].

Practically, similar to (173), the receiver first calculates the vector of logarithmic likelihood ratios (LLRs) with

$$L_n^{(i)} = \log \frac{W(y_i|x_i = 0)}{W(y_i|x_i = 1)}, \tag{185}$$

for each element of the channel output vector and feeds it into a decoder, here the SC decoder.

In SC decoding, information bits are estimated by a hard decision based on the final evolved LLRs λ_i^0. When decoding the i-th bit, if $i \notin \mathcal{A}$, regardless of the final LLR value λ_i^0, $\hat{u}_i$ is set as a frozen bit, that is, $\hat{u}_i = 0$. Otherwise, u_i is decided by a maximum likelihood (ML) rule as equation (186) based on the previously estimated vector $(\hat{u}_1, \ldots, \hat{u}_{i-1})$.

$$\hat{u}_i = h\left(L^{(i_0)}\right) = \begin{cases} 0 & L_0^{(i)} = \log \frac{W_N^{(i)}\left(y_1^N, \hat{u}_0^{i-1} | \hat{u}_i = 0\right)}{W_N^{(i)}\left(y_1^N, \hat{u}_0^{i-1} | \hat{u}_i = 1\right)} > 0, \\ 1 & \text{otherwise} \end{cases} \tag{186}$$

The decoder estimates the transmitted bits successively by computing LLRs of the indexed edges. The LLR of edge (i, j) is computed by

$$L_j^i = \begin{cases} f\left(L^i_{(j+1)}, L^{i+2^j}_{j+1}\right) & \text{if } B(i,j) = 0 \\ g\left(L^{(i-2^j)}_{j+1}, L^i_{(j+1)}, \hat{\beta}_j^{i-2^j}\right) & \text{if } B(i,j) = 1 \end{cases} \tag{187}$$

where $0 \leqslant i < N$, $0 \leqslant j \leqslant n$, $B(i, j) = \lfloor \frac{i}{2^j} \rfloor \mod 2$, and $\hat{\beta}_j^i$ denote the partial sum, which corresponds to the propagation of estimated bits $\hat{u}_i$ backward into the factor graph, from right to left in Fig. 56. Note that i and j denote the bit index and the stage index, respectively, and the channel/intermediate LLRs are sometimes denoted by λ (we use it in the rest of this subsection) or α. The reason is that depending on the decoding algorithm, L might be used for other purposes.

The functions f and g in (187) as illustrated in Fig. 61, equivalent to (188) and (189), respectively, can be obtained by min-sum approximations, similar to decoding algorithm used for Reed-Muller codes in [465], by

$$f(\lambda_a, \lambda_b) \approx \operatorname{sgn}(\lambda_a) \cdot \operatorname{sgn}(\lambda_b) \cdot \min(|\lambda_a|, |\lambda_b|) \tag{188}$$

$$g\left(\lambda_a, \lambda_b, \hat{\beta}\right) = (-1)^{\hat{\beta}} \lambda_a + \lambda_b \tag{189}$$

where λ_a and λ_b are the incoming LLRs to a node and $\hat{\beta}$ is the partial sum of previously decided bits.

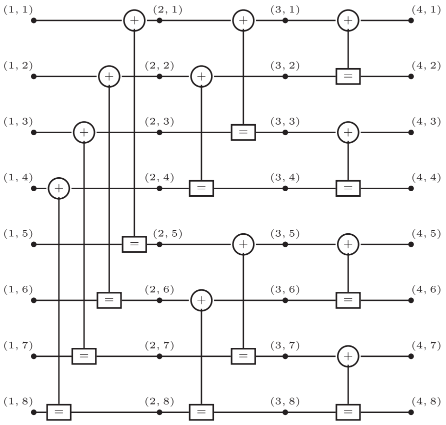


FIGURE 62. The factor graph in Fig. 56 labeled for BP decoding [466].

2) BELIEF PROPAGATION (BP) DECODING

Belief propagation decoding uses graphical models and message passing to iteratively update the beliefs or probabilities of the transmitted codeword symbols based on received channel observations. It is commonly used for low-density parity-check (LDPC) codes (see Section VI) and turbo codes (see Section V). The sparseness of the graph representation of the $\boldsymbol{G}_N$ transformation in Fig. 56 suggests that Gallager's belief propagation algorithm [144] can be effective in decoding polar codes. In Fig. 62, the nodes are labeled with pairs of integers (i, j), $1 \leqslant i \leqslant n + 1$, $1 \leqslant j \leqslant N$. The leftmost nodes, $(1, j)$, correspond to the source data u_j that are to be estimated, while the rightmost nodes $(n + 1, j)$ are associated with channel input variables x_j that are transmitted through a noisy channel. The BP decoder associates two messages to each node (i, j): a right-propagating message $R^{(t)}_{i,j}$ and a left-propagating message $L^{(t)}_{i,j}$, where $t \geqslant 0$ denotes the time index. These messages correspond to log-likelihood ratios (LLR) at time t and are set as

$$L^{(0)}_{n+1,j} = \log \frac{P(x_j = 0 \mid y_j)}{P(x_j = 1 \mid y_j)}$$

$$R^{(0)}_{1,\text{j}} = \log \frac{P(u_j = 0)}{P(u_j = 1)} = \begin{cases} 0 & \text{if } j \in \mathcal{A} \\ \infty & \text{otherwise} \end{cases}$$

Note that a frozen coordinate is by default $u_j = 0, j \in \mathcal{A}^c$, hence $R^{(0)}_{1,j} = \infty$. However, the $u_j = 0$ and 1 are equally likely values for information coordinates $j \in \mathcal{A}$, i.e., $P(u_j = 1) = P(u_j = 0)$, thus $R^{(0)}_{1,j} = 0$. The rest of $R^{(0)}_{i,j}$ and $L^{(0)}_{i,j}$ are initialized to 0.

In each stage of Fig. 62, there are $\frac{1}{2} N \log N = 4$, for $N = 8$ basic computational blocks shown in Fig. 63. The upper node is a check node representing the parity-check constraint, while the lower node is a variable node. This computational block computes the left/right-propagating messages based on three inputs as

$$L^{(t+1)}_{i,j} = f\left(L^{(t)}_{i+1,j}, L^{(t)}_{i+1,j+N_i} R^{(t)}_{i,j+N_i}\right)$$

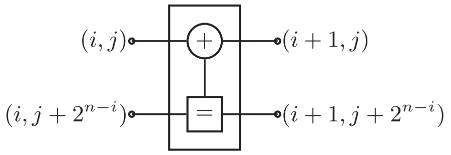


FIGURE 63. The basic computational block in factor graphs [466].

$$
\begin{aligned}
L_{i,j+N_i}^{(t+1)} &= L_{i+1,j+N_i}^{(t)} f\left(L_{i+1,j}^{(t)}, R_{i,j}^{(t)}\right)\\
R_{i+1,j}^{(t+1)} &= f\left(R_{i,j}^{(t)}, L_{i+1,j+N_i}^{(t)} R_{i+1,j+N_i}^{(t)}\right)\\
R_{i+1,j+N_i}^{(t+1)} &= R_{i,j+N_i}^{(t)} f\left(R_{i,j}^{(t)}, L_{i+1,j}^{(t)}\right)
\end{aligned}
$$

where $f(x, y) = (1 + xy)\ /(x + y)$ for real values x, y and $N_i = 2^{n-i}$.

The messages carry information about the reliability of each symbol or constraint and are exchanged between the variable and the check nodes. At each iteration represented by time index t, a variable node calculates its outgoing messages based on the incoming messages from the connected check nodes. Similarly, check nodes update their outgoing messages based on the incoming messages from the connected variable nodes. These message updates are repeated until a stopping criterion is met or until convergence is achieved. The stopping criterion is satisfying $\boldsymbol{x} = \boldsymbol{u}\boldsymbol{G}_N$ where $\boldsymbol{u}$ and $\boldsymbol{x}$ are the hard decision (similar to (186)) made on the left-most and right-most messages, respectively.

Belief propagation decoding is known for its effectiveness in achieving near-optimal decoding performance for low-density parity-check codes due to possessing a sparse parity-check matrix. However, it may suffer from convergence issues in the presence of high noise levels or certain code structures.

3) SC LIST DECODING

As we discussed in Section VII-K1, the successive cancellation decoder locally makes a hard decision for each bit value u_i and proceeds with decoding the subsequent bits. Although the decision at each decoding step is locally optimal, given that the previous bits have been decoded correctly, it is not necessarily globally optimal. To overcome this shortcoming, one can consider both options for the value of each bit, $u_i = 0, 1$, and form a binary tree [467], [468], [469]. A path in the tree from the origin to a leaf is a solution to decoding. Obviously, the path with the highest likelihood will be selected from the list for optimal decoding. That is, the probability to be maximized is

$$
P\left(\hat{u}_0^{N-1} \mid y_0^{N-1}\right) = \prod_{t=0}^{N-1} P\left(\hat{u}_t \mid \hat{u}_0^{i-1}, y_0^{N-1}\right).
$$

However, we cannot explore the entire decoding tree to examine all paths and we constrain the traversal to L paths. Therefore, the solution obtained from the decoding may be sub-optimal. In SCL decoding, the probability of the partial path $l \in [1, L]$ representing the sequence $\hat{u}_0^{i-1} = (\hat{u}_0, \hat{u}_1, \ldots, \hat{u}_{i-1})$ is calculated by

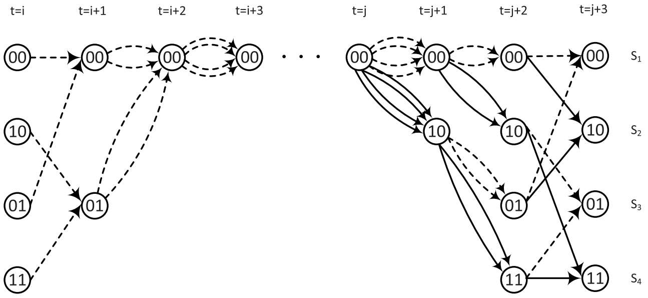


FIGURE 64. Irregular trellis in list Viterbi decoding where $v_t = 0$ for $t = [i+1, \ldots, j]$ and $t \in \mathcal{A}^c$. Note that the paths from $t = i+1$ to $t = j$ are not pruned [418].

$$
P\left(\hat{u}_0^i[l] \mid y_0^{N-1}\right) = \prod_{t=0}^{i} P\left(\hat{u}_i[l] \mid \hat{u}_0^{i-1}[l], y_0^{N-1}\right). \tag{190}
$$

In practice, the logarithm of (190) is used. Since $\log(x) < 0$ for $x < 1$, we multiply the resulting logarithm by -1 to have a positive metric. Therefore, we get the following logarithmic path metric for the sequence $\hat{u}_0^{i-1}$ on path l [415, Sec. 2.5.2]:

$$
\begin{aligned}
PM_l^{(i-1)} &= -\log P\left(\hat{u}_0^{i-1}[l] \mid y_0^{N-1}\right)\\
&= -\sum_{j=0}^{i-1} \log P\left(\hat{u}_j[l] \mid \hat{u}_0^{j-1}[l], y_0^{N-1}\right)\\
&= PM_l^{(i-1)} + \mu_l^{(i)}.
\end{aligned} \tag{191}
$$

where $\mu_l^{(i)} = -\log P(\hat{u}_i[l] \mid \hat{u}_0^{i-1}[l], y_0^{N-1})$ denotes the branch metric and $PM_i^{(-1)} = 0$. A genie such as CRC bits (see Section VII-F) can also help detect the correct path [470], that is, the transmitted data. The same list decoding procedure can be used with other concatenated codes, such as PAC codes or PC-polar codes.

For PAC codes, one can use SC-based list Viterbi decoding [418]. As a result of the convolutional transformation, each path is associated with a single state at each decoding step. Fig. 64 illustrates the trellis as a graphical representation of the Viterbi list decoding with list size 4 (the number of surviving paths), equal to the number of states.

4) SIMPLIFIED/FAST SC-BASED DECODING

The distribution of frozen coordinates in sequence $[0, N-1]$ forms sub-sequences of length $2^s, s = 0, \ldots, n-1$ with specific patterns. The two trivial ones are the sub-sequences with all frozen coordinates and no frozen coordinates. As shown in Fig. 65, the nodes (in the tree that represents the factor graph) associated with these leaves representing these sub-sequences are called rate-0 and rate-1 nodes, respectively. The update rule in (171), that is, for LLRs values of x and y we have $x \boxed{\circledast} y \triangleq 2\,\mathrm{atanh}(\tanh(x/2)\ \tanh(y/2))$, has the following property that can be exploited to simplify the message passing process:

$$
h(x \boxed{\circledast} y) = h(x) \oplus h(y) \text{ if } x \cdot y \neq 0. \tag{192}
$$

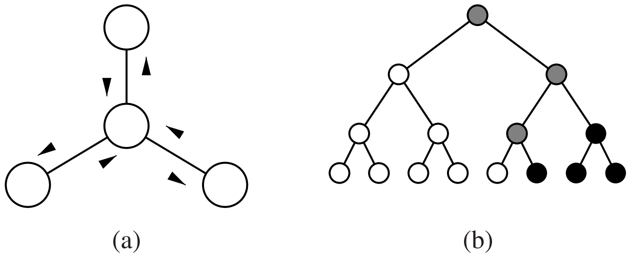


FIGURE 65. (a) Local decoder, (b) Labelling the leaf nodes for $2^n = 8$. The white circles represent rate-0 nodes, while the rate-1 nodes are shown as black circles [475].

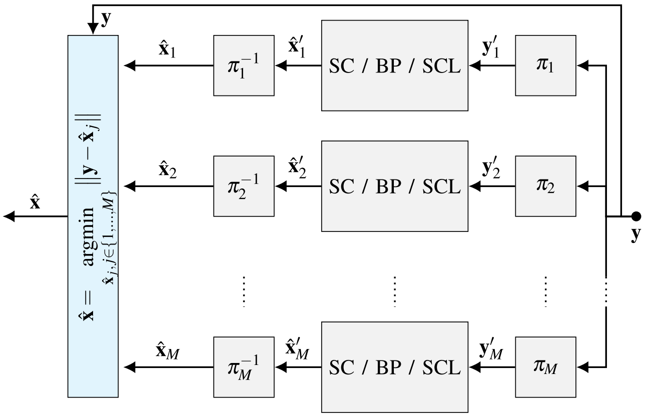


FIGURE 66. Automorphism/Permutation Ensemble Decoding scheme [493].

Let V_v denote the set of nodes of the subtree rooted at node v, and $\ell(u)$ denote the index of a leaf node u. Furthermore, we define the set $\mathcal{I}_v$ for each node v that contains the indices of all leaf nodes that are descendants of node v, as follows:

$$\mathcal{I}_v = \{\ell(u) : u \in V_v \text{ and } u \text{ is a leaf node}\}.$$

Now, when a rate one-1 v is activated, it immediately calculates β_v via

$$\beta_v = h(\lambda_v),$$

and all bits with indices in $\mathcal{I}_v$ can be immediately decoded using

$$(\hat{u}[\min \mathcal{I}_v], \ldots, \hat{u}[\max \mathcal{I}_v]) = \beta_v G_{n-d_v},$$

where d_v indicate the depth of node v. Effectively, the decoding of a rate-1 node is simply finding the constituent code by hard-decision $h(\cdot)$, see (186), of soft-input values λ_v and computing the inverse transform which is $G_{n-d_v}^{-1} = G_{n-d_v}$ (see Section VII-C). The decoding of rate-0 nodes is simpler. Since all descendants of a rate zero node v are themselves rate-0, node v can immediately set the constituent code β_v to an all-zero sequence.

Later, these special nodes were extended to single parity check (SPC) nodes and repetition (REP) nodes [471], type-I,II,III,IV nodes, [472] and sequence repetition node [473], and adapted to other decoding algorithms such as SC list decoding [474].

5) ITERATIVE SC-BASED DECODING

Using CRC bits for block error detection, one can recover the correct path (transmitted data) in additional T attempts by making different decisions, i.e., flipping the initial estimate of the bit (s), in SC decoding [476], [477], or effectively following different paths in the decoding tree in SC list decoding [478], [479]. The earlier decoding is called SC-flip (SCF) decoding, and the latter is referred to as SC list decoding with shifted-pruning (SP), SCL-flip (SCLF) decoding, or other names such as SCL decoding with list-flipping and path flipping [480], [481], [482], [483], [484], [485], [486], [487], [488], [489], [490], [491]. Furthermore, there is an efficient way to start re-decoding in the additional iterations by partial rewinding, that is, the re-decoding process is performed by partially rewinding the SC process, not necessarily from the first bit. This can significantly reduce the complexity of decoding in the SCF and SCLF decoding algorithms. Recently, a different approach called SCL-perturbation decoding has been developed [492], which is based on the idea of adding noise or perturbation to the received sequence in the literature, where small random perturbations are introduced to the received symbols before each SCL decoding attempt. Numerical results show that the perturbation yields a higher coding gain than flipping at a large block length with the same number of decoding attempts.

6) AUTOMORPHISM ENSEMBLE DECODING

A different approach to decoding is to exploit the symmetries of the codes and to have a set of decoders, where each decodes a distinct permuted received sequence $\pi_i(\boldsymbol{y}), i \in [1, M]$, as shown in Fig. 66. The output of every decoder as an estimated transmitted codeword is then de-permuted and the one with maximum likelihood is chosen [493]. That is,

$$\hat{\boldsymbol{x}} = \underset{\hat{\boldsymbol{x}}_j, j\in\{1,\ldots,M\}}{\operatorname{argmin}} \|\hat{\boldsymbol{x}}_j - \boldsymbol{y}\|^2 = \underset{\hat{\boldsymbol{x}}_j, j\in\{1,\ldots,M\}}{\operatorname{argmax}} \sum_{i=0}^{N-1} \hat{x}_{j,i} \cdot y_i, \quad (193)$$

where

$$\hat{\boldsymbol{x}}_j = \pi_j^{-1}(\mathrm{Dec}(\pi_j(\boldsymbol{y}))). \quad (194)$$

An automorphism group of a code $\mathcal{C}$ is the set of permutations that map codewords to other codewords and leave the code globally invariant. This approach, called automorphism group decoding, has been employed for Bose-Chaudhuri-Hocquenghem (BCH) codes in [494]. The automorphism group of decreasing monomial codes, including polar codes, was shown in [380] to be at least the lower-triangular affine group (LTA), which relies on the partial order of sub-channels. This was later extended to block LTA [495].

This approach can decode M permuted received sequences in parallel with potentially the latency of the component decoding, as it does not require the path selection stages used in SCL decoding. However, the choice of permutations to achieve a performance close to SCL decoding is challenging despite an effort in [496] to classify the permutations.

7) OTHER NEAR-ML AND ML DECODING ALGORITHMS

Among other decoding algorithms adapted to polar codes and their variants, we can name the following decoding algorithms:

- Soft CANcellation (SCAN) decoding [497] is an iterative decoder that allows one to reduce the number of iterations of the BP decoder (see Section VII-K2) by adopting the SC schedule (see Section VII-K1). Unlike SC decoding, SCAN decoding propagates soft information in both directions.
- Fano decoding [414], [498], [499], [500] is a variable-complexity ML decoding algorithm where the decoder examines a sequence of adjacent non-frozen bits and explores the partial paths of the code tree by moving forward and backward, aiming to find the maximum likelihood path.
- Stack decoding [398], [501], [502] keeps a stack of size/depth D of partial paths sorted with respect to the path metric. The algorithm extends the path with the best metric at the top of the stack. The stack decoding is a memory-intensive algorithm with a variable time complexity that, instead of backtracking as in the Fano decoding, selects to extend the best partial path in the stack at each time step.
- Sphere decoding [430], [503], [504] based on the depth-first approach first finds a candidate solution close to the received sequence in terms of the Euclidean distance (ED). Then, it searches for a closer candidate to the received sequence by tree pruning, if there exists.
- Generic/universal decoding algorithms such as ordered statistics decoding (OSD) [299], [505] and guessing random additive noise decoding (GRAND) [420], [421], [506] are reliability-based algorithms that guess the error pattern introduced through the channel in a specific order and subsequently check them using a systematic generator matrix or a parity check matrix, respectively, to find the candidate closest to the received sequence. They have shown remarkable performance of ML/near-ML decoding, given enough trials for search. Among them, GRAND best suits high-rate codes.

The algorithms above predominantly provide excellent performance at the cost of high and variable complexity. We compare the hardware implementation of these algorithms in Section VII-O.

L. RATE-COMPATIBLE POLAR CODES

As discussed in Section VII-A, the length of polar codes is restricted to the powers of two, $N = 2^n, n \geqslant 1$. To accommodate the various practical E-length requirements or equivalently rate requirements, a mother code (N, K) is punctured or shortened according to the employed index pattern, by P bits or S bits, respectively. Note that the focus of this section is on non-systematic polar codes.

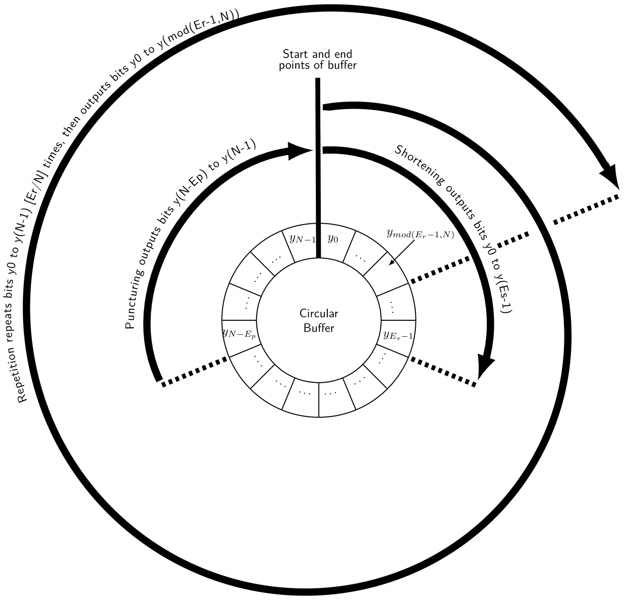

Repetition repeats bits y0 to y(N-1) [Er/N] times, then outputs bits y0 to y(mod(Er-1,N))
Start and end points of buffer
Puncturing outputs bits y(N-Ep) to y(N-1)
Shortening outputs bits y0 to y(Es-1)
y_{N-1}
y_0
$y_{mod(E_r-1,N)}$
Circular Buffer
y_{N-E_p}
y_{E_s-1}

FIGURE 67. The Circular buffer in 5G for rate-matching [509].

The punctured code $(N-P, K)$ with length $E_p = N-P$, where $P < N-K$, clearly has a higher rate, $K/(N-P) > K/N$. The P coded bits are not transmitted, and the decoding is performed on the mother code of length $2^{\lceil \log_2 E_p \rceil}$ considering the punctured bits as erased, that is, the corresponding LLRs are set to zero. The LLRs of these unreliable bits in the decoder are evolved to zero at the output of the decoder, resulting in *incapable bits* [507]. Therefore, these indices should be included in the frozen set $\mathcal{A}^c$ to avoid a drastic error rate. Note that the number of incapable bits is equal to the number of punctured bits [507]. In the 5G standard, the puncturing set $\mathcal{P}$ is given by $\mathcal{P} = \{0, \ldots, P-1\}$, that is, the first P indices of $[0, N-1]$ in natural order (alternatively in bit-reversal order [508]) are considered frozen. The rest of the $N-K$ frozen bits are chosen from the reliability sequence, from the least reliable indices, in a circular buffer configuration [397].

On the other hand, for the shortened polar code $(N-S, K)$ with length $E_s = N-S$, there are S coded bits restricted to zero in all codewords of its mother code (N, K). Therefore, these known bits are not transmitted and the decoding is performed on the mother code of length $2^{\lceil \log_2 M \rceil}$ considering the known shortened bits, That is, the corresponding LLRs are set to $+\infty$ (a large value). The LLRs of these known bits in the decoder are evolved to large values at the output of the decoder, resulting in *overcapable bits*. Therefore, these indices can be included in the frozen set $\mathcal{A}^c$, since the source bits $\boldsymbol{u}_{\mathcal{S}}$ corresponding to the coded bits $\boldsymbol{x}_{\mathcal{S}}$ are also zero, i.e., $\boldsymbol{u}_{\mathcal{S}} = \boldsymbol{x}_{\mathcal{S}} = \mathbf{0}$. Note that similarly to incapable bits, the number of overcapable bits is equal to the number of shortened bits. The shortening set $\mathcal{S}$ in the 5G standard is given by $\mathcal{S} = (N-S, \ldots, N-1)$, the last S indices of $[0, N-1]$ in natural order (alternatively in bit-reversal order [508]). Fig. 67 illustrates the circular buffer used in 5G based on the natural order of the bit indices.

M. POLAR CODED MODULATION (PCM)

Let $W : \mathcal{S} \rightarrow \mathcal{Y}$ be a 2^m-ary channel with input symbol from the constellation set $\mathcal{S}$ of order $|\mathcal{S}| = 2^m$, and output alphabet $\mathcal{Y}$. Each symbol in the constellation is labelled with a binary m-tuple, and we say that the symbols in this constellation have m bit levels. A binary labeling rule, denoted by $\mathcal{L}$: $\{0, 1\}^m \rightarrow \mathcal{S}$, maps binary m-tuples bijectively to the 2^m input symbols $s \in \mathcal{S}$. The popular labelling rules are gray labelling and set partitioning. In the following, we review the two major polar coded modulation schemes.

1) MULTILEVEL POLAR CODED MODULATION (ML-PCM)

In an ML-PCM, there exist m component polar codes, each of length N. At the encoding stage, a binary vector of length mN consisting of both data bits and frozen bits that are placed together according to an overall rate profile is divided into m vectors of equal length. Then, each N-length subvector is polarly encoded as a component polar code. Let $\boldsymbol{x}_j = (x_{j,1}, x_{j,2}, \ldots, x_{j,N})$ denote the coded bits of the j-th component code for $j = 1, 2, \ldots, m$. Then, at the modulation stage, the m-tuple $(x_{1,i}, x_{2,i}, \ldots, x_{m,i})$ for $i = 1, 2, \ldots, N$ is mapped to a constellation symbol for transmission. In this way, in every channel use, the bits of each component code only appear at the corresponding single-bit level. For example, in channel use $i = 1, \ldots, N$ modulated with 4QAM constellation that maps every 2 bits to one symbol point, we will have two component polar codes $\boldsymbol{x}_m, m = 1, 2$ where every bit pair of $x_{1,i}x_{2,i}$ is mapped to a symbol.

In the receiver, the demodulation and decoding are performed for each bit level sequentially, as shown in Fig. 69 (right). In the previous example, the reliability information corresponding to the first bit level $(m = 1)$ of all received symbols is computed as $y_{1,1}y_{1,2}$ and the associated component code is decoded. Then the decoding results are used for demapping $y_{2,1}y_{2,2}$ corresponding to the bit level $m = 2$ and, consequently, decoding. The decoding results are passed for demapping and decoding the next bit levels. Observe that set partitioning better suits this approach, which is called *multi-stage* demodulation and decoding.

As observed above, in a multilevel coding approach, the raw channel W is effectively decomposed into m binary sub-channels while preserving their mutual information. Let us denote this channel decomposition, which is called the sequential binary partition (SBP) [510], as

$$\psi : W \rightarrow \left\{B_\psi^{(1)}, B_\psi^{(2)}, \ldots, B_\psi^{(m)}\right\},$$

where $B_\psi^{(j)} : \{0, 1\} \rightarrow \mathcal{Y} \times \{0, 1\}^{j-1}$ is the binary sub-channel corresponding to the bit level $j = 1, 2, \ldots, m$. Therefore, an M-ary channel W is divided into m bit levels (or sub-channels) $\mathrm{B}_\psi^{(i)} (0 \leqslant i < m)$ that are B-DMCs as long as W is a DMC. In SBP, each sub-channel $B_\psi^{(j)}$ has the knowledge of both the channel output $y \in \mathcal{Y}$, and their previous bit levels obtained through decoding. Mutual information between channel input $\mathcal{S}$ and channel output $\mathcal{Y}$ of W, assuming equiprobable source symbols, is referred to as the coded modulation capacity $C_{\text{cm}}(\mathrm{W})$ [511]:

$$C_{\text{cm}}(\mathrm{W}) := I(X; Y) = \sum_{i=0}^{m-1} I\Big(\ \mathrm{B}_\psi^{(i)}\Big).$$

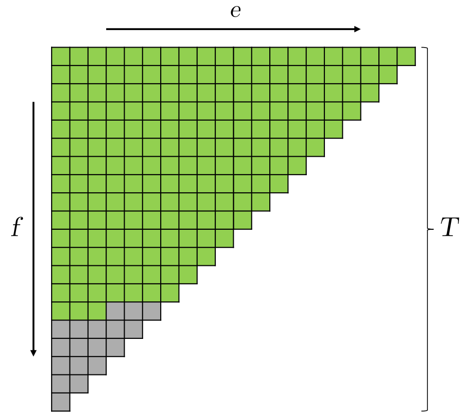


FIGURE 68. Triangular structure for channel interleaving in 5G.

This capacity does not depend on a specific labeling rule $\mathcal{L}$. Note that according to [369, Th. 1], as the blocklength of component codes increases, the polar component codes approach the bit level capacities $I(\ \mathrm{B}_\psi^{(i)})$.

A drawback of the MLC approach for practical use lies in the need to use several relatively short component codes with different rates for various bit levels. Given the length requirement of the power of twos in polar coding, this constraint adds to the disadvantages of the multilevel coding approach.

2) BIT-INTERLEAVED POLAR CODED MODULATION (BI-PCM)

The rate-matched binary coded bits $\boldsymbol{e}$ of length E are permuted by an interleaver. The channel interleaver in the 5G standard is formed by an isosceles triangular structure of length T bits, where the interleaver depth T, the maximum separation between two consecutive bits, is obtained by $T = \lceil \frac{\sqrt{8E+1}-1}{2} \rceil$. The encoded bits in the vector $\boldsymbol{e}$ are written in the rows of the triangular structure shown in Fig. 68, while the interleaved vector denoted by $\boldsymbol{f}$ is obtained by reading the bits column-wise, skipping the NULL entries. The triangular structure can be represented as a matrix $\boldsymbol{V}$ of size $T \times T$, where the entries are formed as

$$V_{i,j} = \begin{cases} \text{NULL} & \text{if } i + j \geqslant T \text{ or } r(i) + j \geqslant E \\ e_{r(i)+j} & \text{otherwise} \end{cases}$$

where $r(i) = \frac{i(2T-i+1)}{2}$.

Then, each subblock of m bits of vector $\boldsymbol{f}$ is mapped to a constellation symbol in $\mathcal{S}$ for channel transmission. When a symbol is received, the demodulator on the other end disregards the correlation between bit levels and calculates soft information for all bit levels based only on the channel observation.

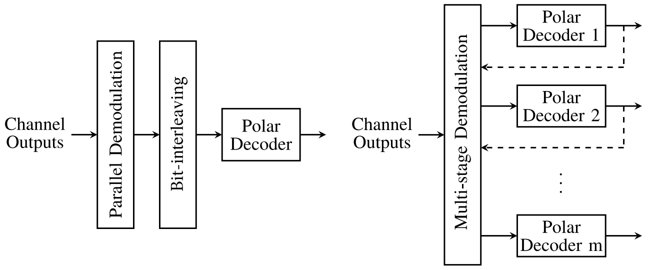


FIGURE 69. Multi-stage decoding vs the decoding of bit-interleaved codewords.

In a BI-PCM scheme, the channel W is decomposed into m binary sub-channels that are viewed as independent channels by the receiver. This channel transform is called *parallel binary partition* (PBP) [510] denoted as

$$\varphi : W \to \left\{B_{\varphi}^{(1)}, B_{\varphi}^{(2)}, \ldots, B_{\varphi}^{(m)}\right\},$$

where the channel W is mapped to a set of mutually independent binary sub-channels $B_{\varphi}^{(j)} : \{0, 1\} \to \mathcal{Y}, j = 1, 2, \ldots, m$ for the j-th bit level. These sub-channels have symmetric capacities of

$$I\left(\ \mathrm{B}_{\varphi}^{(i)}\right) := I(B_i; Y).$$

Obviously, when compared to the corresponding sub-channel in SBP, for all pairs of sub-channels, the following holds:

$$\begin{aligned} I\left(\ \mathrm{B}_{\varphi}^{(i)}\right) &= I(B_i; Y) \\ &\leqslant I(B_i; Y \mid B_0, \ldots, B_{i-1}) = I\left(\ \mathrm{B}_{\varphi}^{(i)}\right) \end{aligned} \quad (195)$$

In PBP, each sub-channel $B_{\varphi}^{(j)}$ only uses the channel output $y \in \mathcal{Y}$, while SBP requires knowledge of previous bits $\{0, 1\}^{j-1}$. The commonly used labeling rule for sub-channels in BI-PCM is gray labeling which generates bit levels that are as independent as possible, assuming the sub-channels correspond to bit levels with symmetric capacities that do not differ significantly.

After demodulation, as shown in Fig. 69 (left), the soft information of all mN bits is de-interleaved, and fed to the de-rate-matcher. Note that to use a single polar decoder for BI-PCM, the order m of the constellation has to be a power of 2.

Unlike BICM where each codeword is independently modulated and transmitted in a single time slot, delayed BICM (D-BICM) [512] divides each codeword into m subblocks and modulates the subblocks from both the previous time slots and the current time slot onto the same signal sequence. The receiver starts decoding when all sub-blocks of a codeword are received. Next, the decoded delayed subblocks in the current time slot are used as a-priori information to improve the detection of the undelayed subblocks in the succeeding time slots.

TABLE 19. Evaluation of the frame error rate performance versus E_b/N_0 for variants of polar codes with two different constructions.

Channel	AWGN		
Modulation	BPSK		
Coding variant	CRC-Polar	PAC	CRC-PAC
Constructions	5G, 5G-RM		
Code rate, R	$1/2$		
Info. length, K (bits w/o CRC)	$64, 512$		
Decoding Algorithm	SCL, $L = 32$		

TABLE 20. Evaluation of the frame error rate performance versus E_b/N_0 for variants of polar codes at three different rates.

Channel	AWGN		
Modulation	BPSK		
Coding variant	CRC-Polar	PAC	CRC-PAC
Constructions	5G, 5G-RM		
Code rate, R	$1/4, 1/2, 3/4$		
Info. length, K (bits w/o CRC)	$64, 512$		
Decoding Algorithm	SCL, $L = 8$		

TABLE 21. Evaluation of the frame error rate performance versus E_b/N_0 for variants of polar codes under different decoding algorithms.

Channel	AWGN				
Modulation	BPSK				
Coding variant	CRC-Polar				PAC
Constructions	5G				
Code rate, R	1/2				
Info. length, K (bits w/o CRC)	$64, 512$				
Decoding Algorithm	SC	BP	SCL	SCLF	SCL

N. ERROR CORRECTION PERFORMANCE

In this section, we compare known variants of polar codes, namely CRC-polar codes and PAC codes (with and without CRC concatenation). These results are obtained for AWGN channel and binary phase shift keying (BPSK) modulation. For this purpose, we choose two codes; the short code (128, 64) and the medium-length code (1024, 512). They are within the length range used for 5G, as mentioned in Table 7. The CRC and convolutional polynomials used for precoding in CRC-polar and PAC codes are $g_{\mathrm{CRC11}}(D) = D^{11} + D^{10} + D^9 + D^5 + 1$ and $g_{\mathrm{Conv6}}(D) = 1 + D^2 + D^3 + D^5 + D^6$, respectively. To have a thorough comparison, we consider three sets of setups where in each setup, only one coding factor is changing. These factors are:

- coding scheme and code construction: As Table 19 shows, we choose CRC-polar, PAC and CRC-PAC constructed using 5G reliability sequence and its modified version called 5G-RM (see Section VII-E),
- code rate: $1/4, 1/2, 3/4$, as Table 20 states, and
- decoding algorithms according to the setup in Table 21.

Note that there are quite a number of methods to construct a code. However, they are either not explicit or they have a high complexity, which makes them impractical and not a good candidate to be adapted to a standard. Similarly, there are other decoding algorithms that we do not use

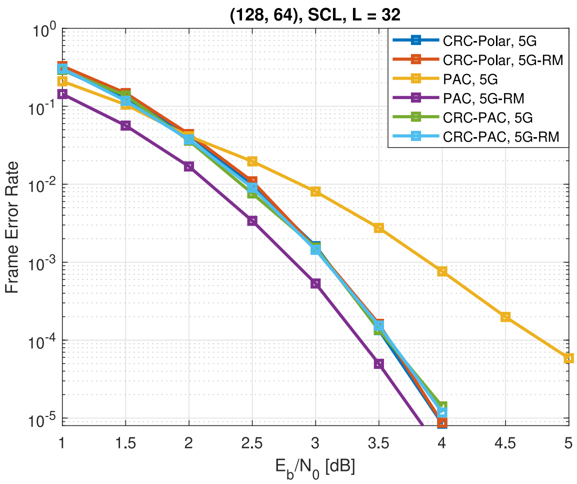


FIGURE 70. FER comparison between code constructions of polar codes and PAC codes with $N = 128$, $K = 64$.

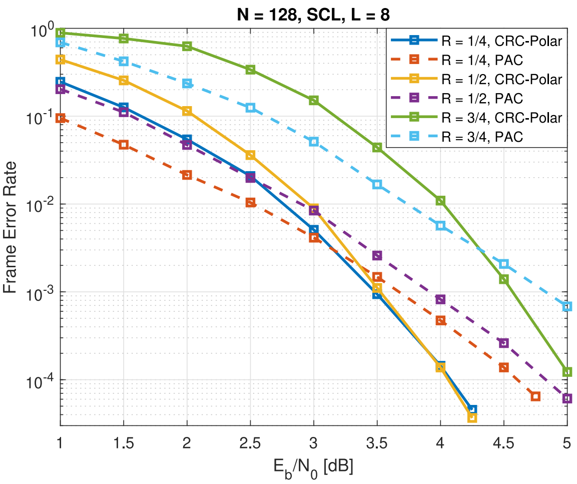


FIGURE 72. FER comparison between rates of polar codes with $N = 128$ under 5G construction.

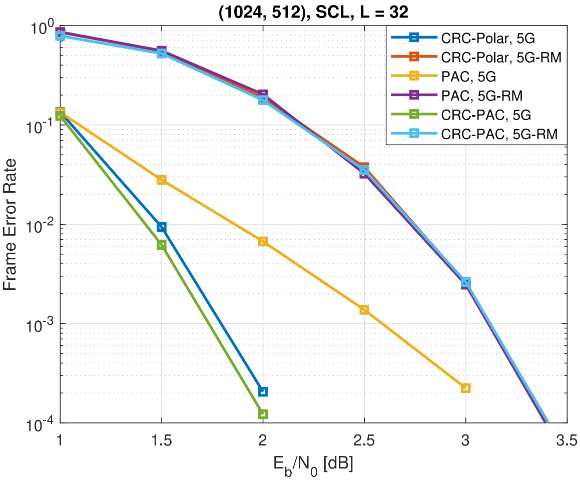


FIGURE 71. FER comparison between code constructions of polar codes and PAC codes with $N = 1024$, $K = 512$.

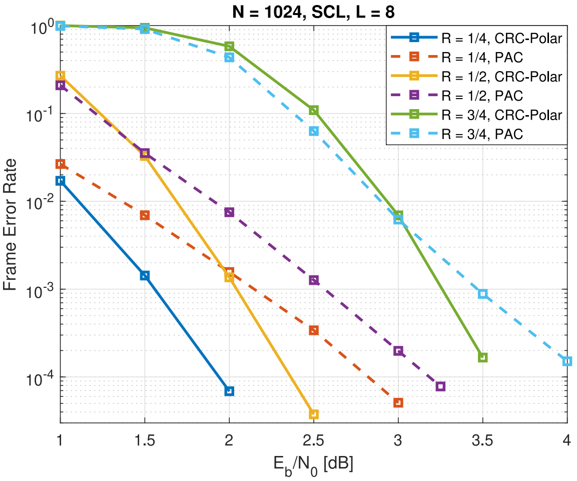


FIGURE 73. FER comparison between rates of polar codes with $N = 1024$ under 5G construction.

in this comparison; however, we have discussed them in Section VII-K. In the following, we discuss each scenario in the same order as above.

Figs. 70 and 71 illustrate the frame (or block) error rate for variants of the polar codes of (128, 64) and (1024, 512), respectively. In this scenario, summarized in Table 19, we use the SCL decoder with the list size $L = 32$ for all codes. As can be seen in Fig. 70, for the relatively short code (128, 64), the PAC code constructed with the 5G-RM construction outperforms the PAC codes with the 5G construction and the CRC-polar codes. Note that the 5G construction gives the minimum distance of $d = 8$ for both PAC and CRC-polar codes, while the 5G-RM's minimum distance is $d = 16$. Furthermore, CRC concatenation with short codes is punishing as it may decrease the minimum distance and utilizes more low-reliability sub-channels. When it comes to medium-length codes such as (1024, 512) in Fig. 71, the 5G-RM construction performs poorly. The reason lies in employing severely bad sub-channels though the corresponding rows of $\boldsymbol{G}_N$ have weight that is larger than minimum weight/distance. Therefore, it is recommended to use the 5G construction for PAC coding in medium blocklength. To obtain a comparable result with CRC-polar codes in this range of blocklength, we need to concatenate PAC codes with CRC.

Now, let us compare the considered codes under different code rates. Figs. 72 and 73 demonstrate the FER performance of the CRC-polar and PAC codes for the two codes of (128, 64) and (1024, 512), respectively. In this scenario, summarized in Table 20, we use the SCL decoder with list size 8 and 5G construction for all codes. As can be seen, PAC codes constructed with the 5G sequence behave differently for short codes and medium-length codes. A short PAC code can outperform the corresponding CRC-polar code at low SNR regimes. At high code rates, this is also the case at medium SNR regimes. Note that short PAC codes

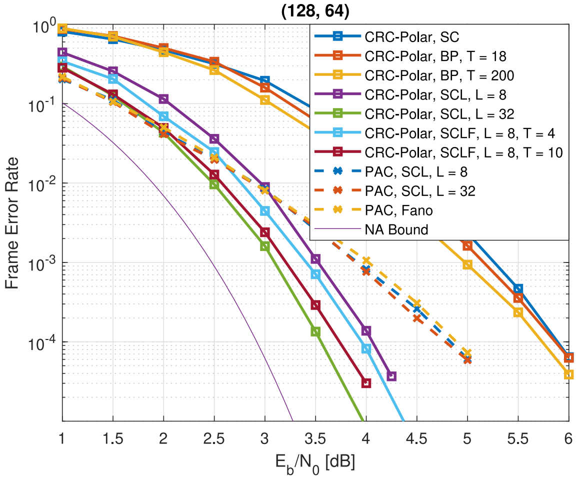


FIGURE 74. FER comparison between decoders for polar codes and PAC codes with $N = 128$, $K = 64$ under 5G construction.

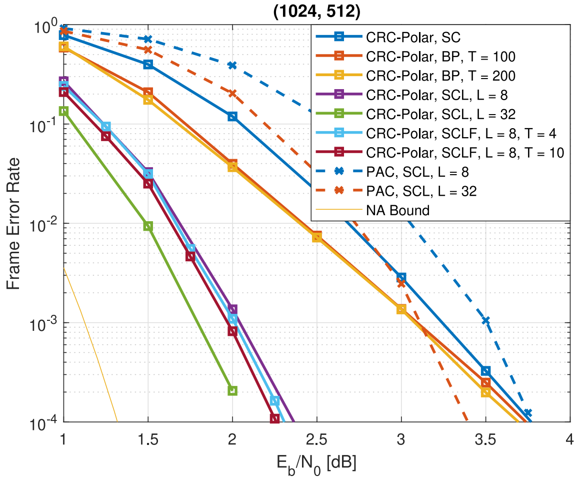


FIGURE 75. FER comparison between decoders for polar codes and PAC codes with $N = 1024$, $K = 512$ under 5G construction.

can outperform CRC-polar codes when constructed with 5G-RM, as we observed in Fig. 70; however, here we use 5G for all PAC codes. On the contrary, at medium blocklengths, PAC codes cannot compete with CRC-polar and we need a CRC concatenation to improve it, as we observed in Fig. 71. However, there is one exception for high-rate PAC codes.

Note that there is an approach [377], [378] based on removing many minimum-weight codewords by a simple modification of the 5G-based construction of PAC codes so that it can outperform CRC-polar codes in low and medium SNR regimes for medium blocklengths and all rates.

Finally, we compare the error correction performance of the CRC-polar codes and PAC codes under various decoding algorithms. Figs. 74 and 75 compare the block (or frame) error rate of the CRC-polar code (128, 64) under the SC, BP, SCL, and flipping SCL decoding algorithms with different decoding parameters of list sizes $(L = 8, 32)$, iterations (T), or both. In this scenario, summarized in Table 21, we use the 5G construction for all codes. As the figures demonstrate, the SCL decoding algorithm with a large list $l = 32$ seems to be the best option as it outperforms the rest. Under resource constraints such as limited silicon area, SCLF with list size $L = 8$ could alternatively be used at the cost of longer latency due to multiple iterations $(T = 4, 10)$. Note that the PAC codes demonstrated here are constructed with 5G reliability sequences, not 5G-RM. As a result, it does not outperform a short CRC-polar code.

O. PERFORMANCE OF DECODING HARDWARE ARCHITECTURES

In this section, we review the hardware architectures designed for the decoders discussed in Section VII-K, and compare them in terms of their throughput, latency, energy efficiency, scalability, etc. Although the most popular decoder for polar codes is SC list decoding as it provides a competitive error correction performance, due to its sequential nature that limits the throughput and high computational complexity, alternative decoders have been adapted to polar codes, which admit parallelism in design, computationally simpler, etc. Taking into account the tradeoff between performance and other parameters, these alternative decoders can be used in different scenarios depending on their requirements.

The architectures of hardware implementations can be categorized on the basis of various scheduling structures. Fig. 76 provides scheduling examples for decoding a single block of a polar code with length N, considering clock cycle (CC) based execution. The depicted scheduling covers SC-based and BP-based architectures, and unrolled architecture (used for SC decoder), all applied to a polar code with $N = 8$, where $L_i, 0 \leqslant i \leqslant n$ represent the stages in the factor graph, while f and g are the computations within the SC decoder. For the SC-based decoder shown in Fig. 76.a, the determination of each bit relies on the LLR propagated from the preceding node. The scheduling is influenced by the time complexity of the decoder, resulting in a successive architecture that requires $2N - 2$ clock cycles (CC) to decode a single block. As described in Section VII-K2, the BP-based decoders operate in stages. Consequently, the scheduling entails $2n \cdot t$ CC, as illustrated in Fig. 76.b, where t represents the iteration number. In the unrolled architecture, all the necessary processing elements (PEs) for the decoding process are listed. The scheduling of this architecture is illustrated in Fig. 76. For decoding the first block, it has the same latency as the structure shown in Fig. 76.a. However, all the PEs are deeply pipelined. Once a PE finishes the computation of one block, it will pass the information to the next PE and start processing the information of the next block. As a result, following the initial block, the unrolled architecture can generate the result of decoding one block for every CC, leading to notably higher throughput than the other two architectures at the cost of much higher demand of the resources.

	block 1														block 2		
CC	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	...
L_2	f							g									
L_1		f			g				f			g			...	...	
L_0			f	g		f	g			f	g		f	g			

a) An example scheduling for SC-based architecture.

	block 1																
	iteration 1						iteration 2						iteration 3				
CC	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	...
Stage	L_2	L_1	L_0	R_0	R_1	R_2	L_2	L_1	L_0	R_0	R_1	R_2	...				

b) An example scheduling for BP-based architecture.

CC	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	...
block 1	f	f	f	g	g	f	g	g	f	f	g	g	f	g			
block 2		f	f	f	g	g	f	g	g	f	f	g	g	f	g		
block 3			f	f	f	g	g	f	g	g	f	f	g	g	f	g	

c) An example scheduling for the unrolled architecture of SC decoder.

FIGURE 76. Examples of scheduling based on clock cycle (CC) for an underlying polar code of length N = 8.

1) SUCCESSIVE CANCELLATION-BASED ARCHITECTURE

In [513], three hardware structures were developed to implement the original successive cancellation (SC) decoder. An example of a top-level architecture for the SC-based decoder is shown in Fig. 77, which includes the main logic modules as follows:

- *Control logic:* The controller generates the control signals for the sub-modules to initiate their operations.
- *RAM:* The memory block stores values of the propagating LLRs for subsequent usage by the decoder. The RAM addresses are acquired from the control logic.
- *Process element (PE):* Each PE computes a pair of functions (i.e., f and g in (188) and (189)) of the decoder. The number of PEs employed depends on the specific architecture design and scheduling.
- *Buffers:* Read-and-write operations of the RAM follow the clock schedule. Therefore, in situations where certain data require immediate input or reuse, buffers are employed to act as an intermediary.
- *$\hat{u}$ logics:* The values of $\boldsymbol{u}$ are determined by hard decisions of the LLRs as computed in equation (186). Moreover, they are designed to update the value of the bit for the node, i.e., update the $\hat{\beta}$ in equation (189), so that it has the value of the partial sum of the previously decided bits.

A semi-parallel structure [514] was introduced to enhance resource utilization of the PEs, while a combinational logic structure [515] focused on reducing the energy consumption of intermediate LLR calculations in the SC decoder. The simplified SC (SSC) [475] and 2-bit-SC [516] were introduced to minimize latency by optimizing the calculations of frozen bits. These approaches aimed to reduce the decoding stages, resulting in lower latency and improved throughput. Building upon this idea, Fast-SSC decoders were proposed in [471], [472], [517], [518], [519], which further optimized the decoding tree by categorizing special nodes, leading to improved throughput. Additionally, SC flip (SCF) decoders were implemented in [520], [521], [522], achieving better error correction performance at the cost of lower throughput.

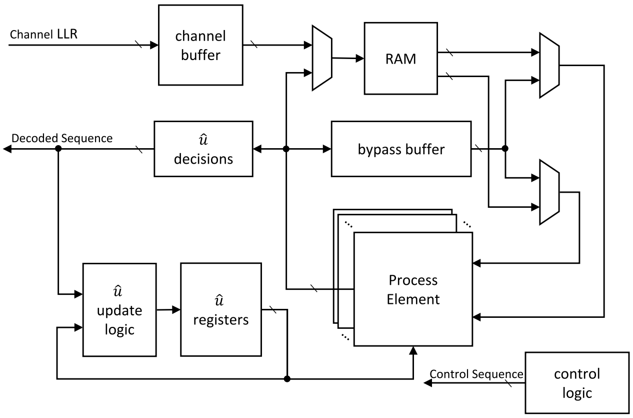


FIGURE 77. An example of a top-level architecture for the SC decoder [514].

Based on iterative message passing following the successive architecture, hardware implementations for the soft-output cancellation (SCAN) [523], [524] and Fast-SCAN decoders [525] were designed. These decoders showed better error correction performance than the fundamental SC decoders. However, because of their combined successive and iterative architectures, they exhibited increased latency and reduced throughput.

Hardware architectures of the SCL and CRC-aided SCL (CA-SCL) decoders were designed in [526], [527], [528]. The implementations of multi-paths in the SCL decoder either require large resources or slow down the throughput to trade with its competitive error correction performance. The sorters were designed to accelerate the path-pruning of the SCL decoder to improve throughput [529]. Node classification schemes and latency reduction schemes were adopted and improved for the SCL decoder in [473], [530], [531], [532], [533], [534], showing significant improvements in latency, throughput, and area efficiency.

An architecture for automorphism (or permutation) ensemble decoding was presented in [535]. The AED employs an ensemble size of SC decoders implemented in parallel, similar to the SCL decoder, but without the path-sorting and pruning procedures, resulting in lower latency and higher throughput compared to SCL-based decoders.

In [536], an SC-based decoder was adopted and designed for multi-kernel polar codes to meet specific requirements in practical applications, offering flexible code lengths, code rates, and kernel sequences. Subsequent improvements in latency and throughput were achieved by implementing the Fast-SSC decoder [537] and a combinational logic structure [538]. These advancements further enhanced the performance and efficiency of hardware implementations of polar codes.

Combining the SC decoder with the Fano decoder, a hardware implementation was developed for PAC codes in [416]. While the Fano decoder offers good error correction performance and a small required area for short codes, its high-complexity iterative decoding processes result in extremely high latency and low throughput.

2) BELIEF PROPAGATION-BASED ARCHITECTURE

BP decoders offer advantages over SC-based decoders as the operations on the factor graphs can be executed in parallel, providing the potential for higher throughput and lower latency [539]. The reduction in latency compared to the SC-based decoder becomes more pronounced as the code length increases. However, one of the drawbacks of BP-based decoders is the uncertainty in the required number of iterations. In worst-case scenarios, when BP-based decoding reaches the maximum number of iterations, it can result in significantly low latency and throughput. Various algorithms and hardware architectures, such as bi-directional, folding, double-column structures, and early stopping criteria [540], [541], [542], [543], [544], [545] have been proposed to reduce latency and improve the throughput of BP decoders. Also, it was shown in [546], [547] that non-uniform quantization of messages can improve the BLER performance over uniform quantization in hardware implementation. Additionally, special node designs have been adopted to further enhance throughput [548]. The BP-flipping (BPF) [549], [550] and BP list (BPL) [551] were introduced, demonstrating error correction performance comparable to SCL decoders, but with higher throughput.

3) UNROLLED ARCHITECTURE

Unrolled architectures deploy all the PEs for individual nodes within the decoder tree of a given polar code. This approach maintains consistent latency while achieving the highest throughput compared to other architectures, albeit with notable resource requirements, allowing each PE to handle a distinct block. Unrolled architectures for the fast SSC decoder and the combined SC-majority logic (MJL) decoder were designed in [552], [553] to boost the throughput with larger area requirements, addressing the challenges of low throughput in successive structures. Unrolled structures of the SCL decoder were implemented in [554], [555], [556]. Moreover, unrolled BP decoders with a fixed number of iterations have been implemented to achieve ultra-high throughput in [356].

4) COMPARISON

The performance of various polar decoders with state-of-the-art hardware architectures in [416], [525], [531], [535], [536], [550], [551], [556], [557] is presented and compared in Table 22. The results are presented relative to the 28nm technology for fair comparisons. The unrolled structure of the CA-SCL decoder [556] achieves an impressive throughput of about 500Gb/s at the cost of a proportionally larger area requirement compared to other structures. For polar codes with a mother code length of $N = 1024$ bits, the parallel decoder in the BPL decoder [551] allows it to achieve a power gain of 0.6 dB in error correction performance over the EBPF decoder [550] with higher average throughput and lower average latency. However, the parallel structure comes with a larger required area. Fast-SSCL [531] and BPL decoders demonstrate the best error correction performance. The implementations of the BP-based decoders show higher throughput than the Fast-SSCL decoder due to their highly paralleled decoding structure, requiring fewer average stages than the successive structure of the SC-based decoder. However, the iterative decoding structure leads to much lower throughput in worst-case scenarios. The Fast-SCANF decoder [525], based on iterative message passing and a successive decoding structure, achieves lower throughput while requiring a smaller area compared to the SCL and BPL decoders. The implementation of the multi-kernel polar codes shows inferior error correction performance and throughput compared to the state-of-the-art SCL-based and BP-based decoders, while it provides smaller area requirements and flexibility in the code rates and code lengths for the polar codes. For hardware implementations that focus on shorter codes, the Fano decoder [416] demonstrates the best error correction performance at the cost of significantly low throughput and high latency due to its high complexity. The multiple paralleled SC decoders enable the AED [535] to have a reasonable error correction performance and throughput, but with a larger required area compared to Fano decoder.

P. FUTURE DIRECTIONS FOR POLAR CODES

Polar coding was proposed about 1.5 decades ago. However, throughout this period, new ideas and initiatives have kept this scheme the focus of research attention in the field of channel coding. Among the major coding schemes, perhaps this is the only scheme that still is gaining popularity in

TABLE 22. Hardware implementation results for polar and PAC codes decoders.

Implementation	[531]	[556]	[553]	[550]	[356]	[525]	[536]	[535]	[416]
Algorithm	Fast-SSCL	CA-SCL*	SC-MJLL*	EBPF	BPL	Fast-SCANF	Multi-Kernel	AED	Fano
Process [nm]	65	28	45	65	28	40	65	12	28
Code	(1024, 512)	(1024, 512)$^{\top 1}$	(1024, 854)	(1024, 512)	(1024, 512)	(1024, 512)	(768, 384)	(128, 60)	(128, 64)$^{+}$
Quantization	6	6	5-to-1	7	7	6	6	-	7
List/Attempt	4	2	1	20	32/50	10	1	8	2^{18}
SNR@BLER=10^{-4}	2.65	2.94	5.54	3.25	2.65	3.07	3.48	3.71	3.14
Area [mm^2]	1.822	7.89	2.4	3.11	0.87	0.44	0.46	0.17	0.059
Frequency [MHz]	840	494	500	319	1333	980	1110	498	500
W.C. Latency [μs]	-	-	-	51.2	0.34	-	-	-	524
Avg. Latency [μs]	0.64	0.31	0.08	0.28$^{\diamond 1}$	0.04$^{\diamond 1}$	0.46$^{\diamond 2}$	2.09	0.022	1.68$^{\diamond 4}$
W.C. T/P [Gb/s]	-	-	-	0.02	0.09	-	-	-	0.074
Coded T/P [Gb/s]	1.61	506	512	3.72$^{\diamond 1}$	25.63$^{\diamond 1}$	2.04$^{\diamond 2}$	0.358	63.7	0.037$^{\diamond 4}$
Area Eff. [Gbps/mm^2]	0.883	64.13	213.33	1.2	29.46	4.63	0.78	375.1	0.646

Normalized to 28nm*

Coded T/P [Gb/s]	3.74	506	823	8.64	25.63	2.91	0.83	27.3	0.037
Area [mm^2]	0.338	7.89	0.94	0.577	0.87	0.216	0.085	0.926	0.059
Area Eff. [Gbps/mm^2]	11.07	64.13	872	14.97	29.46	13.47	9.76	29.48	646

$*$ These work design unrolled architectures.

$\top_{1,2}$ These works employ 6-bit and 11-bit CRC codes for the polar codes respectively.

$+$ This work is developed for PAC codes.

$\diamond_{1,2,3,4}$ Average results reported at SNR = 4, 3, 7.5, 3.5 dB repectively. Worse case is not discussed for $\diamond_1$.

$\star$ Normalized to 28nm technology: Area $\propto \alpha^2$ and frequency $\propto 1/\alpha$, where α is the scaling factor to 28nm.

research, albeit at a slower rate, as Fig. 84 indicates. The popularity of polar codes is mainly due to their superior error correction performance for short codes at a reasonable complexity cost. Nevertheless, there is still room to further improve the performance through pre-transformation and code construction, and reduce the complexity of the decoding algorithms. In the following, we suggest three general directions that researchers can follow.

Concatenation and pre-transformation of polar codes, such as CRC-polar codes and PAC codes, have remarkably improved the performance of polar codes. The available concatenation and transformation schemes can be further improved towards improving the distance properties of underlying codes. For instance, an attempt to improve pre-transformation in PAC codes was made in [430].

Construction of polar codes and their variants considering both the error coefficient (or weight distribution in general, based on the weight structure of polar codes studied extensively in [378], [380], [558], [559]) and sub-channels reliability with a low-complexity approach is desired. As discussed in Section VII-E, reliability-based code construction is optimal for SC decoding. However, (near) ML decoding performance depends on the weight distribution of the code. Fig. 78 shows that improving/reducing the number of minimum weight codewords, or error coefficient $A_{d_{min}}$, improves the block error rate, in particular at high SNR regimes. However, reducing $A_{d_{min}}$ beyond some extent deteriorates the performance due to excessive utilization of bad sub-channels. Hence, we need to consider the trade-off between the weight distribution and the selection of the

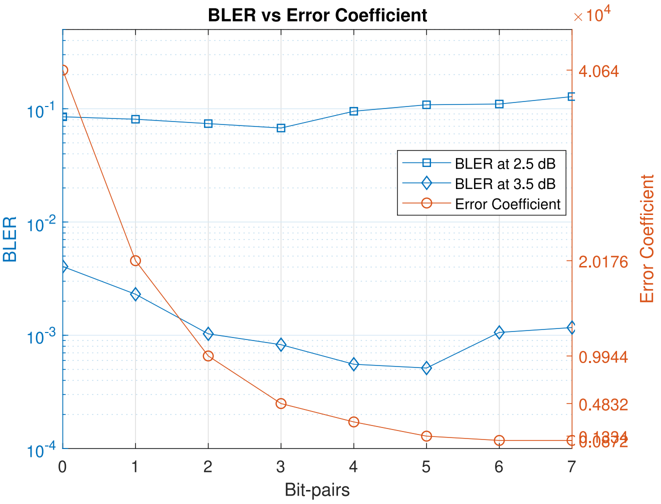


FIGURE 78. BLER at two E_b/N_0's versus error coefficient $A_{d_{min}}$ of PAC code of (512,384) under SCL decoding with $L = 16$. Bit-pairs are the number of bit indices added to/removed from $\mathcal{A}$ which are the indices of the most reliable sub-channels [378].

TABLE 23. Evaluation of the frame error rate performance versus SNR of coding schemes.

Channel	AWGN		
Modulation	QPSK		
Coding Scheme	Turbo	LDPC	Polar
Code rate R	$1/5, 1/3, 1/2, 2/3, 8/9$		
Decoding Algorithm	Max-Log-MAP, Max Iter = 8	Adjusted Min-Sum, Max Iter = 25	SCL, L = 32
Info. length K (bits w/o CRC)	$100, 400, 1000, 4000, 8000$		

most reliable sub-channels. An attempt to achieve this goal was made in [378], [402]. However, this direction requires further attention.

The complexity of SC-based decoding algorithms has been a concern in achieving low latency, despite the fact that polar codes and their variants make high-reliability communication possible. Improving the existing algorithms and designing novel decoding still remain a direction to explore.

As mentioned above and as we can see from the numerical result in Section VIII, polar codes cannot outperform other coding schemes at long block lengths. This is one of the directions that can be investigated to improve the performance of polar codes for longer codes and reduce their decoding complexity, which is another concern for longer codes. This improvement can be achieved through different code construction for longer codes or by a different decoding strategy. However, further studies are required on the properties and differences between long codes and short codes.

VIII. PERFORMANCE COMPARISON OF TURBO, LDPC, AND POLAR CODES

We evaluated the performance of various coding schemes under different decoders, constructions, interleavers, component codes, etc., in Sections V, VI, and VII. We consider the information lengths $K = 100, 400, 1000, 4000, 8000$ bits and the code rates $R = 1/5, 1/3, 1/2, 2/3, 8/9$. Results are obtained from 3GPP report [560], [561]. Note that information block lengths $K = 96, 992$ are employed for turbo codes, whereas $K = 100, 1000$ are used for LDPC and polar codes. For LDPC and polar codes, we use the 5G standard's constructions. The turbo codes in the comparison are enhanced turbo codes with tail-biting termination from [35], [562], where interleavers and puncturers therein are adopted. The extrinsic information of the Max-log-MAP decoder is scaled as $s^{(1)} = 0.6$, $s^{(\ell_{\max})} = 1$, and $s^{(\ell)} = 0.7, \forall \ell \in \{2, \ldots, \ell_{\max} - 1\}$, where $\ell_{\max}$ denoting the maximum number of iterations. The setup of the numerical evaluations is summarized in Table 23. Note that puncturing, shortening, and repetition techniques are used to have the same rate (or the same block lengths) for all three codes.

As can be seen in Figs. 79, 80, and 81, polar codes outperform LDPC and Turbo codes when the length of the information block is $K \leqslant 400$ bits. For $K = 1000$ bits, the performance of polar codes is inferior compared to that of LDPC codes at rates $R \leqslant 1/3$. As the length of the information block increases to $K = 4000$ and 8000 bits, LDPC codes outperform Turbo and polar codes at all rates and the power gain becomes more significant as K increases.

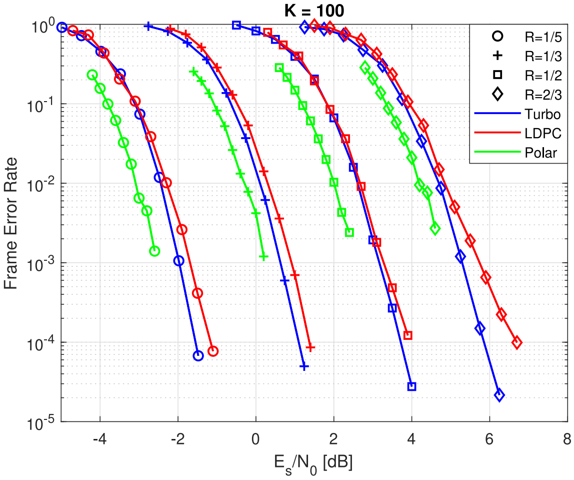


FIGURE 79. FER comparison between codes for information block $K = 100$ at different code rates.

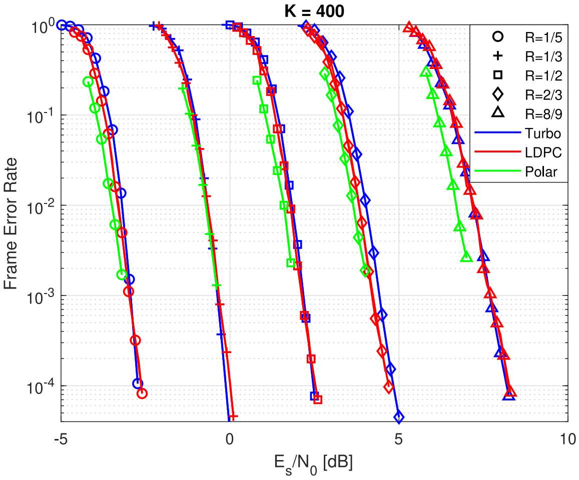


FIGURE 80. FER comparison between codes for information block $K = 400$ at different code rates.

Note that we did consider recent improvements of the coding schemes discussed earlier, in particular with regard to the polar coding scheme. Recent developments are still the subject of research and need to be investigated and improved further to the level needed for adaptation to a standard. Nevertheless, we have discussed and evaluated them in their own section in this paper.

IX. OTHER CODING SCHEMES

The binary codes discussed in the previous sections, i.e., LDPC codes, polar codes, convolutional codes, and turbo codes are the coding schemes that have been adapted in standards, and hence have proved their merits. However, other coding schemes can also be considered. Fig. 84 shows

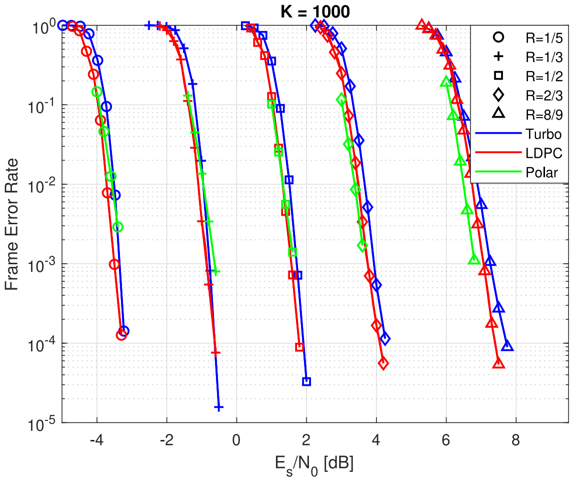


FIGURE 81. FER comparison between codes for information block $K = 1000$ at different code rates.

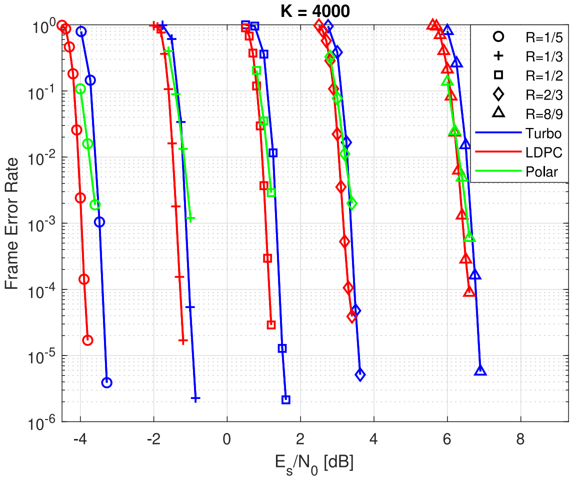


FIGURE 82. FER comparison between codes for information block $K = 4000$ at different code rates.

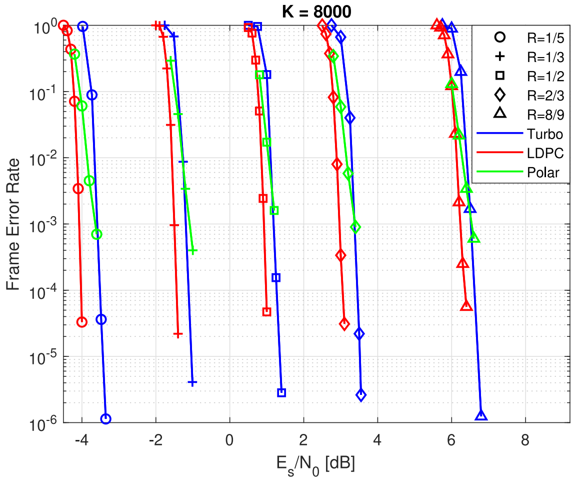


FIGURE 83. FER comparison between codes for information block $K = 8000$ at different code rates.

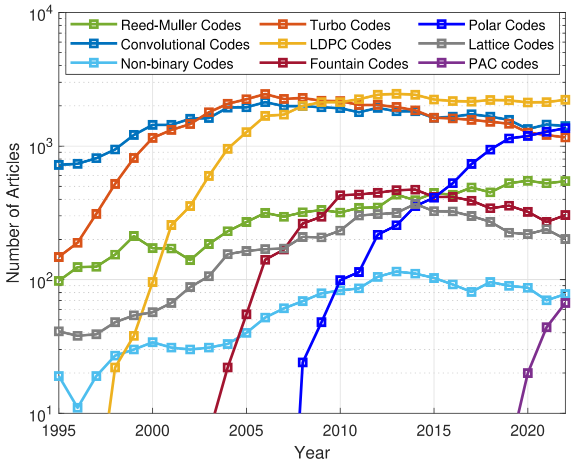


FIGURE 84. The trend of number of articles published annually (extracted from Google Scholar hits).

the trend of the number of publications throughout the last 30 years.[6] As can be seen, while publications on polar codes have exponentially increased over the first decade of their invention, other codes such as LDPC codes have retained their popularity.

In this section, we briefly review other potential coding candidates which are included in Fig. 84 and discuss the major challenges to adapt them to a mobile communications standard, including decoding complexity, real-time requirements, power consumption, scalability in finite block length, and/or compatibility and interoperability with existing systems. Additionally, while the focus of this paper is on physical layer coding, we consider the option of application layer channel coding in the following.

A. APPLICATION LAYER CHANNEL CODING

Application layer (AL) channel coding, predominantly based on fountain codes and Reed-Solomon codes, has been considered in the literature for a variety of applications such as multimedia broadcasting and streaming [564], [565], IoT [566], [567], deep space communications [568], etc. In AL channel coding, additional packets of application-specific data are produced as redundant packets for packet recovery at the application layer, while physical layer channel coding involves mechanisms for error detection and correction in raw data bits, specific to the characteristics of the transmission medium.

One major advantage of this scheme is that operating at a higher layer in the protocol stack makes it easier to implement and manage the coding scheme, particularly in software-based systems, whereas physical layer coding may involve specialized hardware components and signal processing. However, a primary challenge of the scheme is the increase in latency. Additionally, larger memory spaces are required to handle larger source packets. Other challenges are as follows. 1) It is difficult for AL channel coding to fully exploit channel characteristics and potential error correction

[6] This figure is inspired by a similar figure in [563].

capabilities available in the physical layer, which may lead to suboptimal performance. 2) Additional headers or metadata may be required to incorporate AL coding schemes, resulting in increased overhead and reduced overall efficiency. 3) AL coding may not have the same level of error correction capability as physical layer coding techniques, especially in environments with high levels of noise or interference. Lastly, since AL channel coding is application-dependent, it may be difficult for standardization, while physical layer coding must adhere to standardized protocols and modulation techniques to ensure interoperability with different network devices and technologies.

B. NON-BINARY CODES

The trivial alternatives for the well-known coding schemes are non-binary variants of these codes over Galois field GF(q). For binary codes, $q = 2$, whereas for non-binary codes q can be any prime number or a power of a prime number. The specific choice of q depends on the design requirements and characteristics of the code. In the previous sections, we have briefly discussed the non-binary variants of turbo codes (see Section V-I), LDPC codes (see Section VI-J), and polar codes (see Section VII-I). Hereafter, we provide a summary of the benefits and challenges of non-binary codes.

- Advantages: 1) Encoding is directly over the q-ary alphabet corresponding to the signal constellation. This saves the mapping and de-mapping processes by converting between binary bits and non-binary modulation symbols. Most importantly, non-binary codes do not have information loss as in the de-mapping of binary codes (unless with costly iteration between de-mapper and decoder or using multilevel coding). This makes non-binary codes appealing for high spectral efficiency coding over higher order constellations; 2) Non-binary codes such as non-binary turbo codes and non-binary LDPC codes suffer from less performance loss in short blocklength compared to their binary counterparts.
- Challenges: 1) The decoding complexity of non-binary codes increases with the alphabet size q. Consequently, the implementation complexity of non-binary codes is higher than that of binary codes; 2) Existing communication systems and standards are often designed with binary codes in mind. Introducing non-binary codes may raise compatibility and interoperability issues with legacy systems and standards. Transitioning from binary codes to non-binary codes would require significant infrastructure changes.

C. FOUNTAIN CODES

Fountain codes are a class of erasure codes designed for robust communication over unreliable channels, particularly in scenarios with frequent packet loss. They fall into the category of rateless codes capable of generating an infinite number of encoded symbols. In this coding scheme, a source block is partitioned into k equal length sub-blocks, called source symbols. The basic idea is that the transmitter encodes the source symbols and sends the encoded symbols in packets. A receiver uses the encoded symbols received in the packets to recover the original source block. Ideally, fountain codes should possess the following properties [203]:

- Ratelessness: The encoder of a fountain code should be able to generate as many encoded symbols as required for each receiver from the k source symbols of a source block.
- Erasure resilience: Given any subset of k encoded symbols, the fountain code decoder should be able to recover the original source block with high probability, regardless of which subset of the generated encoded symbols is received and independent of whether the subset was generated by one fountain encoder or generated by more than one encoder from the original block of source data.
- Linear-time complexity: The computation time for encoding and decoding should only scale linearly with respect to the size of a source block.

These properties illustrate the concept of a digital fountain, an analogy drawn from a fountain of water. Any receiver who aims to receive the source block holds a bucket under the fountain. It does not matter which particular drops fill the bucket. Only the amount of water, i.e., the number of encoded source symbols received that have been collected in the bucket is important to recover the original data. With a digital fountain, data packets may be obtained from one or more servers, and once enough packets are obtained (from whatever source), file transmission can end. The main figure of merit to assess the performance of a fountain code is the *reception overhead* ϵ, where $(1+\epsilon)k$ received encoded symbols are required to recover the k source symbols. It is also worth pointing out that the underlying characteristic of fountain codes can be exploited to realize secure wireless delivery. That is, secure communication is achieved if the legitimate receiver has received enough fountain-coded packets for decoding before the eavesdropper does [569]. Compared to existing physical layer security strategies, fountain-coding-based secure transmission strategies can significantly increase the transmission rate, which is bounded by Shannon capacity rather than secrecy capacity.

Moving forward, we introduce several popular classes of fountain codes.

1) LUBY TRANSFORM (LT) CODES

The LT code was the first practical realization of fountain codes [204]. It is defined from its degree distribution

$$\Omega(x) = \sum_{d} \Omega_d x^d, \tag{196}$$

where Ω_d is the probability to assign a degree d to an encoded symbol such that d is the number of source symbols participating in the encoding of this symbol. LT codes are a class of low-density generator-matrix (LDGM) codes, enabling the use of BP decoding. It was shown that by using

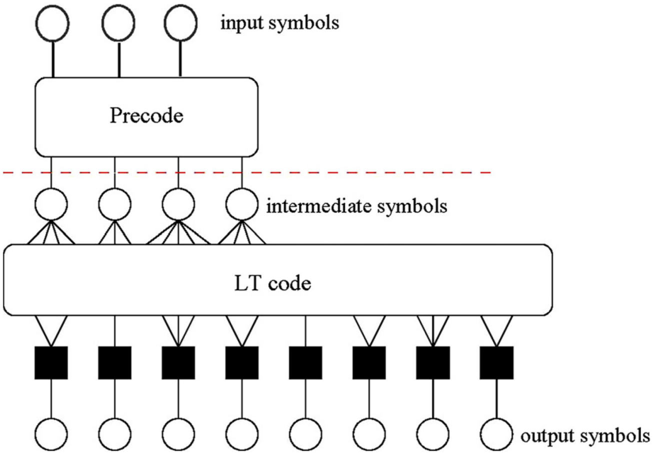


FIGURE 85. Tanner graph representation of Raptor codes [203].

the ideal soliton distribution (197), only k encoded symbols are sufficient to recover the k source symbols by using the ideal soliton distribution.

$$\Omega_k(x) = \frac{x}{k} + \sum_{d=2}^{k} \frac{x^d}{d(d-1)}. \tag{197}$$

However, this distribution works poorly in practice, as it is likely that at some decoding step there will be no available degree-one check node in the graph, leading to decoding failure. To overcome this issue, a robust soliton distribution was then introduced to allow the BP decoder to work well in practice, resulting in an overhead of $O(\log^2(k/\delta)\cdot\sqrt{k})$ and an average degree of $O(\log(k/\delta))$, where δ is the probability of decoding failure. The average number of symbol operations per encoded symbol generated and to decode the k source symbols are is $O(\log(k/\delta))$ and $k\cdot O(\log(k/\delta))$, respectively. It can be seen that on average every encoded symbol needs $O(\log k)$ operations and that the decoding algorithm needs $O(k\log k)$ symbol operations, which are not linear encoding and decoding time.

2) RAPTOR CODES

Raptor codes are a class of fountain codes that achieve linear-time encoding and decoding [203]. They can be regarded as an improvement of their LT relatives. The key idea of Raptor codes $(k, \mathcal{C}, \Omega(x))$ is to concatenate an inner LT code with degree distribution $\Omega(x)$ as in (196) with a high rate outer code or a precode $\mathcal{C}(n, k)$. Raptor codes can be represented using the Tanner graph as shown in Fig. 85. The outer code can be any erasure correction code, e.g., LDPC codes, which can recover up to a fraction δ of erasures among the intermediate symbols, that is, the outer codeword symbols. Meanwhile, the LT inner code is responsible for recovering the remaining $(1-\delta)$-fraction of the intermediate symbols. It has been proved that for any $\epsilon > 0$, there exists a class of universal Raptor codes such that any subset of symbols of size $k(1+\epsilon)$ is sufficient to recover the original k symbols with high probability. Moreover, the average number of symbol operations per generated encoded symbol and to decode the source block are $O(\log(1/\epsilon))$ and $O(k\cdot\log(1/\epsilon))$, respectively.

To design Raptor codes with overhead $k(1+\epsilon)$ for achieving good asymptotic performance, the degree distribution $\Omega(x)$ is designed in such a way that

$$\sup\{x \in [0, 1)|1 - x - e^{-(1+\epsilon)\Omega'(x)} > 0\}, \tag{198}$$

is maximized. To take into account the finite blocklength effect, the degree distribution satisfies the following

$$1 - x - e^{-(1+\epsilon)\Omega'(x)} \geqslant \gamma\sqrt{\frac{1-x}{k}}, \tag{199}$$

for $x \in [0, 1-\delta]$ such that the decoding process will continue with high probability until it has recovered all but a δ-fraction of the intermediate symbols, and γ is a positive design parameter.

In addition to the erasure channel, the design of raptor codes has also been carried out for the BSC [570], the BI-AWGN channels [571], [572], and the fading channels [573], where density evolution and EXIT charts (see Section VI-B) are the main design tools. The benefits provided by Raptor codes were also exploited in a number of communication scenarios, such as wireless relay channels [574], multiple access channels [575], and Gaussian broadcast channels [576]. Raptor codes have been adopted in a number of different standards, such as the 3GPP Multimedia Broadcast Multicast Service and the Internet Engineering Task Force (IETF) Request for Comments (RFC) 6330, for file delivery and streaming [203, Ch. 3.1].

3) SPINAL CODES

Spinal codes are a family of rateless codes proposed in [577] in 2011. They have been proved to achieve the capacity of the BSC and the AWGN channel [578]. The rateless feature allows Spinal codes to be naturally adapted to time-varying channel conditions. Compared to other rateless codes, e.g., Raptor codes, Spinal codes have demonstrated advantages in error performance under different channel conditions and message sizes [579]. Different from conventional algebraic coding, Spinal codes employ hash functions and random number generator (RNG) functions.

First, an n-bit message $\boldsymbol{m}$ is divided into n/k k-bit segments $\boldsymbol{m} = [\boldsymbol{m}_1, \ldots, \boldsymbol{m}_{n/k}]$. Next, the encoder applies a hash function $h(\cdot)$ to sequentially map the message segment to a v-bit hash state as

$$\boldsymbol{s}_i = h(\boldsymbol{s}_{i-1}, \boldsymbol{m}_i),\ i = 1, \ldots, n/k, \tag{200}$$

where $\boldsymbol{s}_0 = \mathbf{0}$ is the initial hash state known by both the encoder and decoder. Then, the v-bit has state is input to the RNG function as a seed to generate a pseudo-random sequence of length c

$$\text{RNG} : \boldsymbol{s}_i \rightarrow \boldsymbol{x}_{i,j} \in \{0, 1\}^c,\ j = 1, 2, 3, \ldots. \tag{201}$$

The Spinal encoder then maps the c-bit sequence to a channel input set Ψ to fit the channel characteristics:

$$f : \boldsymbol{x}_{i,j} \rightarrow \Psi, \tag{202}$$

where f is a constellation mapping function.

Due to the introduction of a hash function, the decoding of Spinal codes usually has a higher complexity than other classes of rateless codes. By leveraging the tree structure of Spinal codes, a tree pruning decoding algorithm called bubble decoding was proposed for Spinal codes [579]. It has a decoding complexity of $O(nB2^k(k + \log_2 B + v))$, where B denotes the pruning width. To reduce the decoding complexity, a forward stack decoding algorithm was proposed in [580], which divides the decoding tree into several layers and searches the decoding paths in each single layer without going back to the previous layer. However, the complexity reduction is limited. Yet, the development of low-complexity and high-performance decoding algorithms for Spinal codes remains to be a challenging problem.

D. LATTICE CODES

Lattice codes, constructed from lattices, serve as the Euclidean counterpart of binary linear codes [581]. They possess numerous favorable properties and elegant mathematical structures [582]. The motivation for employing lattices in channel coding is due to the fact that many elegant properties of the lattices can be carried over to solve practical engineering problems. To preserve lattice symmetry and save complexity, lattice decoding which finds the closest lattice point while ignoring the decision boundary of the code [583], is often used rather than the maximum-likelihood decoding. Remarkably, it was first proved in [584] that there exists a sequence of lattice codes that can achieve the AWGN channel capacity under lattice decoding.

In addition to information-theoretic analysis, another line of research is dedicated to the construction of practical lattice codes with capacity-approaching performance. In general, there are two main approaches. The first involves constructing lattice codes directly in the Euclidean space, e.g., low-density lattice codes (LDLC) [585] and convolutional lattice codes (CLC) [586]. Another approach is to adapt modern capacity-approaching channel codes to construct lattices, i.e., construct lattice codes from LDPC codes [587], [588], IRA codes [589], and polar codes [590]. Their construction methods are based on [582]: 1) Construction A; constructing lattices based on a linear code; 2) Construction D: constructing lattices based on the generator matrices of a series of nested linear codes; and 3) Construction D': constructing lattices based on the parity check matrices of a series of nested linear codes. These methods allow one to construct lattice codes not only with good error performance inherited from capacity-achieving linear codes, but also having relatively lower construction complexity compared with LDLCs and CLCs. The concepts of lattices have also been used to design constellations with large shaping gains and achieving the full diversity of the Rayleigh fading channel [591], [592] and to design full-rate full-diversity space-time coding for MIMO channels [593].

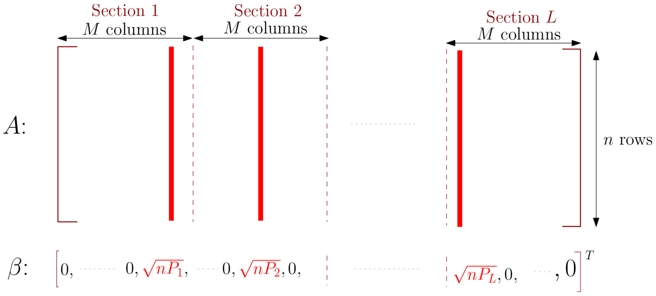

Section 1
M columns
Section 2
M columns
Section L
M columns
A:
n rows
β:
$[0, \cdots 0, \sqrt{nP_1}, \cdots 0, \sqrt{nP_2}, 0, \cdots \sqrt{nP_L}, 0, \cdots, 0]^T$

FIGURE 86. A Gaussian sparse regression codebook of block length *n* [605].

In addition to point-to-point channels, lattice codes have been shown to outperform Gaussian random codes in multiuser communications and interference management. It is worth noting that one of the key properties of lattice codes is that a linear combination of multiple lattice codes is still a lattice code. Leveraging this property, [594] proposed compute-and-forward (C&F) relaying strategy based on nested lattice codes by harnessing structural interference to achieve significantly higher rates than conventional amplify-and-forward and decode-and-forward relaying strategies. Building on the C&F framework, lattice network coding has been proposed and studied in [595], [596], [597]. Multiple access based on lattice partition was proposed and investigated in [598], [599], [600], [601], which enables low-complexity treating interference as noise decoding for interference management as opposed to complicated successive interference cancellation decoding. In all of the aforementioned works, higher-dimensional lattices are required to attain the optimal performance, whose implementation complexity scales with dimension.

E. SPARSE REGRESSION CODES (SPARCS)

SPARCs or sparse superposition codes were first introduced in [602], [603], which have been proved to achieve capacity over the AWGN channel under maximum-likelihood decoding and adaptive successive decoding, respectively. SPARCs step back from the coding/modulation divide and instead use a structured codebook to construct low-complexity capacity-achieving schemes tailored to the AWGN channel. Later, it has been proved that SPARCs also achieve capacity under approximate message passing (AMP) decoding [604].

A SPARC is defined in terms of a design matrix $\boldsymbol{A}$ of dimension $n \times ML$, whose entries are chosen i.i.d. over $\mathcal{N}(0, 1/n)$, where n is the blocklength, M and L are integers associated with n and the code rate R. An example of a Gaussian sparse regression codebook of blocklength n is shown in Fig. 86. Matrix $\boldsymbol{A}$ consists of L sections with M columns each.

The SPARC codeword is generated by

$$\boldsymbol{x} = \boldsymbol{A}\boldsymbol{\beta}, \tag{203}$$

where $\boldsymbol{\beta} = [\beta_1, \dots, \beta_{ML}]$ is the message vector. It satisfies the following property: there is *exactly one* $\beta_j \neq 0$ for $j = 1, \dots, M$, one $\beta_j \neq 0$ for $j = M+1, \dots, 2M$, and so forth. The non-zero value of β in section $l \in \{1, \dots, L\}$ is set to $\sqrt{nP_l}$, where $P_l > 0$ and satisfies $\sum_l^L P_l = P$ and P is the average power per input symbol. Since each of the L sections contains M columns, the codebook size is M^L. To obtain a rate R code, it is required that

$$M^L = 2^{nR} \Rightarrow R = \frac{L \log M}{n}. \tag{204}$$

There are several choices for the pair (M, L) that satisfy (204). In most cases, $M = L^a$ for some constant $a > 0$ [605]. In this case, the code rate becomes $R = \frac{aL \log L}{n}$. Note that SPARCs are non-linear codes. The choice of P_l, known as power allocation, has critical impacts on performance. For example, SPARCs with exponentially decaying power allocation $P_l \propto 2^{-2C/L}$ have been proved to be capacity achieving under AMP decoding, where C denotes channel capacity [604]. Further optimization of power allocation is required to achieve good finite blocklength error performance.

Several works have improved the original SPARCs. First, notice that matrix $\boldsymbol{A}$ is an i.i.d. Gaussian matrix that may not be suitable for practical implementation. Therefore, the SPARC can be defined via a Bernoulli design matrix with entries that are chosen uniformly at random from the set $\{-1, 1\}$ [606], [607]. The capacity-achieving property is still preserved under the optimal ML decoding. In [608], the SPARCs were extended to *modulated SPARCs*, where the information was encoded in both the locations and values of the non-zero entries of $\boldsymbol{\beta}$. To be specific, each non-zero entry of $\boldsymbol{\beta}$ takes values from a K-ary constellation. In this case, the code rate of (204) becomes

$$R = \frac{L \log(KM)}{n}. \tag{205}$$

Adding modulations introduces an extra degree of freedom in the design of SPARCs, which can be used to reduce the decoding complexity without sacrificing finite-length error performance [608]. In addition to power allocation, another way of achieving capacity is by applying spatial coupling [604]. Other extensions, such as concatenated SPARCs with different coding schemes, have been shown to be promising in multiuser channels [609].

X. OTHER RELEVANT TOPICS

Channel coding is one of the cascaded stages of the transmission process. As the main contribution of coding is to the reliability of the wireless link, along with influences on other KPIs, we need to consider other stages in a communication system.

A. SIGNALING AND SHAPING

Modulation is the process of converting a digital signal into an analog signal that can be transmitted over a communication channel. The choice of modulation scheme can have a significant impact on the reliability and data rate of a communication system. For example, higher order modulation schemes, such as 16-QAM and 64-QAM, can achieve higher data rates compared to low order modulation schemes, such as BPSK and QPSK, but they are more susceptible to noise and interference. QAM is widely used in cellular systems, e.g., 2G to 5G systems, because it is relatively simple to modulate and demodulate. However, its constellation points are arranged in equally spaced positions, and the distribution is far from Gaussian. Particularly, in the high SNR regime, QAM has a gap of 1.53 dB to the unrestricted Shannon limit (i.e., not restricted to any signal constellation), which is known as the shaping loss. Amplitude phase shift keying (APSK) is more robust to the non-linearity of the power amplifier (PA) and phase noise than QAM. This makes it a good choice for broadcasting networks and satellite communications. APSK could also be used in 6G systems.

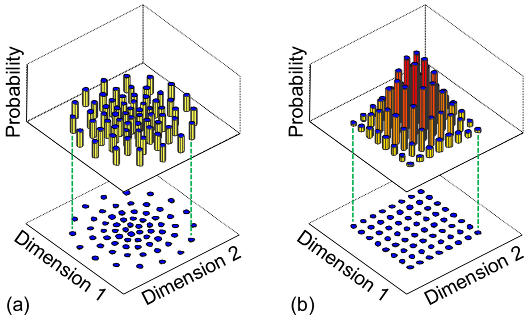


FIGURE 87. (a) Geometric and (b) probabilistic constellation shaping [610].

To close the gap to the unrestricted Shannon limit and to increase the spectral efficiency, signal shaping can be applied. There are two main classes of signal shaping, namely geometric shaping [611] and probabilistic shaping [612]. Illustrations of both shaping methods are shown in Fig 87. In geometric shaping, the constellation points are arranged in a way to mimic the capacity-achieving signal distribution. Constellations built from lattices (see Section IX-D) are an example of geometric shaping. Probabilistic shaping, on the other hand, assigns different probabilities to different constellation points. Compared to geometric shaping, probabilistic shaping builds up on off-the-shelf constellations, hence incurring no additional complexity in system design and implementation.

The best way to improve both the reliability and spectral efficiency of a coded modulation scheme is to jointly design modulation, shaping, and channel coding schemes. This is because the performance of each component depends on the other components. However, jointly designing the modulation, shaping, and channel coding schemes is not a trivial job, in particular for a cellular system, where we need to adapt the coding and modulation parameters with the channel condition.

B. THE CHOICE OF WAVEFORM

Waveform design can play a significant role in improving the KPIs of cellular networks. Some of the ways that waveform design can improve KPIs include: 1) Increased data rates: Waveform design can be used to increase the data rates of cellular networks by increasing the spectral efficiency of the waveforms. This can be done by using techniques such as orthogonal frequency-division multiplexing (OFDM) and its variants. 2) Sensing resolutions: Waveform design can be used to improve the sensing performance of cellular networks by exploiting its time-frequency localization. This can be done by using techniques such as shorter waveforms, lower overhead, and the associated simplified detection algorithms. Improved coverage: Waveform design can be used to improve the coverage of cellular networks by making the waveform more robust to interference and fading. This can be done by using techniques such as multiple-input multiple-output (MIMO) and beamforming. 3) Increased energy efficiency: Waveform design can be used to increase the energy efficiency of cellular networks by reducing the power consumption of generating the waveforms. This can be done by using techniques such as adaptive power control and waveform selection.

Some of the multi-carrier waveform candidates for downlink in 6G cellular networks include: 1) Orthogonal frequency division multiplexing (OFDM) [613]: OFDM is a waveform that has been used in cellular networks since 4G. It is also a good candidate for 6G networks as it is efficient in terms of spectral efficiency, power consumption, and easy to integrate with MIMO. However, it has high out-of-band emission (OOBE) and peak-to-average power ratio (PAPR), and it performs poorly at high mobility. 2) OFDM variants such as filter bank multi-carrier (FBMC) modulation [614]: FBMC achieves good frequency-domain localization by increasing the pulse duration in the time domain and using carefully designed pulse shaping filters. Among the variants of FBMC, offset quadrature amplitude modulation (OQAM–FBMC), is preferred due to handling interference while allowing dense symbol placement in the time–frequency plane. Compared to OFDM, it has lower OOBE and PAPR at the cost of higher complexity and larger bit error rate (BER). Other OFDM variants also suffer from some disadvantages, such as inter-symbol interference (ISI) due to lack of cyclic prefix (CP), challenging MIMO integration, higher complexity, and therefore larger latency. 3) Orthogonal time frequency space (OTFS) modulation [615] and its improved variant orthogonal delay-Doppler modulation (ODDM) [616]: OTFS/ODDM or a general delay-Doppler multicarrier modulation (DDMC) [617] are relatively new waveforms that improve robustness in environments with high-frequency dispersion by processing the signal in the delay-Doppler domain where the signals are sparse. Nevertheless, this advantage comes with a relatively high complexity cost. Researchers are still working to improve the discussed waveforms and investigate further as candidates for 6G cellular networks. A combination of different waveforms will likely be used in 6G networks to meet the diverse requirements of different applications.

C. INTEGRATION WITH NON-TERRESTRIAL NETWORKS AND LASER LINKS

In the pursuit of ubiquitous connectivity for 6G networks, integration of non-terrestrial networks (NTNs) [618] such as unmanned aerial vehicles (UAVs), satellites, and other high altitude platforms (HAPs) such as balloons with terrestrial networks, along with the utilization of alternative communication technologies such as free space optical (FSO) links [619] (a.k.a. laser links) for backhaul/fronthaul becomes imperative. Note that FSO is also an option for point-to-point communications in terrestrial networks; however, due to weather dependency, limited range (as line-of-sight is required), challenges for meeting alignment requirements, etc., they have not been employed. FSO links have already been used for inter-satellite communications and could be an option for linking HAPs to gateways of terrestrial networks. The integration of terrestrial and non-terrestrial networks addresses the limitations of the frequency spectrum available for terrestrial communications, as well as the high data rates envisioned for 6G. The inherent challenges of spectrum congestion and limited bandwidth on earth necessitate the exploration of non-terrestrial alternatives to expand coverage and increase capacity. However, these non-terrestrial technologies may require different design considerations for channel coding and modulation due to factors such as atmospheric conditions, propagation delays, and varying link characteristics due to side effects (namely, absorption, scattering, and turbulence) of the atmospheric channel, which results in fluctuations of the received signal intensity. To model the received intensity distribution according to the turbulence levels, probability density functions such as lognormal, negative exponential, and, in particular, gamma-gamma (G-G) are used. Adaptation of current error correction codes and modulation techniques must account for these factors to ensure reliable and efficient communication over non-terrestrial links. Potential candidates for such adaptations include advanced coding schemes like LDPC codes, polar codes, and turbo codes, which offer robust performance in challenging environments and can be tailored to meet the requirements of non-terrestrial communication channels for both radio frequency (RF) and optical communications. Limited investigation [620], [621] has been carried out on channel coding for FSO where LDPC codes [622], rateless codes [623], and Reed-Solomon codes [624] have been used. Nevertheless, due to the characteristics of the atmospheric channel, improving channel modeling, channel estimation approaches, equalization methods, and waveform design seem to contribute more than channel coding schemes in link performance. Overall, the integration of non-terrestrial networks and innovative communication technologies presents an opportunity to address the scalability and performance challenges of 5G networks while paving the way for seamless connectivity in diverse environments.

XI. SUMMARY AND CONCLUSION

We reviewed the specifications of the mobile communication standards from 1G to 5G, in particular the channel coding schemes employed in every generation, the envisioned 6G requirements, and the potential contributions of channel coding to meet these requirements. We then reviewed Turbo, LDPC, and polar coding schemes, their variants, and recent advances. The comparisons were made in terms of error correction performance and the performance of hardware architectures designed for different decoding algorithms. We also considered other coding schemes, such as fountain codes and lattice codes, and other considerations in the physical layer, such as modulation and waveform.

In our opinion, turbo codes, polar codes, and LDPC codes will remain strong contenders for the next generation of mobile communication standards. The reasons are twofold: 1) These three coding schemes are well established and investigated. Turbo and LDPC codes for code lengths required for data channels are capable of outperforming other codes at reasonable complexity. Similarly, polar codes outperform other codes for the code lengths required for control channels. Based on previous trends, it appears that we need more than the span of one generation of mobile communications, more than a decade, to develop a well-investigated and mature coding scheme for practical applications. 2) Adapting a new coding scheme can be challenging due to the need for standardization and interoperability. The introduction of new coding techniques can require significant modifications to the existing standard, which is costly. Furthermore, the efficient implementation of new codes on mobile devices may present practical challenges. Optimizing the software and hardware for efficient code implementation requires careful consideration of the device's capabilities and constraints.

However, recent advances and promising directions in existing coding schemes, as discussed in Sections V–VII, could lead to some modifications of existing standards. In particular, the spatially coupled and non-binary variants of turbo, LDPC, and polar codes could play a role in some applications.

Furthermore, as discussed in Section I and observed in Section III in particular Table 8, the contribution of coding schemes in improving KPIs is more significant in enhancing reliability, throughput, and coverage. However, these KPIs and others that channel coding has less or no impact on them, can be improved by other components of transmitter/receiver, network architecture, and network management. On the other hand, the decoding process could be the bottleneck of a receiver and further improvement of the decoding algorithm for lower complexity towards terahertz communications is necessary and it is as important as improving the reliability.

REFERENCES

[1] "Introducing 3GPP." Accessed: Sep. 30, 2023. [Online]. Available: https://www.3gpp.org/about-us/introducing-3gpp

[2] "3GPP portal—Specifications." Accessed: Sep. 30, 2023. [Online]. Available: https://portal.3gpp.org/#/55936-specifications

[3] "Framework and overall objectives of the future development of IMT-2000 and systems beyond IMT-2000," ITU, Geneva, Switzerland, ITU-Recommendation M.1645, 2003.

[4] "Guidelines for evaluation of radio interface technologies for IMT-advanced," ITU, Geneva, Switzerland, Rep. 638, 2009.

[5] "IMT vision—Framework and overall objectives of the future development of IMT for 2020 and beyond," ITU, Geneva, Switzerland, ITU-Recommendation 2083, 2015.

[6] "The next wave of 5G—3GPP release 19." Accessed: Jan. 26, 2024. [Online]. Available: https://www.ericsson.com/en/blog/2023/12/3gpp-release-19

[7] M. Giordani, M. Polese, M. Mezzavilla, S. Rangan, and M. Zorzi, "Toward 6G networks: Use cases and technologies," *IEEE Commun. Mag.*, vol. 58, no. 3, pp. 55–61, May 2020.

[8] D. Minoli, *Telecommunications Technology Handbook*. London, U.K.: Artech House, 2003.

[9] *GSM/EDGE Channel Coding*, 3GPP Standard TS 45.003, 2005.

[10] H. Holma and A. Toskala, *LTE for UMTS: Evolution to LTE-Advanced*. Hoboken, NJ, USA: Wiley, 2011.

[11] J. Xu and Y. Yuan, *Channel Coding in 5G New Radio*. Boca Raton, FL, USA: CRC Press, 2022.

[12] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1," in *Proc. IEEE Int. Conf. Commun. (ICC)*, vol. 2, 1993, pp. 1064–1070.

[13] *Multiplexing and Channel Coding (TDD), v3.10.0 (Release 1999)*, 3GPP Standard TS 25.222, 2002.

[14] *Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and Channel Coding, v8.8.0 (Release 8)*, 3GPP Standard TS 36.212, 2009.

[15] *NR; Multiplexing and Channel Coding, v15.13.0 (Release 15)*, 3GPP Standard TS 38.212, 2022.

[16] 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 Veh. Technol. Mag.*, vol. 13, no. 4, pp. 60–69, Dec. 2018.

[17] M. Latva-Aho et al. "Key drivers and research challenges for 6G ubiquitous wireless intelligence." 2019. [Online]. Available: https://www.6gflagship.com/key-drivers-and-research-challenges-for-6g-ubiquitous-wireless-intelligence/

[18] G. D'Aria et al. "Expanded 6G vision, use cases and societal values—Including aspects of sustainability, security and spectrum." 2021. [Online]. Available: https://hexa-x.eu/d1-2-expanded-6g-vision-use-cases-and-societal-values-including-aspects-of-sustainability-security-and-spectrum/

[19] *White Paper: 6G Use Cases and Analysis*, NGMN, Düsseldorf, Germany, Feb. 2022.

[20] *White Paper: On the Road to 6G: Drivers, Challenges and enabling Technologies*, Fraunhofer, Munich, Germany, Nov. 2021.

[21] *White Paper: The Next Hyper-Connected Experience for All*, Samsung, Suwon, South Korea, Jul. 2020.

[22] *White Paper: Orange's Vision for 6G*, Orange, Paris, France, Mar. 2022.

[23] Y. Polyanskiy, H. V. Poor, and S. Verdú, "Channel coding rate in the finite blocklength regime," *IEEE Trans. Inf. Theory*, vol. 56, no. 5, pp. 2307–2359, May 2010.

[24] T. Erseghe, "Coding in the finite-blocklength regime: Bounds based on laplace integrals and their asymptotic approximations," *IEEE Trans. Inf. Theory*, vol. 62, no. 12, pp. 6854–6883, Dec. 2016.

[25] M. C. Coakun et al., "Efficient error-correcting codes in the short blocklength regime," *Phys. Commun.*, vol. 34, pp. 66–79, Jun. 2019.

[26] E. M. Jazi and J. N. Laneman, "Coded modulation for Gaussian channels: Dispersion- and entropy-limited regimes," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, 2015, pp. 528–533.

[27] M. Qiu, Y.-C. Huang, and J. Yuan, "Downlink transmission with heterogeneous URLLC services: Discrete signaling with single-user decoding," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 7, pp. 2261–2277, Jul. 2023.

[28] E. Guizzo, "Closing in on the perfect code [turbo codes]," *IEEE Spectr.*, vol. 41, no. 3, pp. 36–42, Mar. 2004.

[29] *Universal Mobile Telecommunications System (UMTS); Multiplexing and Channel Coding (FDD)*, 3GPP Standard TS 25.212, Apr. 2022.

[30] *Universal Mobile Telecommunications System (UMTS); Multiplexing and Channel Coding (TDD)*, 3GPP Standard TS 25.222, Apr. 2022.

[31] *LTE;. Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and Channel Coding, V17.1.0*, 3GPP Standard TS 36.212, Apr. 2022.

[32] *IEEE Standard for Local and Metropolitan Area Networks, Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems*, IEEE Standard 802.16-2004/Cor 1-2005, Feb. 2006.

[33] *Digital Video Broadcasting (DVB): Second Generation DVB Interactive Satellite System (DVB-RCS2): Part 2: Lower Layers for Satellite Standard*, ETSI, Standard EN 301 545-2, Jul. 2020.

[34] D. Declerq, M. Fossorier, and E. Biglieri, *Channel Coding: Theory, Algorithms, and Applications*. Oxford, U.K.: Academic, 2014.

[35] *Enhanced Turbo Codes for NR: Implementation Details*, 3GPP, Sophia Antipolis, France, Aug. 2016.

[36] R. Pyndiah, "Near-optimum decoding of product codes: Block turbo codes," *IEEE Trans. Commun.*, vol. 46, no. 8, pp. 1003–1010, Aug. 1998.

[37] H. Mukhtar, A. Al-Dweik, and A. Shami, "Turbo product codes: Applications, challenges, and future directions," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 4, pp. 3052–3069, 4th Quart., 2016.

[38] C. Douillard et al., "Iterative correction of intersymbol interference: Turbo-equalization," *Eur. Trans. Telecommun.*, vol. 6, no. 5, pp. 507–511, 1995.

[39] X. Wang and H. Poor, "Iterative (turbo) soft interference cancellation and decoding for coded CDMA," *IEEE Trans. Commun.*, vol. 47, no. 7, pp. 1046–1061, Jul. 1999.

[40] P. Elias, "Error-free coding," *IRE Trans. Inf. Theory*, vol. 4, no. 4, pp. 29–37, 1954.

[41] J. Rolf and K. S. Zigangirov, *Fundamentals of Convolutional Coding*, 2nd ed. Hoboken, NJ, USA: Wiley, 2015.

[42] D. Divsalar and F. Pollara, "Turbo codes for PCS applications," in *Proc. IEEE Int. Conf. Commun. (ICC)*, vol. 1, 1995, pp. 54–59.

[43] H. Ma and J. Wolf, "On tail biting convolutional codes," *IEEE Trans. Commun.*, vol. 34, no. 2, pp. 104–111, Feb. 1986.

[44] C. Weiss, C. Bettstetter, S. Riedel, and D. Costello, "Turbo decoding with tail-biting trellises," in *Proc. URSI Int. Symp. Signals Syst. Electron.*, 1998, pp. 343–348.

[45] C. Weiss, C. Bettstetter, and S. Riedel, "Code construction and decoding of parallel concatenated tail-biting codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 1, pp. 366–386, Jan. 2001.

[46] C. Berrou, *Codes and Turbo Codes*. Paris, France: Springer-Verlag, 2010.

[47] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inf. Theory*, vol. IT-13, no. 2, pp. 260–269, Apr. 1967.

[48] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inf. Theory*, vol. IT-20, no. 2, pp. 284–287, Mar. 1974.

[49] G. Forney, "The viterbi algorithm," *Proc. IEEE*, vol. 61, no. 3, pp. 268–278, May 1973.

[50] J. Hagenauer and P. Hoeher, "A Viterbi algorithm with soft-decision outputs and its applications," in *Proc. IEEE Globecom*, vol. 3, 1989, pp. 1680–1686.

[51] R. Y. Shao, S. Lin, and M. P. C. Fossorier, "Two decoding algorithms for tailbiting codes," *IEEE Trans. Commun.*, vol. 51, no. 10, pp. 1658–1665, Oct. 2003.

[52] N. Seshadri and C.-E. Sundberg, "List Viterbi decoding algorithms with applications," *IEEE Trans. Commun.*, vol. 42, no. 234, pp. 313–323, Feb./Apr. 1994.

[53] H. Yang, E. Liang, M. Pan, and R. D. Wesel, "CRC-aided list decoding of convolutional codes in the short blocklength regime," *IEEE Trans. Inf. Theory*, vol. 68, no. 6, pp. 3744–3766, Jul. 2022.

[54] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," in *Proc. IEEE Int. Conf. Commun. (ICC)*, vol. 2, 1995, pp. 1009–1013.

[55] C. Douillard and C. Berrou, "Turbo codes with rate-M/(M+1) constituent convolutional codes," *IEEE Trans. Commun.*, vol. 53, no. 10, pp. 1630–1638, Oct. 2005.

[56] A. Heim and U. Sorger, "Turbo decoding: Why stopping-criteria do work," in *Proc. Int. Symp. Turbo Codes Rel. Topics (ISTC)*, 2008, pp. 255–259.

[57] F. Kschischang, B. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.

[58] S. Moloudi, M. Lentmaier, and A. Graell i Amat, "Spatially coupled turbo-like codes," *IEEE Trans. Inf. Theory*, vol. 63, no. 10, pp. 6199–6215, May 2019.

[59] D. V. Truhachev, M. Lentmaier, and K. S. Zigangirov, "Some results concerning design and decoding of turbo codes," *Probl. Inf. Transm.*, vol. 37, no. 3, pp. 190–205, 2001.

[60] M. Breiling, "A logarithmic upper bound on the minimum distance of turbo codes," *IEEE Trans. Inf. Theory*, vol. 50, no. 8, pp. 1692–1710, Aug. 2004.

[61] M. Cedervall and R. Johannesson, "A fast algorithm for computing distance spectrum of convolutional codes," *IEEE Trans. Inf. Theory*, vol. 35, no. 6, pp. 1146–1159, Nov. 1989.

[62] I. E. Bocharova, M. Handlery, R. Johannesson, and B. Kudryashov, "A BEAST for prowling in trees," *IEEE Trans. Inf. Theory*, vol. 50, no. 6, pp. 1295–1302, Jun. 2004.

[63] R. Garello, P. Pierleoni, and S. Benedetto, "Computing the free distance of turbo codes and serially concatenated codes with interleavers: Algorithms and applications," *IEEE J. Sel. Areas Commun.*, vol. 19, no. 5, pp. 800–812, May 2001.

[64] E. Rosnes and Y. Ytrehus, "Improved algorithms for the determination of turbo-code weight distributions," *IEEE Trans. Commun.*, vol. 53, no. 1, pp. 20–26, Jan. 2005.

[65] R. Garello and A. Vila-Casado, "The all-zero iterative decoding algorithm for turbo code minimum distance computation," in *Proc. IEEE Int. Conf. Commun. (ICC)*, vol. 1, 2004, pp. 361–364.

[66] S. Crozier, P. Guinand, and A. Hunt, "Estimating the minimum distance of large-block turbo codes using iterative multiple-impulse methods," in *Proc. Int. Symp. Turbo Codes Rel. Topics (ISTC)*, 2006, pp. 1–6.

[67] B. Vucetic and J. Yuan, *Turbo Codes: Principles and Applications*. New York, NY, USA: Springer, 2000.

[68] T. Richardson and R. Urbanke, *Modern Coding Theory*. New York, NY, USA: Cambridge Univ. Press, 2008.

[69] S. Benedetto and G. Montorsi, "Unveiling turbo codes: Some results on parallel concatenated coding schemes," *IEEE Trans. Inf. Theory*, vol. 42, no. 2, pp. 409–428, Mar. 1996.

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

[71] B. Kurkoski, P. Siegel, and J. Wolf, "Exact probability of erasure and a decoding algorithm for convolutional codes on the binary erasure channel," in *Proc. IEEE Globecom*, vol. 3, 2003, pp. 1741–1745.

[72] G. M. Kraidy and V. Savin, "Capacity-approaching irregular turbo codes for the binary erasure channel," *IEEE Trans. Commun.*, vol. 58, no. 9, pp. 2516–2524, Sep. 2010.

[73] I. Andriyanova, "Finite-length scaling of turbo-like code ensembles on the binary erasure channel," *IEEE J. Sel. Areas Commun.*, vol. 27, no. 6, pp. 918–927, Aug. 2009.

[74] C. Measson, R. Urbanke, A. Montanari, and T. Richardson, "Maximum *a posteriori* decoding and turbo codes for general memoryless channels," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2005, pp. 1241–1245.

[75] M. U. Farooq, A. Graell i Amat, and M. Lentmaier, "Threshold computation for spatially coupled turbo-like codes on the AWGN channel," *Entropy*, vol. 23, no. 2, p. 240, 2021.

[76] S. ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Trans. Commun.*, vol. 49, no. 10, pp. 1727–1737, Oct. 2001.

[77] M. El-Hajjar and L. Hanzo, "EXIT charts for system design and analysis," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 127–153, 1st Quart., 2014.

[78] S. ten Brink, G. Kramer, and A. Ashikhmin, "Design of low-density parity-check codes for modulation and detection," *IEEE Trans. Commun.*, vol. 52, no. 4, pp. 670–678, Apr. 2004.

[79] J. Hagenauer, "The EXIT chart—Introduction to extrinsic information transfer in iterative processing," in *Proc. 12th Eur. Signal Process. Conf (EUSIPCO)*, Sep. 2004, pp. 1541–1548.

[80] S. Benedetto and G. Montorsi, "Design of parallel concatenated convolutional codes," *IEEE Trans. Commun.*, vol. 44, no. 5, pp. 591–600, May 1996.

[81] C. Measson, "Conservation laws for coding," Ph.D. dissertation, Dept. Comput. Sci., École polytechnique fédérale de Lausanne, Lausanne, Switzerland, 2006.

[82] B. Vucetic, Y. Li, L. C. Perez, and F. Jiang, "Recent advances in turbo code design and theory," *Proc. IEEE*, vol. 95, no. 6, pp. 1323–1344, Jun. 2007.

[83] P. Popovski, L. Kocarev, and A. Risteski, "Design of flexible-length S-random interleaver for turbo codes," *IEEE Commun. Lett.*, vol. 8, no. 7, pp. 461–463, Jul. 2004.

[84] S. Dolinar and D. Divsalar, 'Weight distributions for turbo codes using random and nonrandom permutations," JPL, Pasadena, CA, USA, Aug. 1995.

[85] S. Crozier, "New high-spread high-distance interleavers for turbo codes," in *Proc. 20th Biennial Symp. Commun.*, 2000, pp. 3–7.

[86] E. Boutillon and D. Gnaedig, "Maximum spread of D-dimensional multiple turbo codes," *IEEE Trans. Commun.*, vol. 53, no. 8, pp. 1237–1242, Aug. 2005.

[87] K. Gracie and S. Crozier, "Convergence performance and EXIT analysis of 4-state partially-systematic turbo codes," in *Proc. Int. Symp. Turbo Codes Rel. Topics (ISTC)*, 2008, pp. 414–419.

[88] S. Crozier and K. Gracie, "Rate-compatible turbo codes designed with puncture-constrained DRP interleavers," in *Proc. IEEE Globecom*, 2011, pp. 1–5.

[89] R. Garzón-Bohórquez, C. A. Nour, and C. Douillard, "Protograph-based interleavers for punctured turbo codes," *IEEE Trans. Commun.*, vol. 66, no. 5, pp. 1833–1844, May 2018.

[90] R. Garzón-Bohórquez, R. Klaimi, C. A. Nour, and C. Douillard, "Mitigating correlation problems in turbo decoders," in *Proc. Int. Symp. Turbo Codes Iterative Inf. Process (ISTC)*, Dec. 2018, pp. 1–5.

[91] R. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inf. Theory*, vol. IT-27, no. 5, pp. 533–547, Sep. 1981.

[92] J. Thorpe, "Low-density parity-check (LDPC) codes constructed from protographs," California Inst. Technol., Pasadena, CA, USA, Rep. 42-154, 2003.

[93] J. Sun and O. Takeshita, "Interleavers for turbo codes using permutation polynomials over integer rings," *IEEE Trans. Inf. Theory*, vol. 51, no. 1, pp. 101–119, Jan. 2005.

[94] E. Rosnes and O. Y. Takeshita, "Optimum distance quadratic permutation polynomial-based interleavers for turbo codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2006, pp. 1988–1992.

[95] O. Y. Takeshita, "Permutation polynomial interleavers: An algebraic-geometric perspective," *IEEE Trans. Inf. Theory*, vol. 53, no. 6, pp. 2116–2132, Jun. 2007.

[96] L. Trifina and D. Tarniceriu, *Permutation Polynomial Interleavers for Turbo Codes*. Singapore: Springer, 2019.

[97] O. Takeshita, "On maximum contention-free interleavers and permutation polynomials over integer rings," *IEEE Trans. Inf. Theory*, vol. 52, no. 3, pp. 1249–1253, Mar. 2006.

[98] S. Crozier and P. Guinand, "High-performance low-memory interleaver banks for turbo-codes," in *Proc. IEEE VTC*, vol. 4, 2001, pp. 2394–2398.

[99] S. Crozier, A. Gracie, P. Guinand, A. Hunt, R. Kerr, and J. Lodge, *Universal Turbo Code Design for Wireless Communication*. Hoboken, NJ, USA: CRC, Feb. 2011.

[100] S. Crozier and P. Guinand, "Distance upper bounds and true minimum distance results for turbo-codes designed with DRP interleavers," *Ann. Telecommun.*, vol. 60, nos. 1–2, pp. 10–28, 2005.

[101] S. Crozier and K. Gracie, "On the error-rate performance of 4-state turbo codes with puncture-constrained DRP interleavers," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2012, pp. 2601–2605.

[102] C. Berrou, Y. Saouter, C. Douillard, S. Kerouedan, and M. Jezequel, "Designing good permutations for turbo codes: Towards a single model," in *Proc. IEEE Int. Conf. Commun. (ICC)*, vol. 1, 2004, pp. 341–345.

[103] *Enhanced Turbo Codes for NR: Performance Evaluation*, 3GPP, Geneva, Switzerland, Aug. 2016.

[104] R. G. Bohórquez, C. A. Nour, and C. Douillard, "On the equivalence of interleavers for turbo codes," *IEEE Wireless Commun. Lett.*, vol. 4, no. 1, pp. 58–61, Feb. 2015.

[105] L. Trifina and D. Tarniceriu, "On the equivalence of cubic permutation polynomial and ARP interleavers for turbo codes," *IEEE Trans. Commun.*, vol. 65, no. 2, pp. 473–485, Feb. 2017.

[106] J. Li, Q. Chen, S. Gao, Z. Ma, and P. Fan, "The optimal puncturing pattern design for rate-compatible punctured turbo codes," in *Proc. Int. Conf. Wireless Commun. Signal Process.*, 2009, pp. 1–5.

[107] J.-F. Cheng, A. Nimbalker, Y. Blankenship, B. Classon, and T. K. Blankenship, "Analysis of circular buffer rate matching for LTE turbo code," in *Proc. IEEE VTC*, 2008, pp. 1–5.

[108] Y. Jiang, S. Kannan, H. Kim, S. Oh, H. Asnani, and P. Viswanath, "Deepturbo: Deep turbo decoder," in *Proc. Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, 2019, pp. 1–5.

[109] Y. He, J. Zhang, S. Jin, C.-K. Wen, and G. Y. Li, "Model-driven DNN decoder for turbo codes: Design, simulation, and experimental results," *IEEE Trans. Commun.*, vol. 68, no. 10, pp. 6127–6140, Oct. 2020.

[110] S. A. Hebbar, R. K. Mishra, S. K. Ankireddy, A. V. Makkuva, H. Kim, and P. Viswanath, "TinyTurbo: Efficient turbo decoders on edge," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2022, pp. 2797–2802.

[111] M. F. Brejza, L. Li, R. G. Maunder, B. M. Al-Hashimi, C. Berrou, and L. Hanzo, "20 years of turbo coding and energy-aware design guidelines for energy-constrained wireless applications," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 8–28, 1st Quart., 2016.

[112] P. H. P. Robertson and E. Villebrun, "Optimal and sub-optimal maximum *a posteriori* algorithms suitable for turbo decoding," *Eur. Trans. Telecommun.*, vol. 8, no. 2, pp. 119–125, 1997.

[113] J.-F. Cheng and T. Ottosson, "Linearly approximated log-MAP algorithms for turbo decoding," in *Proc. IEEE VTC*, vol. 3, 2000, pp. 2252–2256.

[114] H. Wang, H. Yang, and D. Yang, "Improved Log-MAP decoding algorithm for turbo-like codes," *IEEE Commun. Lett.*, vol. 10, no. 3, pp. 186–188, Mar. 2006.

[115] D.-H. Nguyen and H. Nguyen, "An improved Log-MAP algorithm based on polynomial regression function for LTE turbo decoding," in *Proc. IEEE Int. Conf. Commun. Workshop*, 2015, pp. 2163–2167.

[116] A. Worm, P. Hoeher, and N. Wehn, "Turbo-decoding without SNR estimation," *IEEE Commun. Lett.*, vol. 4, no. 6, pp. 193–195, Jun. 2000.

[117] A. F. J. Vogt, "Improving the Max-Log-MAP turbo decoder," *Electron. Lett*, vol. 36, no. 23, pp. 1937–1939, 2000.

[118] H. Claussen, H. R. Karimi, and B. Mulgrew, "Improved Max-Log-MAP turbo decoding by maximization of mutual information transfer," *EURASIP J. Appl. Signal Process.*, vol. 6, pp. 820–827, May 2005.

[119] R. G. Maunder, "A fully-parallel turbo decoding algorithm," *IEEE Trans. Commun.*, vol. 63, no. 8, pp. 2762–2775, Aug. 2015.

[120] M. Fossorier, F. Burkert, S. Lin, and J. Hagenauer, "On the equivalence between SOVA and max-log-MAP decodings," *IEEE Commun. Lett.*, vol. 2, no. 5, pp. 137–139, May 1998.

[121] V. H. S. Le, C. A. Nour, E. Boutillon, and C. Douillard, "Revisiting the max-log-map algorithm with SOVA update rules: New simplifications for high-radix SISO decoders," *IEEE Trans. Commun.*, vol. 68, no. 4, pp. 1991–2004, Apr. 2020.

[122] G. Battail, "Pondaration des symboles docodas par l algorithme de viterbi," *Ann. Telecommun.*, vol. 42, nos. 1–2, pp. 31–38, Jan. 1987.

[123] L. Lin and R. Cheng, "Improvements in SOVA-based decoding for turbo codes," in *Proc. IEEE Int. Conf. Commun. (ICC)*, vol. 3, 1997, pp. 1473–1478.

[124] R. Klaimi, S. Weithoffer, C. A. Nour, and C. Douillard, "Simplified recursion units for max-log-MAP: New trade-offs through variants of local-SOVA," in *Proc. Int. Symp. Topics Coding (ISTC)*, 2021, pp. 1–5.

[125] Y. Ould-Cheikh-Mouhamedou, S. Crozier, K. Gracie, P. Guinand, and P. Kabal, "A method for lowering turbo code error flare using correction impulses and repeated decoding," in *Proc. Int. Symp. Turbo Codes Rel. Topics (ISTC)*, 2006, pp. 1–6.

[126] Y. Ould-Cheikh-Mouhamedou and S. Crozier, "Improving the error rate performance of turbo codes using the forced symbol method," *IEEE Commun. Lett.*, vol. 11, no. 7, pp. 616–618, Jul. 2007.

[127] T. Tonnellier, C. Leroux, B. Le Gal, B. Gadat, C. Jego, and N. Van Wambeke, "Lowering the error floor of turbo codes with CRC verification," *IEEE Wireless Commun. Lett.*, vol. 5, no. 4, pp. 404–407, Aug. 2016.

[128] T. Gendron, E. Boutillon, C. A. Nour, and D. Gnaedig, "Revisiting augmented decoding techniques for LTE turbo codes," in *Proc. Int. Symp. Topics Coding (ISTC)*, 2021, pp. 1–5.

[129] M. J. Thul, F. Gilbert, T. Vogt, G. Kreiselmaier, and N. Wehn, "A scalable system architecture for high-throughput turbo-decoders," *J. VLSI Signal Process. Syst. Signal Video Technol.*, vol. 39, nos. 1–2, pp. 63–77, 2005.

[130] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "A soft-input soft-output maximum a posteriori (MAP) module to decode parallel and serial concatenated codes," JPL, Pasadena, CA, TDA Progr. Rep. 127, Nov. 1996.

[131] V. H. S. Le, "Design of next-generation Tb/s turbo codes," Ph.D. dissertation, Dept. Comput. Sci., IMT Atlantique, Nantes, France, 2021.

[132] T. Ilnseher, F. Kienle, C. Weis, and N. Wehn, "A 2.15GBit/s turbo code decoder for LTE advanced base station applications," in *Proc. Int. Symp. Turbo Codes Iterative Inf. Process (ISTC)*, 2012, pp. 21–25.

[133] R. Shrestha and R. P. Paily, "High-throughput turbo decoder with parallel architecture for LTE wireless communication standards," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 9, pp. 2699–2710, Sep. 2014.

[134] S. Weithoffer, O. Griebel, R. Klaimi, C. A. Nour, and N. Wehn, "Advanced hardware architectures for turbo code decoding beyond 100 Gb/s," in *Proc. WCNC*, 2020, pp. 1–6.

[135] A. Worm, H. Lamm, and N. Wehn, "A high-speed MAP architecture with optimized memory size and power consumption," in *Proc. IEEE Workshop Signal Process. Syst. (SiPS)*, 2000, pp. 265–274.

[136] S. Weithoffer, F. Pohl, and N. Wehn, "On the applicability of trellis compression to turbo-code decoder hardware architectures," in *Proc. Int. Symp. Turbo Codes Iterative Inf. Process (ISTC)*, 2016, pp. 61–65.

[137] G. Wang, H. Shen, Y. Sun, J. R. Cavallaro, A. Vosoughi, and Y. Guo, "Parallel interleaver design for a high throughput HSPA+/LTE multi-standard turbo decoder," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 5, pp. 1376–1389, May 2014.

[138] J. Zhang and M. Fossorier, "Shuffled iterative decoding," *IEEE Trans. Commun.*, vol. 53, no. 2, pp. 209–213, Feb. 2005.

[139] S. Weithoffer, C. A. Nour, N. Wehn, C. Douillard, and C. Berrou, "25 years of turbo codes: From Mb/s to beyond 100 Gb/s," in *Proc. Int. Symp. Turbo Codes Iterative Inf. Process (ISTC)*, 2018, pp. 1–6.

[140] S. Weithoffer, R. Klaimi, C. A. Nour, N. Wehn, and C. Douillard, "Fully pipelined iteration unrolled decoders the road to Tb/s turbo decoding," in *Proc. ICASSP*, 2020, pp. 5115–5119.

[141] E. Rosnes and O. Ytrehus, "On the design of bit-interleaved turbo-coded modulation with low error floors," *IEEE Trans. Commun.*, vol. 54, no. 9, pp. 1563–1573, Sep. 2006.

[142] A. Alvarado, E. Agrell, L. Szczecinski, and A. Svensson, "Exploiting UEP in QAM-based BICM: Interleaver and code design," *IEEE Trans. Commun.*, vol. 58, no. 2, pp. 500–510, Feb. 2010.

[143] R. Klaimi, "Study of non-binary turbo codes for future communication and broadcasting systems," Ph.D. dissertation, Dept. Comput. Sci., IMT Atlantique, Nantes, France, 2019.

[144] J. L. Massey and T. Mittelho, "Convolutional codes over rings," in *Proc. 4th Joint Swedish Soviet Int. Workshop Inf. Theory*, 1989, pp. 14–18.

[145] B. Frey and D. MacKay, "Irregular turbo codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2000, p. 121.

[146] J. Boutros, G. Caire, E. Viterbo, H. Sawaya, and S. Vialle, "Turbo code at 0.03 db from capacity limit," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2002, p. 56.

[147] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Serial concatenation of interleaved codes: Performance analysis, design, and iterative decoding," *IEEE Trans. Inf. Theory*, vol. 44, no. 3, pp. 909–926, May 1998.

[148] A. Graell i Amat, G. Montorsi, and F. Vatta, "Design and performance analysis of a new class of rate compatible serially concatenated convolutional codes," *IEEE Trans. Commun.*, vol. 57, no. 8, pp. 2280–2289, Aug. 2009.

[149] S. Moloudi, M. Lentmaier, and A. Graell i Amat, "Spatially coupled turbo-like codes: A new trade-off between waterfall and error floor," *IEEE Trans. Commun.*, vol. 67, no. 5, pp. 3114–3123, May 2019.

[150] L. Yang, Y. Xie, X. Wu, J. Yuan, X. Cheng, and L. Wan, "Partially information-coupled turbo codes for LTE systems," *IEEE Trans. Commun.*, vol. 66, no. 10, pp. 4381–4392, Oct. 2018.

[151] M. Qiu, X. Wu, A. Graell i Amat, and J. Yuan, "Analysis and design of partially information- and partially parity-coupled turbo codes," *IEEE Trans. Commun.*, vol. 69, no. 4, pp. 2107–2122, Apr. 2021.

[152] M. Qiu, X. Wu, J. Yuan, and A. Graell i Amat, "Generalized spatially-coupled parallel concatenated codes with partial repetition," *IEEE Trans. Commun.*, vol. 70, no. 9, pp. 5771–5787, May 2022.

[153] W. Zhang, M. Lentmaier, K. S. Zigangirov, and D. J. Costello, "Braided convolutional codes: A new class of turbo-like codes," *IEEE Trans. Inf. Theory*, vol. 56, no. 1, pp. 316–331, Jan. 2010.

[154] B. P. Smith, A. Farhood, A. Hunt, F. R. Kschischang, and J. Lodge, "Staircase codes: FEC for 100 Gb/s OTN," *J. Lightw. Technol.*, vol. 30, no. 1, pp. 110–117, Jan. 15, 2012.

[155] M. Qiu, L. Yang, Y. Xie, and J. Yuan, "Terminated staircase codes for NAND flash memories," *IEEE Trans. Commun.*, vol. 66, no. 12, pp. 5861–5875, Dec. 2018.

[156] M. Qiu and J. Yuan, "Sub-block rearranged staircase codes," *IEEE Trans. Commun.*, vol. 70, no. 9, pp. 5695–5710, Jan. 2022.

[157] A. Y. Sukmadji, U. Martinez-Peas, and F. R. Kschischang, "Zipper codes," *J. Lightw. Technol.*, vol. 40, no. 19, pp. 6397–6407, Oct. 1, 2022.

[158] M. Lentmaier, A. Sridharan, D. J. Costello, and K. S. Zigangirov, "Iterative decoding threshold analysis for LDPC convolutional codes," *IEEE Trans. Inf. Theory*, vol. 56, no. 10, pp. 5274–5289, Oct. 2010.

[159] S. Kudekar, T. J. Richardson, and R. L. Urbanke, "Threshold saturation via spatial coupling: Why convolutional LDPC ensembles perform so well over the BEC," *IEEE Trans. Inf. Theory*, vol. 57, no. 2, pp. 803–834, Feb. 2011.

[160] A. R. Iyengar, P. H. Siegel, R. L. Urbanke, and J. K. Wolf, "Windowed decoding of spatially coupled codes," *IEEE Trans. Inf. Theory*, vol. 59, no. 4, pp. 2277–2292, Apr. 2013.

[161] C. Rachinger, J. B. Huber, and R. R. Müller, "Comparison of convolutional and block codes for low structural delay," *IEEE Trans. Commun.*, vol. 63, no. 12, pp. 4629–4638, Dec. 2015.

[162] A. Yedla, Y.-Y. Jian, P. S. Nguyen, and H. D. Pfister, "A simple proof of threshold saturation for coupled scalar recursions," in *Proc. Int. Symp. Turbo Codes Iterative Inf. Process (ISTC)*, 2012, pp. 51–55.

[163] A. Yedla, Y.-Y. Jian, P. S. Nguyen, and H. D. Pfister, "A simple proof of maxwell saturation for coupled scalar recursions," *IEEE Trans. Inf. Theory*, vol. 60, no. 11, pp. 6943–6965, Nov. 2014.

[164] A. Yardi, T. Benaddi, C. Poulliat, and I. Andriyanova, "EBP-GEXIT charts for M-ary AWGN channel for generalized LDPC and turbo codes," *IEEE Trans. Commun.*, vol. 70, no. 6, pp. 3613–3626, Jun. 2022.

[165] C. Measson, A. Montanari, T. J. Richardson, and R. Urbanke, "The generalized area theorem and some of its consequences," *IEEE Trans. Inf. Theory*, vol. 55, no. 11, pp. 4793–4821, Nov. 2009.

[166] C. Yang, S. Zhao, and X. Ma, "Hybrid coupled serially concatenated codes," *IEEE Trans. Commun.*, vol. 70, no. 7, pp. 4301–4315, Jul. 2022.

[167] M. Mahdavi et al., "Spatially coupled serially concatenated codes: Performance evaluation and VLSI design tradeoffs," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 69, no. 5, pp. 1962–1975, May 2022.

[168] M. Zhu, D. G. M. Mitchell, M. Lentmaier, D. J. Costello, and B. Bai, "Error propagation mitigation in sliding window decoding of braided convolutional codes," *IEEE Trans. Commun.*, vol. 68, no. 11, pp. 6683–6698, Nov. 2020.

[169] X. Wu, M. Qiu, and J. Yuan, "Partially information coupled duo-binary turbo codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2020, pp. 461–466.

[170] M. Qiu, X. Wu, J. Yuan, and A. Graell i Amat, "Generalized spatially coupled parallel concatenated convolutional codes with partial repetition," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2021, pp. 581–586.

[171] R. E. G. Bohorquez, "Advanced coding and interleaving techniques for next generation communication and broadcasting systems," Ph.D. dissertation, Dept. Comput. Sci., Telecom Bretagne, Givrand, France, 2015.

[172] M. Baldi, M. Bianchi, F. Chiaraluce, R. Garello, I. A. Sanchez, and S. Cioni, "Advanced channel coding for space mission telecommand links," in *Proc. IEEE VTC*, 2013, pp. 1–5.

[173] G. Liva, E. Paolini, B. Matuz, S. Scalise, and M. Chiani, "Short turbo codes over high order fields," *IEEE Trans. Commun.*, vol. 61, no. 6, pp. 2201–2211, Jun. 2013.

[174] V. Y. F. Tan and M. Tomamichel, "The third-order term in the normal approximation for the AWGN channel," *IEEE Trans. Inf. Theory*, vol. 61, no. 5, pp. 2430–2438, May 2015.

[175] C. Roth, S. Belfanti, C. Benkeser, and Q. Huang, "Efficient parallel turbo-decoding for high-throughput wireless systems," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 6, pp. 1824–1835, Jun. 2014.

[176] A. Li, L. Xiang, T. Chen, R. G. Maunder, B. M. Al-Hashimi, and L. Hanzo, "VLSI implementation of fully parallel LTE turbo decoders," *IEEE Access*, vol. 4, pp. 323–346, 2016.

[177] A. Osmani and H. M. Trujillo, *Analysis of the Finite Length Performance of Spatially Coupled Convolutional Codes*, Lund Univ., Lund, Sweden, 2015.

[178] G. Iro and R. Kabbinale, *Comparison of Various Concatenated Convolutional Code Ensembles Under Spatial Coupling*, Lund Univ., Lund, Sweden, 2017,

[179] M. Mahdavi, M. U. Farooq, L. Liu, O. Edfors, V. Awall, and M. Lentmaier, "The effect of coupling memory and block length on spatially coupled serially concatenated codes," in *Proc. IEEE VTC*, 2021, pp. 1–7.

[180] D. G. M. Mitchell, M. Lentmaier, and D. J. Costello, "Spatially coupled LDPC codes constructed from protographs," *IEEE Trans. Inf. Theory*, vol. 61, no. 9, pp. 4866–4889, Sep. 2015.

[181] D. G. M. Mitchell, M. Lentmaier, A. E. Pusane, and D. J. Costello, "Randomly punctured LDPC codes," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 2, pp. 408–421, Feb. 2016.

[182] M. Zhu, D. G. M. Mitchell, M. Lentmaier, D. J. Costello, and B. Bai, "Braided convolutional codes with sliding window decoding," *IEEE Trans. Commun.*, vol. 65, no. 9, pp. 3645–3658, Sep. 2017.

[183] R. Gallager, "Low-density parity-check codes," *IRE Trans. Inf. Theory*, vol. 8, no. 1, pp. 21–28, Jan. 1962.

[184] D. J. C. MacKay and R. M. Neal, "Good codes based on very sparse matrices," in *Cryptography and Coding* (Lecture Notes in Computer Science), vol. 1025. Berlin, Germany: Springer, 1995.

[185] D. J. C. MacKay and R. M. Neal, "Near shannon limit performance of low density parity check codes," *Electron. Lett.*, vol. 32, no. 18, pp. 1645–1646.

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

[187] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*. Amsterdam, The Netherlands: Elsevier Sci., 1988.

[188] *Digital Video Broadcasting (DVB); Second Generation Framing Structure, Channel Coding and Modulation Systems for Broadcasting, Interactive Services, News Gathering and Other Broadband Satellite Applications; Part 2: DVB-s2 Extensions (DVB-s2x)*, ETSI Standard EN 302 307-2, 2010.

[189] T. Richardson, M. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 619–637, Feb. 2001.

[190] M. Luby, M. Mitzenmacher, M. Shokrollahi, and D. Spielman, "Improved low-density parity-check codes using irregular graphs," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 585–598, Feb. 2001.

[191] 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 Commun. Lett.*, vol. 5, no. 2, pp. 58–60, Feb. 2001.

[192] D. Divsalar and C. Jones, "Protograph based low error floor LDPC coded modulation," in *Proc. MILCOM IEEE Mil. Commun. Conf.*, vol. 1, 2005, pp. 378–385.

[193] D. Divsalar, S. Dolinar, and C. Jones, "Construction of protograph LDPC codes with linear minimum distance," in *Proc. IEEE Int. Symp. Inf. Theory*, 2006, pp. 664–668.

[194] G. Liva, W. E. Ryan, and M. Chiani, "Quasi-cyclic generalized LDPC codes with low error floors," *IEEE Trans. Commun.*, vol. 56, no. 1, pp. 49–57, Jan. 2008.

[195] S. Abu-Surra, D. Divsalar, and W. E. Ryan, "Enumerators for protograph-based ensembles of LDPC and generalized LDPC codes," *IEEE Trans. Inf. Theory*, vol. 57, no. 2, pp. 858–886, Feb. 2011.

[196] D. Divsalar, H. Jin, R. McEliece, "Coding theorms for turbo-like codes," 1998. [Online]. Available: https://api.semanticscholar.org/CorpusID:1045655

[197] A. Abbasfar, D. Divsalar, and K. Yao, "Accumulate-repeat-accumulate codes," *IEEE Trans. Commun.*, vol. 55, no. 4, pp. 692–702, Apr. 2007.

[198] R. Echard and S.-C. Chang, "The pi-rotation low-density parity check codes," in *Proc. IEEE Global Telecommun. Conf.*, vol. 2, 2001, pp. 980–984.

[199] H. Li, B. Bai, X. Mu, J. Zhang, and H. Xu, "Algebra-assisted construction of quasi-cyclic LDPC codes for 5G new radio," *IEEE Access*, vol. 6, pp. 50229–50244, 2018.

[200] "Low density parity check codes for use in near-earth and deep space applications." 2007. [Online]. Available: https://public.ccsds.org/Pubs/131x1o2e2s.pdf

[201] T.-Y. Chen, K. Vakilinia, D. Divsalar, and R. D. Wesel, "Protograph-based raptor-like LDPC codes," *IEEE Trans. Commun.*, vol. 63, no. 5, pp. 1522–1532, May 2015.

[202] S. V. S. Ranganathan, D. Divsalar, and R. D. Wesel, "Quasi-cyclic protograph-based raptor-like LDPC codes for short block-lengths," *IEEE Trans. Inf. Theory*, vol. 65, no. 6, pp. 3758–3777, Jun. 2019.

[203] A. Shokrollahi, "Raptor codes," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2551–2567, Jun. 2006, doi: 10.1561/0100000060.

[204] M. Luby, "LT codes," in *Proc. 43rd Annu. IEEE Symp. Found. Comput. Sci.*, 2002, pp. 271–280.

[205] S.-Y. Chung, T. Richardson, and R. Urbanke, "Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 657–670, Feb. 2001.

[206] A. Ashikhmin, G. Kramer, and S. T. Brink, "Extrinsic information transfer functions: Model and erasure channel properties," *IEEE Trans. Inf. Theory*, vol. 50, no. 11, pp. 2657–2673, Nov. 2004.

[207] W. Ryan and S. Lin, *Channel Codes: Classical and Modern*. Cambridge, U.K.: Cambridge Univ. Press, 2015.

[208] G. Liva and M. Chiani, "Protograph LDPC codes design based on EXIT analysis," in *Proc. IEEE Global Telecommun. Conf.*, 2007, pp. 3250–3254.

[209] D. Divsalar, S. Dolinar, C. R. Jones, and K. Andrews, "Capacity-approaching protograph codes," *IEEE J. Sel. Areas Commun.*, vol. 27, no. 6, pp. 876–888, Aug. 2009.

[210] A. Amraoui, A. Montanari, T. Richardson, and R. Urbanke, "Finite-length scaling for iteratively decoded LDPC ensembles," *IEEE Trans. Inf. Theory*, vol. 55, no. 2, pp. 473–498, Feb. 2009.

[211] X.-Y. Hu, E. Eleftheriou, and D.-M. Arnold, "Progressive edge-growth tanner graphs," in *Proc. IEEE Global Telecommun. Conf.*, vol. 2, 2001, pp. 995–1001.

[212] X.-Y. Hu, E. Eleftheriou, and D. M. Arnold, "Regular and irregular progressive edge-growth tanner graphs," *IEEE Trans. Inf. Theory*, vol. 51, no. 1, pp. 386–398, Jan. 2005.

[213] J. L. Fan, "Array codes as LDPC codes," in *Constrained Coding and Soft Iterative Decoding* (The Springer International Series in Engineering and Computer Science), J. L. Fan, Ed. Boston, MA, USA: Springer, 2001, pp. 195–203.

[214] J. Kang, Q. Huang, L. Zhang, B. Zhou, and S. Lin, "Quasi-cyclic LDPC codes: An algebraic construction," *IEEE Trans. Commun.*, vol. 58, no. 5, pp. 1383–1396, Nov. 2010.

[215] L. Zhang, Q. Huang, S. Lin, K. Abdel-Ghaffar, and I. F. Blake, "Quasi-cyclic LDPC codes: An algebraic construction, rank analysis, and codes on latin squares," *IEEE Trans. Commun.*, vol. 58, no. 11, pp. 3126–3139, Nov. 2010.

[216] Y. Kou, S. Lin, and M. P. C. Fossorier, "Low-density parity-check codes based on finite geometries: A rediscovery and new results," *IEEE Trans. Inf. Theory*, vol. 47, no. 7, pp. 2711–2736, Nov. 2001.

[217] D. V. Nguyen, S. K. Chilappagari, M. W. Marcellin, and B. Vasic, "On the construction of structured LDPC codes free of small trapping sets," *IEEE Trans. Inf. Theory*, vol. 58, no. 4, pp. 2280–2302, Apr. 2012.

[218] B. Ammar, B. Honary, Y. Kou, J. Xu, and S. Lin, "Construction of low-density parity-check codes based on balanced incomplete block designs," *IEEE Trans. Inf. Theory*, vol. 50, no. 6, pp. 1257–1269, Jun. 2004.

[219] I. Djurdjevic, J. Xu, K. Abdel-Ghaffar, and S. Lin, "A class of low-density parity-check codes constructed based on reed-solomon codes with two information symbols," *IEEE Commun. Lett.*, vol. 7, no. 7, pp. 317–319, Jul. 2013.

[220] B. Vasic and O. Milenkovic, “Combinatorial constructions of low-density parity-check codes for iterative decoding,” *IEEE Trans. Inf. Theory*, vol. 50, no. 6, pp. 1156–1176, Jun. 2004.

[221] L. Chen, J. Xu, I. Djurdjevic, and S. Lin, “Near-shannon-limit quasi-cyclic low-density parity-check codes,” *IEEE Trans. Commun.*, vol. 52, no. 7, pp. 1038–1042, Jul. 2004.

[222] M. P. C. Fossorier, “Quasi-cyclic low-density parity-check codes from circulant permutation matrices,” *IEEE Trans. Inf. Theory*, vol. 50, no. 8, pp. 1788–1793, Aug. 2004.

[223] J. Xu, L. Chen, L. Zeng, L. Lan, and S. Lin, “Construction of low-density parity-check codes by superposition,” *IEEE Trans. Commun.*, vol. 53, no. 2, pp. 243–251, Feb. 2005.

[224] H. Tang, J. Xu, S. Lin, and K. Abdel-Ghaffar, “Codes on finite geometries,” *INRIA*, vol. 51, no. 2, pp. 572–596, 2020.

[225] L. Lan, Y. Y. Tai, S. Lin, B. Memari, and B. Honary, “New constructions of quasi-cyclic LDPC codes based on special classes of BIBD’s for the AWGN and binary erasure channels,” *IEEE Trans. Commun.*, vol. 56, no. 1, pp. 39–48, Jan. 2008.

[226] L. Sassatelli and D. Declercq, “Nonbinary hybrid LDPC codes,” *IEEE Trans. Inf. Theory*, vol. 56, no. 10, pp. 5314–5334, Oct. 2010.

[227] S. Song, B. Zhou, S. Lin, and K. Abdel-Ghaffar, “A unified approach to the construction of binary and nonbinary quasi-cyclic LDPC codes based on finite fields,” *IEEE Trans. Commun.*, vol. 57, no. 1, pp. 84–93, Jan. 2009.

[228] Q. Huang, Q. Diao, S. Lin, and K. Abdel-Ghaffar, “Cyclic and quasi-cyclic LDPC codes on constrained parity-check matrices and their trapping sets,” *IEEE Trans. Inf. Theory*, vol. 58, no. 5, pp. 2648–2671, May 2012.

[229] J. Li, K. Liu, S. Lin, and K. Abdel-Ghaffar, “Algebraic quasi-cyclic LDPC codes: Construction, low error-floor, large girth and a reduced-complexity decoding scheme,” *IEEE Trans. Commun.*, vol. 62, no. 8, pp. 2626–2637, Aug. 2014.

[230] D. Divsalar, S. Dolinar, and C. Jones, “Low-rate LDPC codes with simple protograph structure,” in *Proc. IEEE Int. Symp. Inf. Theory*, 2005, pp. 1622–1626.

[231] D. Divsalar, “Ensemble weight enumerators for protograph LDPC codes,” in *Proc. IEEE Int. Symp. Inf. Theory*, 2006, pp. 1554–1558.

[232] K. Andrews, S. Dolinar, and J. Thorpe, “Encoders for block-circulant LDPC codes,” in *Proc. IEEE Int. Symp. Inf. Theory*, 2005, pp. 2300–2304.

[233] D. MacKay, G. Mitchison, and P. McFadden, “Sparse-graph codes for quantum error correction,” *IEEE Trans. Inf. Theory*, vol. 50, no. 10, pp. 2315–2330, Oct. 2004.

[234] Y. Xie and J. Yuan, “Protograph quantum LDPC codes from tensor product of parity-check matrices,” in *Proc. IEEE Globecom Workshops*, 2015, pp. 1–5.

[235] Y. Xie and J. Yuan, “Reliable quantum LDPC codes over GF(4),” in *Proc. IEEE Globecom Workshops*, 2016, pp. 1–5.

[236] Y. Xie, J. Yuan, and Q. T. Sun, “Design of quantum LDPC codes from quadratic residue sets,” *IEEE Trans. Commun.*, vol. 66, no. 9, pp. 3721–3735, Sep. 2018.

[237] Z. Li and B. Kumar, “A class of good quasi-cyclic low-density parity check codes based on progressive edge growth graph,” in *Proc. Conf. 38th Asilomar Conf. Signals Syst. Comput.*, vol. 2, 2004, pp. 1990–1994.

[238] X. Liu, W. Zhang, and Z. Fan, “Construction of quasi-cyclic LDPC codes and the performance on the PR4-equalized MRC channel,” *IEEE Trans. Mag.*, vol. 45, no. 10, pp. 3699–3702, Oct. 2009.

[239] Y. Xie, J. C. Mu, and J. Yuan, “Design of rate-compatible protograph-based LDPC codes with mixed circulants,” in *Proc. 6th Int. Symp. Turbo Codes Iterative Inf. Process.*, 2010, pp. 434–438.

[240] R. Smarandache and D. G. M. Mitchell, “A unifying framework to construct QC-LDPC tanner graphs of desired girth,” *IEEE Trans. Inf. Theory*, vol. 68, no. 9, pp. 5802–5822, Sep. 2022.

[241] “TM synchronization and channel coding.” 2020. [Online]. Available: https://public.ccsds.org/Pubs/130x1g3.pdf

[242] R. C. Bose and D. K. Ray-Chaudhuri, “On a class of error correcting binary group codes,” *Inf. Control*, vol. 3, no. 1, pp. 68–79, 1960.

[243] A. Hocquenghem. “Codes correcteurs derreurs.” 2021. [Online]. Available: http://v.vincent.u-bourgogne.fr/0ENS/THEO-INFO/CM_M1_3.pdf

[244] I. S. Reed and G. Solomon, “Polynomial codes over certain finite fields,” *J. Soc. Ind. Appl. Math.*, vol. 8, no. 2, pp. 300–304, Jun. 1960.

[245] A. Jimenez and K. Zigangirov, “Periodic time-varying convolutional codes with low-density parity-check matrices,” in *Proc. IEEE Int. Symp. Inf. Theory*, 1999, p. 305.

[246] A. J. Felstrom and K. S. Zigangirov, “Time-varying periodic convolutional codes with low-density parity-check matrix,” *IEEE Trans. Inf. Theory*, vol. 45, no. 6, pp. 2181–2191, Sep. 1999.

[247] D. J. Costello, L. Dolecek, T. E. Fuja, J. Kliewer, D. G. Mitchell, and R. Smarandache, “Spatially coupled sparse codes on graphs: Theory and practice,” *IEEE Commun. Mag.*, vol. 52, no. 7, pp. 168–176, Jul. 2014.

[248] T. Richardson and R. Urbanke, *Modern Coding Theory*. Cambridge, U.K.: Cambridge Univ. Press, 1990.

[249] S. Kudekar, T. Richardson, and R. L. Urbanke, “Spatially coupled ensembles universally achieve capacity under belief propagation,” *IEEE Trans. Inf. Theory*, vol. 59, no. 12, pp. 7761–7813, Dec. 2013.

[250] A. E. Pusane, R. Smarandache, P. O. Vontobel, and D. J. Costello, “Deriving good LDPC convolutional codes from LDPC block codes,” *IEEE Trans. Inf. Theory*, vol. 57, no. 2, pp. 835–857, Feb. 2011.

[251] A. Yedla, P. S. Nguyen, H. D. Pfister, and K. R. Narayanan, “Universal codes for the Gaussian MAC via spatial coupling,” in *Proc. 49th Annu. Allerton Conf. Commun. Control Comput. (Allerton)*, 2011, pp. 1801–1808.

[252] A. R. Iyengar, M. Papaleo, P. H. Siegel, J. K. Wolf, A. Vanelli-Coralli, and G. E. Corazza, “Windowed decoding of protograph-based LDPC convolutional codes over erasure channels,” *IEEE Trans. Inf. Theory*, vol. 58, no. 4, pp. 2303–2320, Apr. 2012.

[253] A. E. Pusane, A. J. Feltstrom, A. Sridharan, M. Lentmaier, K. S. Zigangirov, and D. J. Costello, “Implementation aspects of LDPC convolutional codes,” *IEEE Trans. Commun.*, vol. 56, no. 7, pp. 1060–1069, Jul. 2008.

[254] P. Kang, Y. Xie, L. Yang, and J. Yuan, “Reliability-based windowed decoding for spatially coupled LDPC codes,” *IEEE Commun. Lett.*, vol. 22, no. 7, pp. 1322–1325, Jul. 2018.

[255] M. Lentmaier, M. M. Prenda, and G. P. Fettweis, “Efficient message passing scheduling for terminated LDPC convolutional codes,” in *Proc. IEEE Int. Symp. Inf. Theory*, 2011, pp. 1826–1830.

[256] N. Ul Hassan, A. E. Pusane, M. Lentmaier, G. P. Fettweis, and D. J. Costello, “Non-uniform window decoding schedules for spatially coupled LDPC codes,” *IEEE Trans. Commun.*, vol. 65, no. 2, pp. 501–510, Mar. 2017.

[257] I. Ali, J.-H. Kim, S.-H. Kim, H. Kwak, and J.-S. No, “Improving windowed decoding of SC LDPC codes by effective decoding termination, message reuse, and amplification,” *IEEE Access*, vol. 6, pp. 9336–9346, 2017.

[258] K. Klaiber, S. Cammerer, L. Schmalen, and S. T. Brink, “Avoiding burst-like error patterns in windowed decoding of spatially coupled LDPC codes,” in *Proc. IEEE 10th Int. Symp. Turbo Codes Iterative Inf. Process. (ISTC)*, 2018, pp. 1–5.

[259] Y. Xie, L. Yang, P. Kang, and J. Yuan, “Euclidean geometry-based spatially coupled LDPC codes for storage,” *IEEE J. Sel. Areas Commun.*, vol. 34, no. 9, pp. 2498–2509, Sep. 2016.

[260] D. Truhachev, D. G. M. Mitchell, M. Lentmaier, D. J. Costello, and A. Karami, “Code design based on connecting spatially coupled graph chains,” *IEEE Trans. Inf. Theory*, vol. 65, no. 9, pp. 5604–5617, Sep. 2019.

[261] A. E. Dehaghani, M.-R. Sadeghi, and F. Amirzade, “Improving asymptotic properties of loop construction of SC-LDPC chains over the BEC,” *IEEE Commun. Lett.*, vol. 26, no. 3, pp. 495–499, Mar. 2022.

[262] Y. Liao, M. Qiu, and J. Yuan, “Self-connected spatially coupled LDPC codes with improved termination,” *IEEE Commun. Lett.*, vol. 27, no. 8, pp. 1959–1963, Aug. 2023.

[263] J. Li, S. Lin, K. Abdel-Ghaffar, W. E. Ryan, and D. J. Costello, “Globally coupled LDPC codes,” in *Proc. Inf. Theory Appl. Workshop (ITA)*, 2010, pp. 1–10.

[264] M. Nasseri, X. Xiao, B. Vasić, and S. Lin, “Globally coupled finite geometry and finite field LDPC coding schemes,” *IEEE Trans. Veh. Technol.*, vol. 70, no. 9, pp. 9207–9216, Sep. 2021.

[265] J. Li, K. Liu, S. Lin, and K. Abdel-Ghaffar, “Reed–solomon based nonbinary globally coupled LDPC codes: Correction of random errors and bursts of erasures,” in *Proc. IEEE Int. Symp. Inf. Theory*, 2017, pp. 381–385.

[266] J. Zhang, B. Bai, X. Mu, H. Xu, Z. Liu, and H. Li, "Construction and decoding of rate-compatible globally coupled LDPC codes," *Wireless Commun. Mobile Comput.*, vol. 2018, Feb. 2018, Art. no. 4397671.

[267] Q. Wang, S. Cai, and X. Ma, "Free-ride coding for constructions of coupled LDPC codes," *IEEE Trans. Commun.*, vol. 71, no. 3, pp. 1259–1270, Mar. 2023.

[268] L. Yang, Y. Xie, J. Yuan, X. Cheng, and L. Wan, "Chained LDPC codes for future communication systems," *IEEE Commun. Lett.*, vol. 22, no. 5, pp. 898–901, May 2018.

[269] *IEEE Standard for Information Technology—Telecommunications and Information Exchange Between Systems—LAN/MAN—Specific Requirements Part 3: CSMA/CD Access Method and Physical Layer Specifications—Amendment: Physical Layer and Management Parameters for 10 GB/S Operation, Type 10gbase-t*, IEEE Standard 802.11bd-2022, 2022.

[270] *IEEE Standard for Information Technology—Local and Metropolitan Area Networks—Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 5: Enhancements for Higher Throughput*, IEEE Standard TS 302 663, Jan. 2020.

[271] Z. Li, L. Chen, L. Zeng, S. Lin, and W. Fong, "Efficient encoding of quasi-cyclic low-density parity-check codes," *IEEE Trans. Commun.*, vol. 54, no. 1, pp. 71–81, Nov. 2005.

[272] *IEEE Standard for Information Technology—Telecommunications and Information Exchange Between Systems—Local and Metropolitan Area Networks—Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 3: Enhancements for Very High Throughput in the 60 GHz Band*, IEEE Standard 802.11bd-2022, 2022.

[273] T. Richardson and R. Urbanke, "Efficient encoding of low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 638–656, Feb. 2001.

[274] *IEEE Standard for Local and Metropolitan Area Networks—Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems—Amendment for Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands*, IEEE Standard 802.16-2004, 2004.

[275] *IEEE Standard for Information Technology—Local and Metropolitan Area Networks—Specific Requirements—Part 22: Cognitive Wireless RAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Policies and Procedures for Operation in the TV Bands*, IEEE Standard 802.15, 2005.

[276] T. Tian, C. Jones, J. Villasenor, and R. Wesel, "Selective avoidance of cycles in irregular LDPC code construction," *IEEE Trans. Commun.*, vol. 52, no. 8, pp. 1242–1247, Aug. 2004.

[277] *Digital Video Broadcasting (DVB); Second Generation Framing Structure, Channel Coding and Modulation Systems for Broadcasting, Interactive Services, News Gathering and Other Broadband Satellite Applications (DVB-s2)*, ETSI Standard EN 302 307, 2009.

[278] *Chairman's Notes RAN1_87_7.1.5_Final*, 3GPP, Sophia Antipolis, France, 2012.

[279] *5G; NR; Multiplexing and Channel Coding (3GPP TS 38.212 Version 17.4.0 Release 17)*, 3GPP Standard 138 212, 2023.

[280] M. Fossorier, M. Mihaljevic, and H. Imai, "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation," *IEEE Trans. Commun.*, vol. 47, no. 5, pp. 673–680, May 1999.

[281] E. Eleftheriou, T. Mittelholzer, and A. Dholakia, "Reduced-complexity decoding algorithm for low-density parity-check codes," *IEEE Trans. Circuits Syst., I, Regular Papers*, vol. 37, no. 2, pp. 102–104, Jul. 2014.

[282] X.-Y. Hu, E. Eleftheriou, D.-M. Arnold, and A. Dholakia, "Efficient implementations of the sum-product algorithm for decoding LDPC codes," in *Proc. IEEE Global Telecommun. Conf.*, vol. 2, 2001, pp. 1036–1036.

[283] J. Chen, A. Dholakia, E. Eleftheriou, M. Fossorier, and X.-Y. Hu, "Reduced-complexity decoding of LDPC codes," *IEEE Trans. Commun.*, vol. 53, no. 8, pp. 1288–1299, Aug. 2005.

[284] J. Chen and M. Fossorier, "Near optimum universal belief propagation based decoding of low-density parity check codes," *IEEE Trans. Commun.*, vol. 50, no. 3, pp. 406–414, Mar. 2002.

[285] J. Chen, R. Tanner, C. Jones, and Y. Li, "Improved min-sum decoding algorithms for irregular LDPC codes," in *Proc. IEEE Int. Symp. Inf. Theory*, 2005, pp. 449–453.

[286] J. Chen and M. P. C. Fossorier, "Density evolution for two improved BP-based decoding algorithms of LDPC codes," *IEEE Commun. Lett.*, vol. 6, no. 5, pp. 208–210, May 2002.

[287] J. Zhao, F. Zarkeshvari, and A. Banihashemi, "On implementation of min-sum algorithm and its modifications for decoding low-density parity-check (LDPC) codes," *IEEE Trans. Commun.*, vol. 53, no. 4, pp. 549–554, Apr. 2005.

[288] J. Zhang, M. Fossorier, D. Gu, and J. Zhang, "Two-dimensional correction for min-sum decoding of irregular LDPC codes," *IEEE Commun. Lett.*, vol. 10, no. 3, pp. 180–182, Mar. 2006.

[289] P. Kang, Y. Xie, L. Yang, C. Zheng, J. Yuan, and Y. Wei, "Enhanced quasi-maximum likelihood decoding of short LDPC codes based on saturation," in *Proc. IEEE Inf. Theory Workshop (ITW)*, 2019, pp. 1–5.

[290] P. Kang, Y. Xie, L. Yang, and J. Yuan, "Enhanced quasi-maximum likelihood decoding based on 2D modified min-sum algorithm for 5G LDPC codes," *IEEE Trans. Commun.*, vol. 68, no. 11, pp. 6669–6682, Nov. 2020.

[291] V. Savin, "Self-corrected min-sum decoding of LDPC codes," in *Proc. IEEE Int. Symp. Inf. Theory*, 2008, pp. 146–150.

[292] J. Andrade, G. Falcao, V. Silva, J. P. Barreto, N. Goncalves, and V. Savin, "Near-LSPA performance at MSA complexity," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2013, pp. 3281–3285.

[293] E. Amador, V. Rezard, and R. Pacalet, "Energy efficiency of SISO algorithms for turbo-decoding message-passing LDPC decoders," in *Proc. 17th IFIP Int. Conf. Very Large Scale Integr. (VLSI-SoC)*, 2009, pp. 95–100.

[294] E. Amador, R. Knopp, V. Rezard, and R. Pacalet, "Hybrid iteration control on LDPC decoders," in *Proc. 6th Int. Conf. Wireless Mobile Commun.*, 2010, pp. 102–106.

[295] C. Jones, E. Valles, M. Smith, and J. Villasenor, "Approximate-MIN constraint node updating for LDPC code decoding," in *Proc. IEEE Mil. Commun. Conf.*, vol. 1, 2003, pp. 157–162.

[296] T. J. Richardson, S. Kudekar, and V. Loncke, "Adjusted min-sum decoder," U.S. Patent 20 180 109 269, 2023.

[297] *LDPC Decoding With Adjusted Min-Sum*, 3GPP, Sophia Antipolis, France, 2010.

[298] T. Etzion, A. Trachtenberg, and A. Vardy, "Which codes have cycle-free tanner graphs?" *IEEE Trans. Inf. Theory*, vol. 45, no. 6, pp. 2173–2181, Sep. 1999.

[299] M. Fossorier and S. Lin, "Soft-decision decoding of linear block codes based on ordered statistics," *IEEE Trans. Inf. Theory*, vol. 41, no. 5, pp. 1379–1396, Sep. 1995.

[300] M. Fossorier, "Iterative reliability-based decoding of low-density parity check codes," *IEEE J. Sel. Areas Commun.*, vol. 19, no. 5, pp. 908–917, May 2001.

[301] N. Varnica, M. P. C. Fossorier, and A. Kavcic, "Augmented belief propagation decoding of low-density parity check codes," *IEEE Trans. Commun.*, vol. 55, no. 7, pp. 1308–1317, Jul. 2007.

[302] S. Scholl, P. Schlafer, and N. Wehn, "Saturated min-sum decoding: An 'afterburner' for LDPC decoder hardware," in *Proc. Design Autom. Test Europe Conf. Exhibit. (DATE)*, 2016, pp. 1219–1224.

[303] S. Kudekar, T. J. Richardson, G. Sarkis, and V. Loncke, "Efficient list decoding of LDPC codes," U.S. Patent 10 511 328 B2, 2020.

[304] B. Liu, Y. Xie, L. Yang, and J. Yuan, "An iterative soft-decision decoding algorithm with dynamic saturation for short reed-solomon codes," in *Proc. IEEE Inf. Theory Workshop (ITW)*, 2018, pp. 1–5.

[305] M. Borgerding, P. Schniter, and S. Rangan, "AMP-inspired deep networks for sparse linear inverse problems," *IEEE Trans. Signal Process.* vol. 65, no. 16, pp. 4293–4308, Jul. 2017.

[306] A. Zappone, M. Di Renzo, and M. Debbah, "Wireless networks design in the era of deep learning: Model-based, AI-based, or both?" *IEEE Trans. Commun.*, vol. 67, no. 10, pp. 7331–7376, Oct. 2019.

[307] H. He, C.-K. Wen, S. Jin, and G. Y. Li, "Model-driven deep learning for MIMO detection," *IEEE Trans. Signal Process.*, vol. 68, pp. 1702–1715, 2020.

[308] B. Liu, Z. Wei, W. Yuan, J. Yuan, and M. Pajovic, "Channel estimation and user identification with deep learning for massive machine-type communications," *IEEE Trans. Veh. Technol.*, vol. 70, no. 10, pp. 10709–10722, Oct. 2021.

[309] B. Liu, S. Li, Y. Xie, and J. Yuan, "A novel sum-product detection algorithm for faster-than-nyquist signaling: A deep learning approach," *IEEE Trans. Commun.*, vol. 69, no. 9, pp. 5975–5987, Sep. 2021.

[310] J. R. Hershey, J. L. Roux, and F. Weninger. "Deep unfolding: Model-based inspiration of novel deep architectures." 2014. [Online]. Available: https://arxiv.org/abs/1409.2574

[311] E. Nachmani, Y. Be'ery, and D. Burshtein, "Learning to decode linear codes using deep learning," in *Proc. 54th Annu. Allerton Conf. Commun. Control Comput. (Allerton)*, 2016, pp. 341–346.

[312] E. Nachmani et al., "Deep learning methods for improved decoding of linear codes," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 119–131, Feb. 2018.

[313] I. Be'Ery, N. Raviv, T. Raviv, and Y. Be'Ery, "Active deep decoding of linear codes," *IEEE Trans. Commun.*, vol. 68, no. 2, pp. 728–736, Nov. 2019.

[314] B. Liu, Y. Xie, and J. Yuan, "A deep learning assisted node-classified redundant decoding algorithm for BCH codes," *IEEE Trans. Commun.*, vol. 68, no. 9, pp. 5338–5349, Sep. 2020.

[315] A. Buchberger, C. Hager, H. D. Pfister, L. Schmalen, and A. Graell i Amat, "Learned decimation for neural belief propagation decoders: Invited paper," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2021, pp. 8273–8277.

[316] X. Wu, M. Jiang, and C. Zhao, "Decoding optimization for 5G LDPC codes by machine learning," *IEEE Access*, vol. 6, pp. 50179–50186, 2018.

[317] B. Vasić, X. Xiao, and S. Lin, "Learning to decode LDPC codes with finite-alphabet message passing," in *Proc. Inf. Theory Appl. Workshop (ITA)*, 2018, pp. 1–9.

[318] L. Wang, S. Chen, J. Nguyen, D. Dariush, and R. Wesel, "Neural-network-optimized degree-specific weights for LDPC MinSum decoding," in *Proc. 11th Int. Symp. Topics Coding (ISTC)*, 2021, pp. 1–5.

[319] J. Dai et al., "Learning to decode protograph LDPC codes," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 1983–1999, Jul. 2021.

[320] M. Sandell and A. Ismail, "Machine learning for LLR estimation in flash memory with LDPC codes," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 68, no. 2, pp. 792–796, Feb. 2021.

[321] M. Geiselhart et al., "Learning quantization in LDPC decoders," in *Proc. IEEE Globecom Workshops*, 2022, pp. 467–472.

[322] 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 J. Commun. Soc.*, vol. 1, pp. 646–660, 2020.

[323] A. Buchberger, C. Hager, H. D. Pfister, L. Schmalen, and A. Graell i Amat, "Pruning and quantizing neural belief propagation decoders," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 1957–1966, Jul. 2021.

[324] M. Davey and D. MacKay, "Low density parity check codes over GF(q)," in *Proc. Inf. Theory Workshop*, 2006, pp. 70–71.

[325] D. J. C. MacKay and M. C. Davey, "Evaluation of Gallager codes for short block length and high rate applications," in *Codes, Systems, and Graphical Models* (The IMA Volumes in Mathematics and its Applications). New York, NY, USA: Springer, 2001, pp. 113–130.

[326] L. Barnault and D. Declercq, "Fast decoding algorithm for LDPC over GF(2/sup q/)," in *Proc. IEEE Inf. Theory Workshop*, 2003, pp. 70–73.

[327] D. Declercq and M. Fossorier, "Decoding algorithms for nonbinary LDPC codes over GF(q)," *IEEE Trans. Commun.*, vol. 55, no. 4, pp. 633–643, Apr. 2007.

[328] V. Savin, "Min–max decoding for non binary LDPC codes," in *Proc. IEEE Int. Symp. Inf. Theory*, 2008, pp. 960–964.

[329] B. Liu, J. Gao, G. Dou, and W. Tao, "Weighted symbol-flipping decoding for nonbinary LDPC codes," in *Proc. 2nd Int. Conf. Netw. Security Wireless Commun. Trusted Comput.*, vol. 1, 2018, pp. 223–226.

[330] F. Garcia-Herrero, D. Declercq, and J. Valls, "Non-binary LDPC decoder based on symbol flipping with multiple votes," *IEEE Commun. Lett.*, vol. 18, no. 5, pp. 749–752, May 2014.

[331] S. Wang, Q. Huang, and Z. Wang, "Symbol flipping decoding algorithms based on prediction for non-binary LDPC codes," *IEEE Trans. Commun.*, vol. 65, no. 5, pp. 1913–1924, May 2017.

[332] Z. Zhao, X. Jiao, J. Mu, H. Han, and Y.-C. He, "Momentum-based symbol flipping decoding algorithms for non-binary LDPC codes," *IEEE Trans. Veh. Technol.*, vol. 72, no. 7, pp. 9579–9584, Jul. 2023.

[333] M.-R. Li, W.-X. Chu, H.-C. Lee, and Y.-L. Ueng, "An efficient high-rate non-binary LDPC decoder architecture with early termination," *IEEE Access*, vol. 7, pp. 20302–20315, 2019.

[334] V. B. Wijekoon, E. Viterbo, and Y. Hong, "Decoding of NB-LDPC codes over subfields," *IEEE Trans. Commun.*, vol. 69, no. 2, pp. 716–727, Feb. 2021.

[335] E. B. Yacoub and G. Liva, "Trapping and absorbing set enumerators for nonbinary protograph-based low-density parity-check code ensembles," *IEEE Trans. Commun.*, vol. 71, no. 4, pp. 1847–1862, Apr. 2023.

[336] T. V. Nguyen, A. Nosratinia, and D. Divsalar, "The design of rate-compatible protograph LDPC codes," *IEEE Trans. Commun.*, vol. 60, no. 10, pp. 2841–2850, Oct. 2012.

[337] D. G. M. Mitchell, M. Lentmaier, and D. J. Costello, "New families of LDPC block codes formed by terminating irregular protograph-based LDPC convolutional codes," in *Proc. IEEE Int. Symp. Inf. Theory*, 2010, pp. 824–828.

[338] S. Johnson and G. Lechner, "Spatially coupled repeat-accumulate codes," *IEEE Commun. Lett.*, vol. 17, no. 2, pp. 373–376, Feb. 2013.

[339] M. Stinner and P. M. Olmos, "On the waterfall performance of finite-length SC-LDPC codes constructed from protographs," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 2, pp. 345–361, Feb. 2016.

[340] M. Helmling et al., "Database of channel codes and ML simulation results," 2023. [Online]. Available: https://rptu.de/channel-codes/ml-simulation-results

[341] F. Kschischang and B. Frey, "Iterative decoding of compound codes by probability propagation in graphical models," *IEEE J. Sel. Areas Commun.*, vol. 16, no. 2, pp. 219–230, Feb. 1998.

[342] D. Hocevar, "A reduced complexity decoder architecture via layered decoding of LDPC codes," in *Proc. IEEE Workshop Signal Process. Syst. (SIPS)*, 2004, pp. 107–112.

[343] A. I. V. Casado, M. Griot, and R. D. Wesel, "Informed dynamic scheduling for belief-propagation decoding of LDPC codes," in *Proc. IEEE Int. Conf. Commun.*, 2007, pp. 932–937.

[344] A. Darabiha, A. Carusone, and F. Kschischang, "A bit-serial approximate min-sum LDPC decoder and FPGA implementation," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2006, pp. 149–152.

[345] P. Schläfer, N. Wehn, M. Alles, and T. Lehnigk-Emden, "A new dimension of parallelism in ultra high throughput LDPC decoding," in *Proc. IEEE Workshop Signal Process. Syst. (SIPS)*, 2013, pp. 153–158.

[346] Y. Chen and K. Parhi, "Overlapped message passing for quasi-cyclic low-density parity check codes," *IEEE Trans. Circuits Syst., I, Regular Papers*, vol. 51, no. 6, pp. 1106–1113, Jun. 2004.

[347] Z. Wang and Z. Cui, "A memory efficient partially parallel decoder architecture for quasi-cyclic LDPC codes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 15, no. 4, pp. 483–488, Apr. 2007.

[348] X. Chen, Q. Huang, S. Lin, and V. Akella, "FPGA-based low-complexity high-throughput tri-mode decoder for quasi-cyclic LDPC codes," in *Proc. 47th Annu. Allerton Conf. Commun. Control Comput. (Allerton)*, 2009, pp. 600–606.

[349] X. Chen, J. Kang, S. Lin, and V. Akella, "Memory system optimization for FPGA-based implementation of quasi-cyclic LDPC codes decoders," *IEEE Trans. Circuits Syst. I, Regular Papers*, vol. 58, no. 1, pp. 98–111, Jan. 2011.

[350] S. Nimara, O. Boncalo, A. Amaricai, and M. Popa, "FPGA architecture of multi-codeword LDPC decoder with efficient BRAM utilization," in *Proc. IEEE 19th Int. Symp. Design Diagn. Electron. Circuits Syst. (DDECS)*, 2016, pp. 1–4.

[351] V. E. Benea, "Optimal rearrangeable multistage connecting networks," *Bell Syst. Tech. J.*, vol. 43, no. 4, pp. 1641–1656, Jul. 1964.

[352] D. Oh and K. K. Parhi, "Low-complexity switch network for reconfigurable LDPC decoders," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 1, pp. 85–94, Jan. 2010.

[353] X. Chen, S. Lin, and V. Akella, "QSN—A simple circular-shift network for reconfigurable quasi-cyclic LDPC decoders," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 57, no. 10, pp. 782–786, Nov. 2020.

[354] S. Shao et al., "Survey of turbo, LDPC, and polar decoder ASIC implementations," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2309–2333, 3rd Quart., 2019.

[355] M. Li, V. Derudder, K. Bertrand, C. Desset, and A. Bourdoux, "High-speed LDPC decoders towards 1 Tb/s," *IEEE Trans. Circuits Syst. I, Regular Papers*, vol. 68, no. 5, pp. 2224–2233, May 2021.

[356] L. Lopacinski et al., "Ultra high-speed BP decoder for polar codes achieving 1.4 Tbps in 28 nm CMOS," in *Proc. Joint Eur. Conf. Netw. Commun. 6G Summit (EuCNC/6G Summit)*, Jun. 2022, pp. 434–439.

[357] L. Lopacinski et al., "A hardware optimized high throughput LDPC decoder supporting 3 Tb/s in 28 nm CMOS," in *Proc. IEEE 33rd Annu. Int. Symp. Pers. Indoor Mobile Radio Commun. (PIMRC)*, 2022, pp. 1326–1331.

[358] K. Cushon, P. Larsson-Edefors, and P. Andrekson, "Low-power 400-gbps soft-decision LDPC FEC for optical transport networks," *J. Lightw. Technol.*, vol. 34, no. 18, pp. 4304–4311, Sep. 15, 2016.

[359] R. Ghanaatian et al., "A 588-gb/s LDPC decoder based on finite-alphabet message passing," *IEEE Trans. Very Large Scale Integr. VLSI Syst.*, vol. 26, no. 2, pp. 329–340, Feb. 2018.

[360] S. Lee, S. Park, B. Jang, and I.-C. Park, "Multi-mode QC-LDPC decoding architecture with novel memory access scheduling for 5G new-radio standard," *IEEE Trans. Circuits Syst. I, Regular Papers*, vol. 69, no. 5, pp. 2035–2048, May 2022.

[361] "Enabling practical wireless Tb/s communications with next generation channel coding -EPIC project—Fact sheet—H2020."

[362] R. Peng and R.-R. Chen, "WLC45-2: Application of nonbinary LDPC codes for communication over fading channels using higher order modulations," in *Proc. IEEE Globecom*, 2006, pp. 1–5.

[363] X. Chen and C.-L. Wang, "High-throughput efficient non-binary LDPC decoder based on the simplified min-sum algorithm," *IEEE Trans. Circuits Syst. I, Regular Papers*, vol. 59, no. 11, pp. 2784–2794, Nov. 2012.

[364] D. Feng, H. Xu, J. Zheng, and B. Bai, "Nonbinary LDPC-coded spatial modulation," *IEEE Trans. Wireless Commun.*, vol. 17, no. 4, pp. 2786–2799, Apr. 2018.

[365] C.-X. Wang et al., "On the road to 6G: Visions, requirements, key technologies, and testbeds," *IEEE Commun. Surveys Tuts.*, vol. 25, no. 2, pp. 905–974, 2nd Quart., 2023.

[366] C. Condo, M. Martina, and G. Masera, "VLSI implementation of a multi-mode turbo/LDPC decoder architecture," *IEEE Trans. Circuits Syst. I, Regular Papers*, vol. 60, no. 6, pp. 1441–1454, Jun. 2013.

[367] S. Cao, T. Lin, S. Zhang, S. Xu, and C. Zhang, "A reconfigurable and pipelined architecture for standard-compatible LDPC and polar decoding," *IEEE Trans. Veh. Technol.*, vol. 70, no. 6, pp. 5431–5444, Jun. 2021.

[368] Y. Wang, Z. Zhang, S. Zhang, S. Cao, and S. Xu, "A unified deep learning based polar-LDPC decoder for 5G communication systems," in *Proc. 10th Int. Conf. Wireless Commun. Signal Process. (WCSP)*, 2018, pp. 1–6.

[369] E. Arı kan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. Inf. Theory*, vol. 55, no. 7, pp. 3051–3073, Jul. 2009.

[370] E. Arıkan, "On the origin of polar coding," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 2, pp. 209–223, Feb. 2016.

[371] J. Massey, "Capacity, cutoff rate, and coding for a direct-detection optical channel," *IEEE Trans. Commun.*, vol. C-29, no. 11, pp. 1615–1621, Nov. 1981.

[372] M. S. Pinsker, "On the complexity of decoding," *Problemy Peredachi Informatsii*, vol. 1, no. 1, pp. 113–116, 1965.

[373] H. Imai and S. Hirakawa, "A new multilevel coding method using error-correcting codes," *IEEE Trans. Inf. Theory*, vol. IT-23, no. 3, pp. 371–377, May 1977.

[374] N. Goela, S. B. Korada, and M. Gastpar, "On LP decoding of polar codes," in *Proc. IEEE Inf. Theory Workshop*, 2010, pp. 1–5.

[375] G. D. Forney, "Codes on graphs: Normal realizations," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 520–548, Feb. 2001.

[376] N. Hussami, S. B. Korada, and R. Urbanke, "Performance of polar codes for channel and source coding," in *Proc. IEEE Int. Symp. Inf. Theory*, 2009, pp. 1488–1492.

[377] M. Rowshan, S. H. Dau, and E. Viterbo, "Error coefficient-reduced polar/PAC codes," 2021, *arXiv:2111.08843*.

[378] M. Rowshan, S. H. Dau, and E. Viterbo, "On the formation of min-weight codewords of polar/PAC codes and its applications," *IEEE Trans. Inf. Theory*, vol. 69, no. 12, pp. 7627–7649, Dec. 2023.

[379] M. Bardet, J. Chaulet, V. Dragoi, A. Otmani, and J.-P. Tillich, "Cryptanalysis of the mCeliece public key cryptosystem based on polar codes," in *Proc. Post Quant. Cryptography 7th Int. Workshop (PQCrypto)*, 2016, pp. 118–143.

[380] M. Bardet, V. Dragoi, A. Otmani, and J.-P. Tillich, "Algebraic properties of polar codes from a new polynomial formalism," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2016, pp. 230–234.

[381] E. Arikan and E. Telatar, "On the rate of channel polarization," in *Proc. IEEE Int. Symp. Inf. Theory*, 2009, pp. 1493–1495.

[382] S. B. Korada, E. Şaşoğlu, and R. Urbanke, "Polar codes: Characterization of exponent, bounds, and constructions," *IEEE Trans. Inf. Theory*, vol. 56, no. 12, pp. 6253–6264, Dec. 2010.

[383] M.-K. Lee and K. Yang, "The exponent of a polarizing matrix constructed from the kronecker product," *Designs Codes Cryptography*, vol. 70, pp. 313–322, May 2014.

[384] E. Arikan, "A survey of reed-muller codes from polar coding perspective," in *Proc. IEEE Inf. Theory Workshop Inf. Theory (ITW)*, 2010, pp. 1–5.

[385] R. Mori and T. Tanaka, "Performance of polar codes with the construction using density evolution," *IEEE Commun. Lett.*, vol. 13, no. 7, pp. 519–521, Jul. 2009.

[386] T. Richardson and R. Urbanke, *Modern Coding Theory*. Cambridge, U.K.: Cambridge Univ. Press, 2008.

[387] C. Hartmann and L. Rudolph, "An optimum symbol-by-symbol decoding rule for linear codes," *IEEE Trans. Inf. Theory*, vol. IT-22, no. 5, pp. 514–517, Sep. 1976.

[388] G. Battail, M. Decouvelaere, and P. Godlewski, "Replication decoding," *IEEE Trans. Inf. Theory*, vol. IT-25, no. 3, pp. 332–345, May 1979.

[389] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. Inf. Theory*, vol. 42, no. 2, pp. 429–445, Mar. 1996.

[390] M. Luby, M. Mitzenmacher, A. Shokrollah, and D. Spielman, "Analysis of low density codes and improved designs using irregular graphs," in *Proc. 13th Annu. ACM Symp. Theory Comput.*, 1998, pp. 249–258.

[391] I. Tal and A. Vardy, "How to construct polar codes," *IEEE Trans. Inf. Theory*, vol. 59, no. 10, pp. 6562–6582, Oct. 2013.

[392] S. B. Korada, A. Montanari, E. Telatar, and R. Urbanke, "An empirical scaling law for polar codes," in *Proc. IEEE Int. Symp. Inf. Theory*, 2010, pp. 884–888.

[393] P. Trifonov, "Efficient design and decoding of polar codes," *IEEE Trans. Commun.*, vol. 60, no. 11, pp. 3221–3227, Nov. 2012.

[394] C. Schürch, "A partial order for the synthesized channels of a polar code," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2016, pp. 220–224.

[395] G. He et al., "Beta-expansion: A theoretical framework for fast and recursive construction of polar codes," in *Proc. GLOBECOM IEEE Global Commun. Conf.*, 2017, pp. 1–6.

[396] H. Huawei, *Polar Code Design and Rate Matching*, 3GPP, Sophia Antipolis, France, 2016.

[397] *5G; NR; Multiplexing and Channel Coding*, 3GPP, Sophia Antipolis, France, Jul. 2020.

[398] M. Rowshan, A. Burg, and E. Viterbo, "Polarization-adjusted convolutional (PAC) codes: Sequential decoding vs list decoding," *IEEE Trans. Veh. Technol.*, vol. 70, no. 2, pp. 1434–1447, Feb. 2021.

[399] X. Gu, M. Rowshan, and J. Yuan, "Rate-compatible shortened pac codes," in *Proc. IEEE/CIC Int. Conf. Commun. China (ICCC Workshops)*, 2023, pp. 1–6.

[400] B. Li, H. Shen, and D. Tse. "A RM-polar codes." Jul. 2014. [Online]. Available: https://arxiv.org/abs/1407.5483v1

[401] Z. Cai, L. Chen, W. Liu, and H. Zhang, "Modified PAC codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2023, pp. 1908–1913.

[402] M. Rowshan, S. H. Dau, and E. Viterbo, "Improving the error coefficient of polar codes," in *Proc. IEEE Inf. Theory Workshop (ITW)*, 2022, pp. 249–254.

[403] M. Rowshan and E. Viterbo, "How to modify polar codes for list decoding," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2019, pp. 1772–1776.

[404] M. Rowshan and E. Viterbo, "Stepped list decoding for polar codes," in *Proc. IEEE 10th Int. Symp. Turbo Codes Iterative Inf. Process. (ISTC)*, 2023, pp. 1–5.

[405] P. Trifonov and V. Miloslavskaya, "Polar codes with dynamic frozen symbols and their decoding by directed search," in *Proc. IEEE Inf. Theory Workshop (ITW)*, 2013, pp. 1–5.

[406] X. Wu, L. Yang, and J. Yuan, "Information coupled polar codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2018, pp. 861–865.

[407] X. Wu, L. Yang, Y. Xie, and J. Yuan, "Partially information coupled polar codes," *IEEE Access*, vol. 6, pp. 63689–63702, 2018.

[408] X. Wu, M. Qiu, and J. Yuan, "Partially information coupled bit-interleaved polar coded modulation," *IEEE Trans. Commun.*, vol. 69, no. 10, pp. 6409–6423, Oct. 2021.

[409] K.-H. Wang, W. Hou, S. Lu, P.-Y. Wu, Y.-L. Ueng, and J. Cheng, "Improving polar codes by spatial coupling," in *Proc. Int. Symp. Inf. Theory Appl. (ISITA)*, 2018, pp. 432–436.

[410] E. Arıkan, "From sequential decoding to channel polarization and back again," 2019, *arXiv:1908.09594*.

[411] M. Rowshan and E. Viterbo, "On convolutional precoding in PAC codes," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2021, pp. 1–6.

[412] M. Rowshan and J. Yuan, "Fast enumeration of minimum weight codewords of PAC codes," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Nov. 2022, pp. 255–260.

[413] M. Rowshan and J. Yuan, "On the minimum weight codewords of pac codes: The impact of pre-transformation," *IEEE J. Sel. Areas Inf. Theory*, vol. 4, no. 1, pp. 487–498, Feb. 2010.

[414] M. Rowshan, A. Burg, and E. Viterbo, "Complexity-efficient fano decoding of polarization-adjusted convolutional (PAC) codes," in *Proc. IEEE Int. Symp. Inf. Theory Appl. (ISITA)*, 2020, pp. 200–204.

[415] M. Rowshan, "Towards enhanced decoding of polar codes and PAC codes," Ph.D. dissertation, School Comput., Monash Univ., Melbourne, VIC, Australia, 2021.

[416] A. Mozammel, "Hardware implementation of FANO decoder for polarization-adjusted convolutional (PAC) codes," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 69, no. 3, pp. 1632–1636, Mar. 2022.

[417] H. Yao, A. Fazeli, and A. Vardy, "List decoding of Arıkan's PAC codes," *Entropy*, vol. 23, no. 7, p. 841, 2021.

[418] M. Rowshan and E. Viterbo, "List viterbi decoding of PAC codes," *IEEE Trans. Veh. Technol.*, vol. 70, no. 3, pp. 2428–2435, Mar. 2021.

[419] X. Zhang, M. Jiang, M. Zhu, K. Liu, and C. Zhao, "CRC-aided adaptive BP decoding of PAC codes," *Entropy*, vol. 24, no. 8, p. 1170, 2022.

[420] M. Rowshan and J. Yuan, "Constrained error pattern generation for grand," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2022, pp. 1767–1772.

[421] M. Rowshan and J. Yuan, "Segmented grand: Combining sub-patterns in near-ML order," 2023, *arXiv:2305.14892*.

[422] Y. Wang, Z. Shi, Z. Han, Z. Wei, and K. Li, "Improved ORB-grand for PAC codes," in *Proc. 8th Int. Conf. Commun. Image Signal Process. (CCISP)*, 2023, pp. 477–481.

[423] Z. Jiang, Z. Huang, Y. Zhang, and B. Zhou, "A soft-output stack decoding of polarization-adjusted convolutional codes," in *Proc. Int. Conf. Cryptography Netw. Security Commun. Technol. (CNSCT)*, vol. 12641, 2023, pp. 272–279.

[424] Y. Wu, Z. Huang, Y. Zhang, and S. Zhou, "Multi-stack decoding algorithm for polarization-adjusted convolutional codes," in *Proc. IEEE 3rd Int. Conf. Electron. Inf. Eng. Comput. Sci. (EIECS)*, 2023, pp. 1194–1197.

[425] L. Zhang, H. Liu, and Y. He, "Improved stack decoding for PAC codes," in *Proc. 32nd Wireless Opt. Commun. Conf. (WOCC)*, 2023, pp. 1–5.

[426] H. Saber, H. Hatami, and J. H. Bae, "Simplified successive cancellation list decoding of PAC codes," 2024, *arXiv:2401.13922*.

[427] J. Dai, H. Yin, Y. Lv, Y. Wang, and R. Lv, "Fast list decoding of PAC codes with new nodes," *IEEE Commun. Lett.*, vol. 28, no. 3, pp. 449–453, Mar. 2024.

[428] Q. Yu and Z. Shi, "Threshold-based list decoding for PAC codes," in *Proc. IEEE 21st Int. Conf. Commun. Technol. (ICCT)*, 2021, pp. 1296–1299.

[429] W. Zhang, "A novel SCL bit-flipping decoding of polarization-adjusted convolutional (PAC) codes," 2023, *arXiv:2307.05871*.

[430] X. Gu, M. Rowshan, and J. Yuan, "Selective reverse PAC coding for sphere decoding," 2022, *arXiv:2212.00254*.

[431] X. Gu, M. Rowshan, and J. Yuan, "Improved convolutional precoder for PAC codes," in *Proc. GLOBECOM IEEE Global Commun. Conf.*, 2023, pp. 1836–1841.

[432] B. Feng, Y. Yang, J. Jiao, and Q. Zhang, "On tail-biting polarization-adjusted convolutional (TB-PAC) codes and small-sizes list decoding," *IEEE Wireless Commun. Lett.*, vol. 27, no. 2, pp. 433–437, Feb. 2023.

[433] S. Gelincik, P. Mary, A. Savard, and J.-Y. Baudais, "A pre-transformation method to increase the minimum distance of polar-like codes," 2022, *arXiv:2202.04366*.

[434] A. Zunker et al., "Row-merged polar codes: Analysis, design and decoder implementation," 2023. [Online]. Available: https://arxiv.org/abs/2312.14749

[435] S. Ying, S. Jin, and Z. Zhang, "A spatially coupled PAC coding scheme and its list decoding," in *Proc. IEEE/CIC Int. Conf. Commun. China (ICCC Workshops)*, 2022, pp. 82–87.

[436] Q. Yu, Y. Zhang, R. Luo, L. Wang, and X. Li, "Parity-check polarization-adjusted convolutional coding," *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, vol. 107, no. 2, pp. 187–191, 2024.

[437] S. K. Mishra and K. Kim, "Selectively precoded polar codes," 2020, *arXiv:2011.04930*.

[438] G. Choi and N. Lee, "Deep polar codes," 2023, *arXiv:2308.03004*.

[439] A. Liu, B. Feng, C. Liang, J. Xu, and Q. Zhang, "A novel hamming check concatenated polarization-adjusted convolutional (PAC) codes," in *Proc. IEEE/CIC Int. Conf. Commun. China (ICCC)*, 2023, pp. 1–5.

[440] H. Wan, J. Cho, and C. J. Zhang, "Polar codes with enhanced weight distribution," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, 2023, pp. 1237–1242.

[441] M. Abdullah and W. H. Mow, "New search for the polarization-adjusted convolutional codes with respect to the AFER-optimality criterion," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2023, pp. 1723–1728.

[442] S. K. Mishra, D. Katyal, and S. A. Ganapathi, "A heuristic algorithm for rate-profiling of polarization adjusted convolutional (PAC) codes," 2023. [Online]. Available: https://www.techrxiv.org/users/682226/articles/677906-a-heuristic-algorithm-for-rate-profiling-of-polarization-adjusted-convolutional-pac-codes

[443] W. Liu, L. Chen, and X. Liu, "A weighted sum based construction of PAC codes," *IEEE Commun. Lett.*, vol. 27, no. 1, pp. 28–31, Jan. 2023.

[444] M.-C. Chiu and Y.-S. Su, "Design of polar codes and PAC codes for SCL decoding," *IEEE Trans. Commun.*, vol. 71, no. 5, pp. 2587–2601, May 2023.

[445] H. Sun, E. Viterbo, and R. Liu, "Optimized rate-profiling for PAC 8276 codes," 2021, *arXiv:2106.04074*.

[446] E. Arikan, "Systematic encoding and shortening of PAC codes," *Entropy*, vol. 22, no. 11, p. 1301, 2020. [Online]. Available: https://www.mdpi.com/1099-4300/22/11/1301

[447] X. Zhang, M. Jiang, M. Zhu, C. Zhao, and L. Hu, "Rate-compatible puncturing and shortening of short PAC codes for 6G URLLC," in *Proc. IEEE 24th Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, 2023, pp. 301–305.

[448] Z. Qiu and Y. He, "Construction of shortened systematic PAC codes based on Monte-Carlo algorithm," in *Proc. 32nd Wireless Opt. Commun. Conf. (WOCC)*, 2023, pp. 1–5.

[449] S. Jiang, J. Wang, C. Xia, and X. Li, "Construction of PAC codes with list-search and path-splitting critical sets," 2023, *arXiv:2304.11554*.

[450] Z. Sun, D. Lin, Y. Xiao, and M. Xiao, "An efficient construction of polarization-adjusted convolutional codes," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, 2023, pp. 1243–1248.

[451] H. Sun, E. Viterbo, and R. Liu, "Analysis of polarization-adjusted convolutional codes (PAC): A source-channel coding method," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, 2021, pp. 1–6.

[452] T. Kann, S. Kudekar, and M. Bloch, "Source polarization-adjusted convolutional codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2023, pp. 1896–1901.

[453] M. Zheng and C. Ling, "PAC codes for source and joint source-channel coding," 2023, *arXiv:2308.05472*.

[454] R. Mori and T. Tanaka, "Non-binary polar codes using reed-solomon codes and algebraic geometry codes," in *Proc. IEEE Inf. Theory Workshop*, 2010, pp. 1–5.

[455] S. E. Anderson and G. L. Matthews, "Exponents of polar codes using algebraic geometric code kernels," *Designs Codes Cryptography*, vol. 73, no. 1, pp. 699–717, 2014.

[456] A. Eid and I. Duursma, "Using concatenated algebraic geometry codes in channel polarization," 2013, *arXiv:1310.7159*.

[457] N. Presman, O. Shapira, S. Litsyn, T. Etzion, and A. Vardy, "Binary polarization kernels from code decompositions," *IEEE Trans. Inf. Theory*, vol. 61, no. 5, pp. 2227–2239, May 2015.

[458] H.-P. Lin, S. Lin, and K. A. Abdel-Ghaffar, "Linear and nonlinear binary kernels of polar codes of small dimensions with maximum exponents," *IEEE Trans. Inf. Theory*, vol. 61, no. 10, pp. 5253–5270, Oct. 2015.

[459] G. Trofimiuk and P. Trifonov, "Window processing of binary polarization kernels," *IEEE Trans. Commun.*, vol. 69, no. 7, pp. 4294–4305, Jul. 2021.

[460] P. Trifonov, "Binary successive cancellation decoding of polar codes with Reed–Solomon kernel," in *Proc. IEEE Int. Symp. Inf. Theory*, 2014, pp. 2972–2976.

[461] F. Abbasi and E. Viterbo, "Large kernel polar codes with efficient window decoding," *IEEE Trans. Veh. Technol.*, vol. 69, no. 11, pp. 14031–14036, Nov. 2020.

[462] N. Presman, O. Shapira, and S. Litsyn, "Mixed-kernels constructions of polar codes," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 2, pp. 239–253, Feb. 2016.

[463] V. Bioglio, F. Gabry, I. Land, and J.-C. Belfiore, "Multi-kernel polar codes: Concept and design principles," *IEEE Trans. Commun.*, vol. 68, no. 9, pp. 5350–5362, Sep. 2020.

[464] M. Benammar, V. Bioglio, F. Gabry, and I. Land, "Multi-kernel polar codes: Proof of polarization and error exponents," in *Proc. IEEE Inf. Theory Workshop (ITW)*, 2017, pp. 101–105.

[465] G. Schnabl and M. Bossert, "Soft-decision decoding of Reed–Muller codes as generalized multiple concatenated codes," *IEEE Trans. Inf. Theory*, vol. 41, no. 1, pp. 304–308, Jan. 1995.

[466] E. Arıkan, "Polar codes: A pipelined implementation," in *Proc. 4th ISBC*, 2010, pp. 11–14.

[467] I. Tal and A. Vardy, "List decoding of polar codes," *IEEE Trans. Inf. Theory*, vol. 61, no. 5, pp. 2213–2226, May 2015.

[468] I. Dumer and K. Shabunov, "Soft-decision decoding of Reed–Muller codes: Recursive lists," *IEEE Trans. Inf. Theory*, vol. 52, no. 3, pp. 1260–1266, Mar. 2006.

[469] R. Lucas, M. Bossert, and A. Dammann, "Improved soft-decision decoding of Reed–Muller codes as generalized multiple concatenated codes," in *Proc. ITG FACHBERICHT*, 1998, pp. 137–142.

[470] K. Niu and K. Chen, "CRC-aided decoding of polar codes," *IEEE Wireless Commun. Lett.*, vol. 16, no. 10, pp. 1668–1671, Jan. 2012.

[471] G. Sarkis, P. Giard, A. Vardy, C. Thibeault, and W. J. Gross, "Fast polar decoders: Algorithm and implementation," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 5, pp. 946–957, May 2014.

[472] M. Hanif and M. Ardakani, "Fast successive-cancellation decoding of polar codes: Identification and decoding of new nodes," *IEEE Commun. Lett.*, vol. 21, no. 11, pp. 2360–2363, Nov. 2017.

[473] Y. Ren et al., "A sequence repetition node-based successive cancellation list decoder for 5G polar codes: Algorithm and implementation," *IEEE Trans. Signal Process.*, vol. 70, pp. 5592–5607, 2022.

[474] S. A. Hashemi, C. Condo, and W. J. Gross, "List sphere decoding of polar codes," in *Proc. IEEE 49th Asilomar Conf. Signals Syst. Comput.*, 2015, pp. 1346–1350.

[475] A. Alamdar-Yazdi and F. R. Kschischang, "A simplified successive-cancellation decoder for polar codes," *IEEE Commun. Lett.*, vol. 15, no. 12, pp. 1378–1380, Dec. 2011.

[476] O. Afisiadis, A. Balatsoukas-Stimming, and A. Burg, "A low-complexity improved successive cancellation decoder for polar codes," in *Proc. 48th Asilomar Conf. Signals Syst. Comput.*, 2014, pp. 2116–2120.

[477] L. Chandesris, V. Savin, and D. Declercq, "Dynamic-SCflip decoding of polar codes," *IEEE Trans. Commun.*, vol. 66, no. 6, pp. 2333–2345, Jun. 2018.

[478] M. Rowshan and E. Viterbo, "Improved list decoding of polar codes by shifted-pruning," in *Proc. IEEE Inf. Theory Workshop (ITW)*, 2019, pp. 1–5.

[479] M. Rowshan and E. Viterbo, "Efficient partial rewind of successive cancellation-based decoders for polar codes," *IEEE Trans. Commun.*, vol. 70, no. 11, pp. 7160–7168, Nov. 2022.

[480] M. Rowshan and E. Viterbo, "SC list-flip decoding of polar codes by shifted pruning: A general approach," *Entropy*, vol. 24, no. 9, p. 1210, 2022.

[481] M. Rowshan and E. Viterbo, "Shifted pruning for path recovery in list decoding of polar codes," in *Proc. IEEE 11th Annu. Comput. Commun. Workshop Conf. (CCWC)*, 2021, pp. 1179–1184.

[482] F. Cheng, A. Liu, Y. Zhang, and J. Ren, "Bit-flip algorithm for successive cancellation list decoder of polar codes," *IEEE Access*, vol. 7, pp. 58346–58352, 2019.

[483] Y. Lv, H. Yin, and Y. Wang, "An adaptive ordered shifted-pruning list decoder for polar codes," *IEEE Access*, vol. 8, pp. 225181–225190, 2020.

[484] Y. Shen, A. Balatsoukas-Stimming, X. You, C. Zhang, and A. P. Burg, "Dynamic SCL decoder with path-flipping for 5G polar codes," *IEEE Wireless Commun. Lett.*, vol. 11, no. 2, pp. 391–395, Feb. 2022.

[485] F. Ivanov, V. Morishnik, and E. Krouk, "Improved generalized successive cancellation list flip decoder of polar codes with fast decoding of special nodes," *J. Commun. Netw.*, vol. 23, no. 6, pp. 417–432, 2021.

[486] H. Liu, J. Sha, and X. Wang, "An improved critical set for list decoding of polar codes," *IEEE Commun. Lett.*, vol. 27, no. 9, pp. 2269–2273, Sep. 2023.

[487] Y. Lv, H. Yin, Z. Yang, Y. Wang, J. Dai, and J. Huan, "On the performance of generalized SCL- flip decoding for polar codes," in *Proc. 2nd Asia Symp. Signal Process. (ASSP)*, 2021, pp. 148–152.

[488] W. Zhang and X. Wu, "Low-latency SCL bit-flipping decoding of polar codes," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2023, pp. 132–135.

[489] Y. Lv, H. Yin, Z. Yang, Y. Wang, and J. Dai, "Adaptive list flip decoder for polar codes with high-order error correction capability and a simplified flip metric," *Entropy*, vol. 24, no. 12, p. 1806, 2022. [Online]. Available: https://www.mdpi.com/1099-4300/24/12/1806

[490] Y. Lu, M. Zhao, M. Lei, C. Wang, and M. Zhao, "Deep learning aided SCL decoding of polar codes with shifted-pruning," *China Commun.*, vol. 20, no. 1, pp. 153–170, 2023.

[491] M. Rowshan, E. Viterbo, R. Micheloni, and A. Marelli, "Repetition-assisted decoding of polar codes," *Electron. Lett.*, vol. 55, no. 5, pp. 270–272, 2019.

[492] X. Wang, H. Zhang, J. Tong, J. Wang, J. Ma, and W. Tong, "Perturbation-enhanced SCL decoder for polar codes," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, 2023, pp. 1674–1679.

[493] M. Geiselhart, A. Elkelesh, M. Ebada, S. Cammerer, and S. ten Brink, "Automorphism ensemble decoding of Reed–Muller codes," *IEEE Trans. Commun.*, vol. 69, no. 10, pp. 6424–6438, May 2021.

[494] T. Hehn, O. Milenkovic, S. Laendner, and J. B. Huber, "Permutation decoding and the stopping redundancy hierarchy of cyclic and extended cyclic codes," *IEEE Trans. Inf. Theory*, vol. 54, no. 12, pp. 5308–5331, Dec. 2008.

[495] Y. Li et al., "The complete affine automorphism group of polar codes," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2021, pp. 1–6.

[496] C. Pillet, V. Bioglio, and I. Land, "Classification of automorphisms for the decoding of polar codes," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2022, pp. 110–115.

[497] U. U. Fayyaz and J. R. Barry, "Low-complexity soft-output decoding of polar codes," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 5, pp. 958–966, May 2014.

[498] J. Massey, "Variable-length codes and the FANO metric," *IEEE Trans. Inf. Theory*, vol. IT-18, no. 1, pp. 196–198, Jan. 1972.

[499] M.-O. Jeong and S.-N. Hong, "SC-FANO decoding of polar codes," *IEEE Access*, vol. 7, pp. 81682–81690, 2019.

[500] I. Timokhin and F. Ivanov, "On the improvements of successive cancellation creeper decoding for polar codes," *Digit. Signal Process.*, vol. 137, Jun. 2023, Art. no. 104008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1051200423001033

[501] F. Jelinek, "Fast sequential decoding algorithm using a stack," *IBM J. Res. Develop.*, vol. 13, no. 6, pp. 675–685, 1969.

[502] K. Niu and K. Chen, "Stack decoding of polar codes," *Electron. Lett.*, vol. 48, no. 12, pp. 695–697, 2012.

[503] M. Pohst, "On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications," *ACM SIGSAM Bull.*, vol. 15, no. 1, pp. 37–44, 1981.

[504] S. Kahraman and M. E. Çelebi, "Code based efficient maximum-likelihood decoding of short polar codes," in *Proc. IEEE Int. Symp. Inf. Theory*, 2012, pp. 1967–1971.

[505] D. Wu, Y. Li, X. Guo, and Y. Sun, "Ordered statistic decoding for short polar codes," *IEEE Commun. Lett.*, vol. 20, no. 6, pp. 1064–1067, Jun. 2016.

[506] K. R. Duffy, W. An, and M. Médard, "Ordered reliability bits guessing random additive noise decoding," *IEEE Trans. Signal Process.*, vol. 70, pp. 4528–4542, 2022.

[507] D.-M. Shin, S.-C. Lim, and K. Yang, "Design of length-compatible polar codes based on the reduction of polarizing matrices," *IEEE Trans. Commun.*, vol. 61, no. 7, pp. 2593–2599, Jul. 2013.

[508] V. Bioglio, F. Gabry, and I. Land, "Low-complexity puncturing and shortening of polar codes," in *Proc. IEEE Wireless Commun. Netw. Conf. Workshops (WCNCW)*, Mar. 2017, pp. 1–6.

[509] Z. B. K. Egilmez, L. Xiang, R. G. Maunder, and L. Hanzo, "The development, operation and performance of the 5G polar codes," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 1, pp. 96–122, 1st Quart., 2020.

[510] M. Seidl, A. Schenk, C. Stierstorfer, and J. B. Huber, "Polar-coded modulation," *IEEE Trans. Commun.*, vol. 61, no. 10, pp. 4108–4119, Oct. 2013.

[511] U. Wachsmann, R. F. Fischer, and J. B. Huber, "Multilevel codes: Theoretical concepts and practical design rules," *IEEE Trans. Inf. Theory*, vol. 45, no. 5, pp. 1361–1391, Jul. 1999.

[512] X. Wu, M. Qiu, and J. Yuan, "Delayed bit-interleaved polar coded modulation with superposition gray labeling," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, 2021, pp. 1–6.

[513] C. Leroux, A. J. Raymond, G. Sarkis, I. Tal, A. Vardy, and W. J. Gross, "Hardware implementation of successive-cancellation decoders for polar codes," *J. Signal Process. Syst.*, vol. 69, pp. 305–315, May 2012.

[514] C. Leroux, A. J. Raymond, G. Sarkis, and W. J. Gross, "A semi-parallel successive-cancellation decoder for polar codes," *IEEE Trans. Signal Process.*, vol. 61, no. 2, pp. 289–299, Jan. 2013.

[515] O. Dizdar and E. Arıkan, "A high-throughput energy-efficient implementation of successive cancellation decoder for polar codes using combinational logic," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 63, no. 3, pp. 436–447, Mar. 2016.

[516] B. Yuan and K. K. Parhi, "Low-latency successive-cancellation polar decoder architectures using 2-bit decoding," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 4, pp. 1241–1254, Apr. 2014.

[517] F. Ercan, C. Condo, and W. J. Gross, "Reduced-memory high-throughput fast-SSC polar code decoder architecture," in *Proc. IEEE Int. Workshop Signal Process. Syst. (SiPS)*, 2017, pp. 1–6.

[518] H. Rezaei, V. Ranasinghe, N. Rajatheva, M. Latva-Aho, G. Park, and O.-S. Park, "Implementation of ultra-fast polar decoders," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, 2022, pp. 235–241.

[519] H. Zheng, A. Balatsoukas-Stimming, Z. Cao, and T. Koonen, "Implementation of a high-throughput fast-SSC polar decoder with sequence repetition node," in *Proc. IEEE Workshop Signal Process. Syst. (SiPS)*, 2020, pp. 1–6.

[520] F. Ercan, T. Tonnellier, and W. J. Gross, "Energy-efficient hardware architectures for fast polar decoders," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 1, pp. 322–335, Jan. 2020.

[521] F. Ercan, T. Tonnellier, N. Doan, and W. J. Gross, "Practical dynamic SC-flip polar decoders: Algorithm and implementation," *IEEE Trans. Signal Process.*, vol. 68, pp. 5441–5456, 2020.

[522] J. Zeng, Y. Zhou, J. Lin, and Z. Wang, "Hardware implementation of improved fast-SSC-flip decoder for polar codes," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, 2019, pp. 580–585.

[523] G. Berhault, C. Leroux, C. Jego, and D. Dallet, "Hardware implementation of a soft cancellation decoder for polar codes," in *Proc. Conf. Design Architect. Signal Image Process. (DASIP)*, 2015, pp. 1–8.

[524] J. Lin, Z. Yan, and Z. Wang, "Efficient soft cancelation decoder architectures for polar codes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 1, pp. 87–99, Sep. 2016.

[525] L. Zhang, Y. Sun, Y. Shen, W. Song, X. You, and C. Zhang, "Efficient fast-scan flip decoder for polar codes," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2021, pp. 1–5.

[526] C. Zhang, X. You, and J. Sha, "Hardware architecture for list successive cancellation polar decoder," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2014, pp. 209–212.

[527] A. Balatsoukas-Stimming, M. B. Parizi, and A. Burg, "LLR-based successive cancellation list decoding of polar codes," *IEEE Trans. Signal Process.*, vol. 63, no. 19, pp. 5165–5179, Oct. 2015.

[528] X. Liu et al., "A 5.16Gbps decoder asic for polar code in 16nm FinFET," in *Proc. 15th Int. Symp. Wireless Commun. Syst. (ISWCS)*, 2018, pp. 1–5.

[529] A. Balatsoukas-Stimming, M. B. Parizi, and A. Burg, "On metric sorting for successive cancellation list decoding of polar codes," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2015, pp. 1993–1996.

[530] G. Sarkis, P. Giard, A. Vardy, C. Thibeault, and W. J. Gross, "Fast list decoders for polar codes," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 2, pp. 318–328, Feb. 2016.

[531] S. A. Hashemi, C. Condo, and W. J. Gross, "Fast and flexible successive-cancellation list decoders for polar codes," *IEEE Trans. Signal Process.*, vol. 65, no. 21, pp. 5756–5769, Nov. 2017.

[532] Y. Fan et al., "A low-latency list successive-cancellation decoding implementation for polar codes," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 2, pp. 303–317, Feb. 2016.

[533] J. Lin, C. Xiong, and Z. Yan, "A high throughput list decoder architecture for polar codes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 6, pp. 2378–2391, Jun. 2016.

[534] H.-Y. Lee, Y.-H. Pan, and Y.-L. Ueng, "A node-reliability based CRC-aided successive cancellation list polar decoder architecture combined with post-processing," *IEEE Trans. Signal Process.*, vol. 68, pp. 5954–5967, 2020. [Online]. Available: https://ieeexplore.ieee.org/xpl/tocresult.jsp?isnumber=8933520&punumber=78

[535] C. Kestel, M. Geiselhart, L. Johannsen, S. T. Brink, and N. Wehn, "Automorphism ensemble polar code decoders for 6G URLLC," in *Proc. 26th Int. ITG Workshop Smart Antennas 13th Conf. Syst. Commun. Coding (WSA & SCC)*, 2023, pp. 1–6.

[536] G. Coppolino, C. Condo, G. Masera, and W. J. Gross, "A multi-kernel multi-code polar decoder architecture," *IEEE Trans. Circuits Syst. I, Regular Papers*, vol. 65, no. 12, pp. 4413–4422, 2018.

[537] H. Rezaei, N. Rajatheva, and M. Latva-Aho, "Low-latency multi-kernel polar decoders," *IEEE Access*, vol. 10, pp. 119460–119474, 2022.

[538] H. Rezaei, N. Rajatheva, and M. Latva-Aho, "High-throughput rate-flexible combinational decoders for multi-kernel polar codes," 2023, *arXiv:2301.10445*.

[539] A. Pamuk, "An FPGA implementation architecture for decoding of polar codes," in *Proc. IEEE 8th Int. Symp. Wireless Commun. Syst.*, 2011, pp. 437–441.

[540] Y. Zhang, Q. Zhang, X. Pan, Z. Ye, and C. Gong, "A simplified belief propagation decoder for polar codes," in *Proc. IEEE Int. Wireless Symp. (IWS)*, 2014, pp. 1–4.

[541] B. Yuan and K. K. Parhi, "Architecture optimizations for BP polar decoders," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2013, pp. 2654–2658.

[542] Y.-T. Chen, W.-C. Sun, C.-C. Cheng, T.-L. Tsai, Y.-L. Ueng, and C.-H. Yang, "An integrated message-passing detector and decoder for polar-coded massive mu-mimo systems," *IEEE Trans. Circuits Syst. I, Regular Papers*, vol. 66, no. 3, pp. 1205–1218, Mar. 2019.

[543] B. Yuan and K. K. Parhi, "Architectures for polar BP decoders using folding," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2014, pp. 205–208.

[544] Y. S. Park, Y. Tao, S. Sun, and Z. Zhang, "A 4.68 gb/s belief propagation polar decoder with bit-splitting register file," in *Proc. Symp. VLSI Circuits Dig. Tech. Papers*, 2014, pp. 1–2.

[545] J. Sha, X. Liu, Z. Wang, and X. Zeng, "A memory efficient belief propagation decoder for polar codes," *China Commun.*, vol. 12, no. 5, pp. 34–41, 2015.

[546] X. Gu, M. Rowshan, and J. Yuan, "A non-uniform quantization-based hardware architecture for BP decoding of polar codes," *Electronics*, vol. 11, no. 1, p. 93, 2023.

[547] M. Rowshan, E. Viterbo, R. Micheloni, and A. Marelli, "Logarithmic non-uniform quantization for list decoding of polar codes," in *Proc. IEEE 11th Annu. Comput. Commun. Workshop Conf. (CCWC)*, 2021, pp. 1161–1166.

[548] S. M. Abbas, Y. Fan, J. Chen, and C.-Y. Tsui, "High-throughput and energy-efficient belief propagation polar code decoder," *IEEE Trans. Very Large Scale Integr. VLSI Syst.*, vol. 25, no. 3, pp. 1098–1111, Mar. 2017.

[549] H. Ji, Y. Shen, W. Song, Z. Zhang, X. You, and C. Zhang, "Hardware implementation for belief propagation flip decoding of polar codes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 68, no. 3, pp. 1330–1341, Mar. 2021.

[550] Y. Shen, W. Song, Y. Ren, H. Ji, X. You, and C. Zhang, "Enhanced belief propagation decoder for 5G polar codes with bit-flipping," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 67, no. 5, pp. 901–905, May 2020.

[551] Y. Ren et al., "High-throughput flexible belief propagation list decoder for polar codes," 2022, *arXiv:2210.13887*.

[552] P. Giard, G. Sarkis, C. Thibeault, and W. J. Gross, "237 gbit/s unrolled hardware polar decoder," *Electron. Lett.*, vol. 51, no. 10, pp. 762–763, 2015.

[553] A. Süral, E. G. Sezer, Y. Ertuğrul, O. Arikan, and E. Arikan, "Terabits-per-second throughput for polar codes," in *Proc. IEEE 30th Int. Symp. Pers. Indoor Mobile Radio Commun. (PIMRC Workshops)*, 2019, pp. 1–7.

[554] P. Giard et al., "A multi-gbps unrolled hardware list decoder for a systematic polar code," in *Proc. 50th Asilomar Conf. Signals Syst. Comput.*, 2016, pp. 1194–1198.

[555] P. Giard, C. Thibeault, and W. J. Gross, *High-Speed Decoders for Polar Codes*. Heidelberg, Germany: Springer, 2017.

[556] C. Kestel et al., "A 506Gbit/s polar successive cancellation list decoder with CRC," in *Proc. IEEE 31st Annu. Int. Symp. Pers. Indoor Mobile Radio Commun.*, 2020, pp. 1–7.

[557] J. Xiao, Y. Zhou, S. Song, and Z. Wang, "A low-latency and area-efficient orbgrand decoder for polar codes," in *Proc. 4th Inf. Commun. Technol. Conf. (ICTC)*, 2023, pp. 10–15.

[558] M. Rowshan, V.-F. Drăgoi, and J. Yuan, "On the closed-form weight enumeration of polar codes: 1.5d-weight codewords," 2023, *arXiv:2305.02921*.

[559] M. Rowshan, V.-F. Drăgoi, and J. Yuan, "Weight structure of low/high-rate polar codes and its applications," 2024, *arXiv:2402.12707*.

[560] *Enhanced Turbo Codes for NR: Performance Evaluation for EMBB and URLLC*, 3GPP, Sophia Antipolis, France, Oct. 2016.

[561] *NR Channel Coding BLER Database*, 3GPP, Sophia Antipolis, France, Oct. 2016.

[562] *Enhanced Turbo Codes for NR: Performance Evaluation for eMBB and URLLC*, 3GPP, Sophia Antipolis, France, Nov. 2016.

[563] T. Koike-Akino and Y. Wang, "Evolution of polar coding," in *Proc. IEEE Int. Symp. Topics Coding (ISTC)*, 2021, p. 5.

[564] T. Stockhammer, A. Shokrollahi, M. Watson, M. Luby, and T. Gasiba, "Application layer forward error correction for mobile multimedia broadcasting," in *Handbook of mobile broadcasting*. London, U.K.: Auerbach, 2008, pp. 239–278.

[565] D. Gomez-Barquero, D. Gozalvez, and N. Cardona, "Application layer FEC for mobile TV delivery in IP datacast over DVB-H systems," *IEEE Trans. Broadcast.*, vol. 55, no. 2, pp. 396–406, Jun. 2009.

[566] P. J. Marcelis, V. Rao, and R. V. Prasad, "DARE: Data recovery through application layer coding for LoRaWAN," in *Proc. 2nd Int. Conf. Internet Things Design Implement.*, 2017, pp. 97–108.

[567] M. Sandell and U. Raza, "Application layer coding for IoT: Benefits, limitations, and implementation aspects," *IEEE Syst. J.*, vol. 13, no. 1, pp. 554–561, Mar. 2019.

[568] D. Hou, K. Zhao, and W. Li, "Application layer channel coding for space DTN," in *Proc. Int. Conf. Mach. Learn. Intell. Commun.*, 2017, pp. 347–354.

[569] H. Niu, M. Iwai, K. Sezaki, L. Sun, and Q. Du, "Exploiting fountain codes for secure wireless delivery," *IEEE Commun. Lett.*, vol. 18, no. 5, pp. 777–780, May 2014.

[570] O. Etesami and A. Shokrollahi, "Raptor codes on binary memoryless symmetric channels," *IEEE Trans. Inf. Theory*, vol. 52, no. 5, pp. 2033–2051, May 2006.

[571] Z. Cheng, J. Castura, and Y. Mao, "On the design of raptor codes for binary-input Gaussian channels," *IEEE Trans. Commun.*, vol. 57, no. 11, pp. 3269–3277, Jun. 2009.

[572] S. Jayasooriya, M. Shirvanimoghaddam, L. Ong, and S. J. Johnson, "Analysis and design of raptor codes using a multi-edge framework," *IEEE Trans. Commun.*, vol. 65, no. 12, pp. 5123–5136, Dec. 2017.

[573] J. Castura and Y. Mao, "Rateless coding over fading channels," *IEEE Wireless Commun. Lett.*, vol. 10, no. 1, pp. 46–48, Jan. 2006.

[574] J. Castura and Y. Mao, "Rateless coding and relay networks," *IEEE Signal Process. Mag.*, vol. 24, no. 5, pp. 27–35, Jan. 2007.

[575] M. Shirvanimoghaddam, M. Dohler, and S. J. Johnson, "Massive multiple access based on superposition Raptor codes for cellular M2M communications," *IEEE Trans. Wireless Commun.*, vol. 16, no. 1, pp. 307–319, Jan. 2017.

[576] Y.-M. Chen, W.-M. Lai, and Y.-L. Ueng, "Rateless coded multiplexing for downlink transmission with two users: Performance analysis and system design," *IEEE Access*, vol. 7, pp. 50440–50452, 2019.

[577] J. Perry, H. Balakrishnan, and D. Shah, "Rateless spinal codes," in *Proc. 10th ACM Workshop Hot Topics Netw.*, 2011, pp. 1–6.

[578] H. Balakrishnan, P. Iannucci, J. Perry, and D. Shah, "De-randomizing Shannon: The design and analysis of a capacity-achieving rateless code," 2012, *arXiv:1206.0418*.

[579] J. Perry, P. A. Iannucci, K. E. Fleming, H. Balakrishnan, and D. Shah, "Spinal codes," *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 49–60, Aug. 2012.

[580] W. Yang, Y. Li, X. Yu, and J. Li, "A low complexity sequential decoding algorithm for rateless Spinal codes," *IEEE Commun. Lett.*, vol. 19, no. 7, pp. 1105–1108, Jul. 2015.

[581] R. Zamir, *Lattice Coding for Signals and Networks: A Structured Coding Approach to Quantization, Modulation, and Multiuser Information Theory*. Cambridge, U.K.: Cambridge Univ. Press, 2014.

[582] J. H. Conway, N. J. A. Sloane, and E. Bannai, *Sphere-Packings, Lattices, and Groups*. New York, NY, USA: Springer, 1999.

[583] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger, "Closest point search in lattices," *IEEE Trans. Inf. Theory*, vol. 48, no. 8, pp. 2201–2214, Aug. 2002.

[584] U. Erez and R. Zamir, "Achieving 1/2 log (1+SNR) on the AWGN channel with lattice encoding and decoding," *IEEE Trans. Inf. Theory*, vol. 50, no. 10, pp. 2293–2314, Oct. 2004.

[585] N. Sommer, M. Feder, and O. Shalvi, "Low-density lattice codes," *IEEE Trans. Inf. Theory*, vol. 54, no. 4, pp. 1561–1585, Apr. 2008.

[586] O. Shalvi, N. Sommer, and M. Feder, "Signal codes: Convolutional lattice codes," *IEEE Trans. Inf. Theory*, vol. 57, no. 8, pp. 5203–5226, Aug. 2011.

[587] M.-R. Sadeghi, A. Banihashemi, and D. Panario, "Low-density parity-check lattices: Construction and decoding analysis," *IEEE Trans. Inf. Theory*, vol. 52, no. 10, pp. 4481–4495, Oct. 2006.

[588] N. di Pietro and J. J. Boutros, "Leech constellations of construction—A lattices," *IEEE Trans. Commun.*, vol. 65, no. 11, pp. 4622–4631, Nov. 2017.

[589] M. Qiu, L. Yang, Y. Xie, and J. Yuan, "On the design of multi-dimensional irregular repeat-accumulate lattice codes," *IEEE Trans. Commun.*, vol. 66, no. 2, pp. 478–492, Feb. 2018.

[590] L. Liu, Y. Yan, C. Ling, and X. Wu, "Construction of capacity-achieving lattice codes: Polar lattices," *IEEE Trans. Commun.*, vol. 67, no. 2, pp. 915–928, Feb. 2019.

[591] J. Boutros, E. Viterbo, C. Rastello, and J. C. Belfiore, "Good lattice constellations for both Rayleigh fading and Gaussian channels," *IEEE Trans. Inf. Theory*, vol. 42, no. 2, pp. 502–518, Mar. 1996.

[592] F. Oggier and E. Viterbo, "Algebraic number theory and code design for Rayleigh fading channels," *Found. Trends Commun. Inf. Theory*, vol. 1, no. 3, pp. 333–416, Dec. 2004.

[593] F. Oggier, J.-C. Belfiore, and E. Viterbo, "Cyclic division algebras: A tool for space–time coding," *Found. Trends Commun. Inf. Theory*, vol. 4, no. 1, pp. 1–95, 2007, doi: 10.1561/0100000016.

[594] B. Nazer and M. Gastpar, "Compute-and-forward: Harnessing interference through structured codes," *IEEE Trans. Inf. Theory*, vol. 57, no. 10, pp. 6463–6486, Oct. 2011.

[595] Q. T. Sun, J. Yuan, T. Huang, and K. W. Shum, "Lattice network codes based on eisenstein integers," *IEEE Trans. Commun.*, vol. 61, no. 7, pp. 2713–2725, Jul. 2013.

[596] C. Feng, D. Silva, and F. R. Kschischang, "An algebraic approach to physical-layer network coding," *IEEE Trans. Inf. Theory*, vol. 59, no. 11, pp. 7576–7596, Nov. 2013.

[597] N. E. Tunali, Y. C. Huang, J. J. Boutros, and K. R. Narayanan, "Lattices over Eisenstein integers for compute-and-forward," *IEEE Trans. Inf. Theory*, vol. 61, no. 10, pp. 5306–5321, Oct. 2015.

[598] M. Qiu, Y.-C. Huang, S.-L. Shieh, and J. Yuan, "A lattice-partition framework of downlink non-orthogonal multiple access without SIC," *IEEE Trans. Commun.*, vol. 66, no. 6, pp. 2532–2546, Jun. 2018.

[599] M. Qiu, Y.-C. Huang, J. Yuan, and C.-L. Wang, "Lattice-partition-based downlink non-orthogonal multiple access without SIC for slow fading channels," *IEEE Trans. Commun.*, vol. 67, no. 2, pp. 1166–1181, Feb. 2019.

[600] M. Qiu, Y.-C. Huang, and J. Yuan, "Downlink non-orthogonal multiple access without SIC for block fading channels: An algebraic rotation approach," *IEEE Trans. Wireless Commun.*, vol. 18, no. 8, pp. 3903–3918, Aug. 2019.

[601] M. Qiu, Y.-C. Huang, and J. Yuan "Discrete signaling and treating interference as noise for the Gaussian interference channel," *IEEE Trans. Inf. Theory*, vol. 67, no. 11, pp. 7253–7284, Nov. 2021.

[602] A. Joseph and A. R. Barron, "Least squares superposition codes of moderate dictionary size are reliable at rates up to capacity," *IEEE Trans. Inf. Theory*, vol. 58, no. 5, pp. 2541–2557, May 2012.

[603] A. Joseph and A. R. Barron, "Fast sparse superposition codes have near exponential error probability for R<C," *IEEE Trans. Inf. Theory*, vol. 60, no. 2, pp. 919–942, Feb. 2014.

[604] C. Rush, K. Hsieh, and R. Venkataramanan, "Capacity-achieving spatially coupled sparse superposition codes with AMP decoding," *IEEE Trans. Inf. Theory*, vol. 67, no. 7, pp. 4446–4484, Jul. 2021.

[605] R. Venkataramanan, S. Tatikonda, and A. Barron, "Sparse regression codes," *Found. Trends Commun. Inf. Theory*, vol. 15, nos. 1–2, pp. 1–195, 2019. doi: 10.1561/0100000092.

[606] Y. Takeishi, M. Kawakita, and J. Takeuchi, "Least squares superposition codes with Bernoulli dictionary are still reliable at rates up to capacity," *IEEE Trans. Inf. Theory*, vol. 60, no. 5, pp. 2737–2750, May 2014.

[607] Y. Takeishi and J. Takeuchi, "An improved upper bound on block error probability of least squares superposition codes with unbiased Bernoulli dictionary," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2016, pp. 1168–1172.

[608] K. Hsieh and R. Venkataramanan, "Modulated sparse superposition codes for the complex AWGN channel," *IEEE Trans. Inf. Theory*, vol. 67, no. 7, pp. 4385–4404, Jul. 2021.

[609] A. Fengler, P. Jung, and G. Caire, "SPARCs for unsourced random access," *IEEE Trans. Inf. Theory*, vol. 67, no. 10, pp. 6894–6915, Jul. 2021.

[610] J. Cho and P. J. Winzer, "Probabilistic constellation shaping for optical fiber communications," *J. Lightw. Technol.*, vol. 37, no. 6, pp. 1590–1607, Mar. 15, 2019.

[611] M. F. Barsoum, C. Jones, and M. Fitz, "Constellation design via capacity maximization," in *Proc. IEEE Int. Symp. Inf. Theory*, 2007, pp. 1821–1825.

[612] G. Böcherer, "Probabilistic amplitude shaping," *Found. Trends Commun. Inf. Theory*, vol. 20, no. 4, pp. 390–511, 2023, doi: 10.1561/0100000111.

[613] R. V. Nee and R. Prasad, *OFDM for Wireless Multimedia Communications*. London, U.K.: Artech House, 2000.

[614] P. Siohan, C. Siclet, and N. Lacaille, "Analysis and design of OFDM/OQAM systems based on filterbank theory," *IEEE Trans. Signal Process.*, vol. 50, no. 5, pp. 1170–1183, May 2002.

[615] R. Hadani et al., "Orthogonal time frequency space modulation," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, 2017, pp. 1–6.

[616] H. Lin and J. Yuan, "Orthogonal delay-doppler division multiplexing modulation," *IEEE Trans. Wireless Commun.*, vol. 21, no. 12, pp. 11024–11037, Dec. 2022.

[617] H. Lin, J. Yuan, W. Yu, J. Wu, and L. Hanzo, "Multi-carrier modulation: An evolution from time-frequency domain to delay-Doppler domain," 2023, *arXiv:2308.01802*.

[618] M. Giordani and M. Zorzi, "Non-terrestrial networks in the 6G era: Challenges and opportunities," *IEEE Netw.*, vol. 35, no. 2, pp. 244–251, Mar./Apr. 2020.

[619] H.-B. Jeon et al., "Free-space optical communications for 6G wireless networks: Challenges, opportunities, and prototype validation," *IEEE Commun. Mag.*, vol. 61, no. 4, pp. 116–121, Apr. 2023.

[620] H. G. Sandalidis, "Coded free-space optical links over strong turbulence and misalignment fading channels," *IEEE Trans. Commun.*, vol. 59, no. 3, pp. 669–674, Mar. 2011.

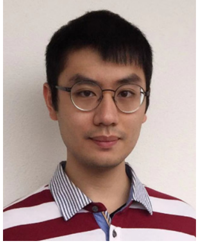
[621] M. Chochol, A. Rissons, J. Lacan, N. Vedrenne, and G. Artaud, "Evaluation of error correcting code performances of a free space optical communication system between LEO satellite and ground station," in *Proc. SPIE*, vol. 9647, 2015, pp. 93–101.

[622] A. Dixit et al., "Analytical determination of thresholds of LDPC codes in free space optical channel," *IEEE Open J. Commun. Soc.*, vol. 2, pp. 35–47, 2020.

[623] A. A. Hussein, A. Oka, T. T. Nguyen, and L. Lampe, "Rateless coding for hybrid free-space optical and radio-frequency communication," *IEEE Trans. Wireless Commun.*, vol. 9, no. 3, pp. 907–913, Mar. 2010.

[624] S. Kumar et al., "Impact of reed solomon forward error correction code in enhancing performance of free space optical communication link," in *Proc. Laser Commun. Propag. Atmosphere Oceans IX*, vol. 11506, 2020, pp. 25–32.

MOHAMMAD ROWSHAN (Member, IEEE) received the B.Eng. degree (Hons.) in electrical engineering from the University of Nottingham in 2015 (ranked 1), the M.Sc. degree in electrical engineering from The Hong Kong University of Science and Technology in 2016, and the Ph.D. degree in electrical engineering from Monash University in 2021. He is currently an Engineering ECA Fellow (Associate Lecturer) with the School of Electrical Engineering and Telecommunications, The University of New South Wales, Sydney, Australia. His research interests include channel coding, signal processing for communication systems, and hardware architecture design.



MIN QIU (Member, IEEE) received the Ph.D. degree in electrical engineering from The University of New South Wales, Sydney, Australia, in 2019, where he is a Postdoctoral Research Associate. His research interests include channel coding, information theory, and their applications for building reliable communication systems. He received the Australian Government Research Training Program Scholarship for the duration of his Ph.D. degree, the Australia Awards-Endeavour Research Fellowship in 2018, and the Chinese Government Award for Outstanding Self-Financed Students Abroad in 2019. He was honored as an Exemplary Reviewer of the IEEE TRANSACTIONS ON COMMUNICATIONS in 2018, 2019, 2021, and 2022, and the IEEE COMMUNICATIONS LETTERS from 2021 to 2023.

YIXUAN XIE (Member, IEEE) received the Ph.D. degree in electrical engineering from the University of New South Wales (UNSW), Sydney, in 2016. Since then, he has been a pivotal figure, focusing on FPGA implementation of channel codecs and prototyping communication schemes on software-defined radio platforms with the Wireless Communication Laboratory, UNSW Sydney, where he is currently with the School of Electrical Engineering and Telecommunications. His research focuses on error control coding, iterative detection/decoding methods, digital communication systems and signal processing, and multiple access schemes.

XINYI GU (Graduate Student Member, IEEE) received the B.E. and M.Phil. degrees in electrical engineering from the University of New South Wales, Australia, in 2020 and 2023, respectively, where she is currently pursuing the Ph.D. degree with the School of Electrical Engineering and Telecommunications. Her research interests include channel coding and hardware architecture design for decoders.

JINHONG YUAN (Fellow, IEEE) received the B.E. and Ph.D. degrees in electronics engineering from the Beijing Institute of Technology, Beijing, China, in 1991 and 1997, respectively. From 1997 to 1999, he was a Research Fellow with the School of Electrical Engineering, University of Sydney, Sydney, Australia. In 2000, he joined the School of Electrical Engineering and Telecommunications, University of New South Wales, Sydney, Australia, where he is currently a Professor and the acting Head of School and the Head of Telecommunication Group with the School. He has published two books, five book chapters, over 300 papers in telecommunications journals and conference proceedings, and 50 industrial reports. He is a co-inventor of one patent on MIMO systems and four patents on low-density-parity-check codes. His current research interests include error control coding and information theory, communication theory, and wireless and underwater communications. He has coauthored four Best Paper Awards and one Best Poster Award, including the Best Paper Award from the IEEE International Conference on Communications, Kansas City, USA, in 2018, the Best Paper Award from IEEE Wireless Communications and Networking Conference, Cancun, Mexico, in 2011, and the Best Paper Award from the IEEE International Symposium on Wireless Communications Systems, Trondheim, Norway, in 2007. He is currently serving as an Associate Editor for the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS and IEEE TRANSACTIONS ON COMMUNICATIONS. He served as the IEEE NSW Chapter Chair of Joint Communications/Signal Processions/Ocean Engineering Chapter from 2011 to 2014, and served as an Associate Editor for the IEEE TRANSACTIONS ON COMMUNICATIONS from 2012 to 2017.