Digital Communications and Networks 8 (2022) 359–372

Contents lists available at ScienceDirect

Digital Communications and Networks

journal homepage: www.keaipublishing.com/dcan

Research on encoding and decoding of non-binary polar codes over $GF(2^m)$

Shufeng Li [a,*], Mingyu Cai [a], Robert Edwards [b], Yao Sun [c], Libiao Jin [a]

[a] The State Key Laboratory of Media Convergence and Communication, School of Information and Engineering, Communication University of China, Beijing, 100024, China
[b] The Wolfson School of Mechanical Electrical and Manufacturing Engineering, Loughborough University, Leicestershire, LE11 3TE, UK
[c] James Watt School of Engineering, University of Glasgow, G12 8QQ, Scotland, UK

A R T I C L E I N F O

Keywords:
Non-binary polar code
Log-likelihood ratio
Successive-cancellation list
Perfect polarization based-SCL
Decoding complexity

A B S T R A C T

Binary Polar Codes (BPCs) have advantages of high-efficiency and capacity-achieving but suffer from large latency due to the Successive-Cancellation List (SCL) decoding. Non-Binary Polar Codes (NBPCs) have been investigated to obtain the performance gains and reduce latency under the implementation of parallel architectures for multi-bit decoding. However, most of the existing works only focus on the Reed-Solomon matrix-based NBPCs and the probability domain-based non-binary polar decoding, which lack flexible structure and have a large computation amount in the decoding process, while little attention has been paid to general non-binary kernel-based NBPCs and Log-Likelihood Ratio (LLR) based decoding methods. In this paper, we consider a scheme of NBPCs with a general structure over $GF(2^m)$. Specifically, we pursue a detailed Monte-Carlo simulation implementation to determine the construction for proposed NBPCs. For non-binary polar decoding, an SCL decoding based on LLRs is proposed for NBPCs, which can be implemented with non-binary kernels of arbitrary size. Moreover, we propose a Perfect Polarization-Based SCL (PPB-SCL) algorithm based on LLRs to reduce decoding complexity by deriving a new update function of path metric for NBPCs and eliminating the path splitting process at perfect polarized (i.e., highly reliable) positions. Simulation results show that the bit error rate of the proposed NBPCs significantly outperforms that of BPCs. In addition, the proposed PPB-SCL decoding obtains about a 40% complexity reduction of SCL decoding for NBPCs.

1. Introduction

Polar codes originally proposed by E.Arikan can provably achieve the symmetric capacity of binary-input discrete memoryless channels [1]. Due to the high data rates and spectrum efficiency [2–4], polar codes have been adopted by 5G new radio technology. However, considering the suboptimality and seriality of Successive Cancellation (SC) decoding [5,6], the BER performance of Binary Polar Codes (BPCs) is inferior to that of Low-Density Parity-Check (LDPC) codes, especially for long code length.

For further improvement, SC-List (SCL) decoding proposed by Tal and Vardy has shown the comparable performance of maximum-likelihood decoding with the increase of list size [7], which has been widely used in the hardware implementation of polar codes [8,9]. Particularly, the Bit Error Rate (BER) performance of polar codes can be further improved by Cyclic Redundancy Check (CRC) aided SCL decoding [10], which makes a better BER performance than that of LDPC codes [11]. The future wireless communication system has an urgent call for low latency [12–14]. However, updating and sorting Path Metrics (PMs) of the SCL decoding is complicated for parallel implementation leading to a high latency especially when the list size is large. As a development, the parallel output-based SCL decoding with the multi-bit decision has shown a considerable latency gain [15–17] and an excellent performance for practical systems [18].

As a promising technology, Non-Binary Polar Codes (NBPCs) were studied to improve the latency and BER performance of BPCs. It was proved that when the input alphabet size q is a prime number, a similar construction of binary counterparts leads to polarization [19]. Especially, sufficient conditions for channel polarization on kernel were derived in Ref. [20]. In Refs. [21,22], channel polarization provably holds when q is not a prime number, which shows that polar codes could be transmitted reliably with q-ary inputs [23]. In Refs. [24–28], several constructions of NBPCs were proposed. To be more specific, the performance of Reed-Solomon (RS) matrix based NBPCs outperform the BPCs over

* Corresponding author.
E-mail addresses: lishufeng@cuc.edu.cn (S. Li), cccmy1c@cuc.edu.cn (M. Cai), r.m.edwards@lboro.ac.uk (R. Edwards), Yao.Sun@glasgow.ac.uk (Y. Sun), libiao@cuc.edu.cn (L. Jin).

https://doi.org/10.1016/j.dcan.2022.01.005
Received 21 December 2020; Received in revised form 19 January 2022; Accepted 26 January 2022
Available online 5 February 2022

Binary Input Additive White Gaussian Noise (BI-AWGN) channels [24, 25]. Importantly, multiple bits of NBPCs are decoded together as one symbol due to the nature of SC decoding, which leads to a latency gain [26]. In Ref. [27], a probability domain-based decoding is proposed for NBPCs with the non-binary kernel over $GF(q)$. Moreover, NBPCs for high-order modulation schemes were designed to further enhance the performance contrast to BPCs [28].

While most of the existing works focus on the RS matrix-based NBPCs and probability domain-based non-binary polar decoding, there are still some challenges that need to be well addressed before fully enjoying the advantages of NBPCs. First, the encoding and decoding structures of RS-based NBPCs lack flexibility and universality since the size of the non-binary kernel increases with the field size. Second, the probability domain-based decoding leads to a high computation and memory in the decoding process [29], which is not friendly to hardware implementation. Third, the trade-off between the decoding complexity and BER performance of NBPCs needs to find an optimal balance for applications [30,31].

To address the above challenges, in this paper, we design an NBPC with a general structure over $GF(2^m)$, where the popular Log-Likelihood Ratio (LLR) based SCL decoding can be employed flexibly with no constraint on the field size. The set of frozen positions is found by the Monte-Carlo simulation. As a further development, the LLR-based SCL decoding algorithm is presented by using a generalized LLR recursive function for NBPCs. Moreover, a Perfect Polarization Based-SCL (PPB-SCL) algorithm based on LLRs is proposed to reduce the complexity of original non-binary SCL decoding. Simulation results show that the proposed NBPCs with PPB-SCL decoding obtain a 0.3 dB–0.5 dB gain compared to BPCs, where the additional complexity cost is reduced from 120% to 35%. The proposed scheme provides an appropriate trade-off between computational complexity and reliability.

Specifically, the main contributions of this paper are summarized as follows:

1) A CRC-aided NBPC based on general non-binary kernels over $GF(2^m)$ is proposed, which is not limited by the structure of RS kernels and thus can be implemented flexibly. Specifically, based on the binary kernel and the polarization conditions, we present the generalized non-binary polarization kernel, which can be easily extended to a large scale. For the channel polarization of NBPCs, the Monte-Carlo simulation method with Genie-aided SC decoding algorithm is considered to select the frozen symbols and information symbols.
2) For the universality of the NBPC decoding structure, the LLR-based SCL decoding algorithm for proposed NBPCs is considered. According to the partial distances of the kernel and the probability domain-based decoding of NBPCs, the LLR recursive function of proposed NBPCs is generally presented for the arbitrary kernel size. Particularly, we directly give the LLR for the common 2×2 kernel by employing the general functions. Furthermore, the LLR-based PM for proposed NBPCs is introduced by analyzing the penalty value of PMs for BPCs.
3) To trade-off between the decoding complexity and BER performance, PPB-SCL decoding is proposed for NBPCs. There are two main improvements over original non-binary SCL decoding. First, an optimized update function of PMs is derived to reduce the computational complexity, where only one class of LLR calculation is required just like BPCs. Then, inspired by the hybrid decoder, the redundant split paths are pruned in the decoding process if the current channel position shows high reliability in the Monte-Carlo simulation.

The rest of this paper is organized as follows. In Section 2, the system model of the proposed NBPCs is given. Section 3 and Section 4 describe the encoding and decoding schemes respectively. In Section 5, the performance of NBPCs we constructed is evaluated by numerical simulations. Finally, the conclusions and future research ideas are drawn in Section 6.

Notations: Let u_1^N and u_i^j denote row vectors $(u_1, \ldots, u_N)$ of length N and its sub-vectors $(u_i,\ldots,u_j)$, $1 \le i \le j \le N$. Let $u_{1,o}^N$ and $u_{1,e}^N$ denote odd and even index vectors of u_1^N, respectively. The estimate of u_1^N is defined as $\widehat{u}_1^N$. Let $\mathcal{A}^c$ and $|\mathcal{A}|$ denote the complement set and the cardinality of $\mathcal{A}$, respectively. $\mathcal{A} \setminus \mathcal{B}$ represents the subtraction between $\mathcal{A}$ and $\mathcal{B}$. G_T represents a kernel matrix with size T, where $G_{T(r)}$ is the r-th column vector of G_T. Let $G_{i,j}$, $i,j \in \{1, \ldots, T\}$ correspond to the element at position (i,j) in G_T. Let $\otimes$ refer to the Kronecker product of a matrix with the recursive formula $F^{\otimes n} = F \otimes F^{\otimes(n-1)}, n \ge 1$. In this paper, we consider the Galois field $GF(q)$, $q = 2^m$, i.e., the extension field of $GF(2)$. Binary and decimal numbers are employed to map elements over Galois field. Each element can be represented by binary m-tuples or decimal numbers between 0 and $q-1$.

2. System model

Similar to that in Ref. [27], the system we established is shown in Fig. 1, where frozen symbols are added via Monte Carlo simulation after the bit-to-symbol converter while frozen bits are added after the addition of CRC bits in Ref. [27]. Besides, the soft information converter is removed since we employ the polar decoder based on LLRs instead of probability domain.

For NBPCs, N_s and K_s denote code length and information code length in symbols respectively, then N_b and K_b denote them in bits respectively. Obviously, we have $N_b = mN_s$.

First, we transform the element of $GF(2^m)$ in the symbol vector $h_1^{K_s - l_{CRC}/m}$ into a binary m-tuple to obtain the binary vector with a length of $K_b - l_{CRC}$, where l_{CRC} denotes the CRC bit length. After adding l_{CRC} CRC bits, we get the vector $h_1^{K_s}$ by bit-to-symbol conversion. Then, according to frozen symbol position set from the Monte-Carlo simulation, we add frozen symbols to $h_1^{K_s}$ and send it to the encoder. All frozen symbols are set to 0. Finally, the output encoding vector $x_1^{N_s}$ is obtained. Since we use l_{CRC} CRC bits in NBPCs, the size $|\mathcal{A}|$ of information symbol position set $\mathcal{A}$ is fixed as K_s in order to keep the code rate unchanged. The last l_{CRC}/m positions in $\mathcal{A}$ will transmit CRC bits of original information bits. Note that CRC bits are also regarded as information bits. As a result, the code rate is $R = K_s/N_s = K_b/N_b$, and the effective information rate is $(K_b - l_{CRC})/N_b$. The calculation and Monte-Carlo simulation of the encoder will be described in Section 3.

In the transmission stage, we choose the BI-AWGN channels. Thus we need to convert the encoding vector $x_1^{N_s}$ into a binary vector $x_1^{N_b}$. After BPSK modulation, the obtained vector $w_1^{N_b}$ will be sent into the BI-AWGN channels. The channel model can be described as $V_j = W_j + Z_j (j = 1,\ldots, N_b)$ with the random variables of input and output, where $Z_1, \ldots, Z_{N_b}$ are N_b independent and identically distributed Gaussian random variables with a mean of 0 and a variance of σ^2. The Gaussian noise variance σ^2 is calculated from the Signal-to-Noise Ratio (SNR), namely

$$\sigma^2 = \frac{1}{2 \times 10^{\frac{\mathrm{SNR}}{10}} \times R} \tag{1}$$

Finally, the vector $v_1^{N_b}$ output from the BI-AWGN channels is obtained.

In the decoding stage, the initial LLR value used for the recursive operation in the decoder will first be derived from $v_1^{N_b}$. The estimated sequence $\widehat{u}_1^{N_s}$ without operation of the encoder is obtained by the decoder. Before entering the CRC check, we should remove the frozen symbol (the element 0 in all frozen positions) in $\widehat{u}_1^{N_s}$ and convert it into a binary vector $\widehat{h}_1^{K_b}$. The L decoding paths from the decoder perform the above operations sequentially in the ascending order of the PMs. Finally, the vector $\widehat{h}_1^{K_b - l_{CRC}}$ first passing CRC will be transformed into the estimated information symbol vector $\widehat{h}_1^{K_s - l_{CRC}/m}$ by bit-to-symbol conversion. BER performance is analyzed by comparing $h_1^{K_s - l_{CRC}/m}$ and $\widehat{h}_1^{K_s - l_{CRC}/m}$.

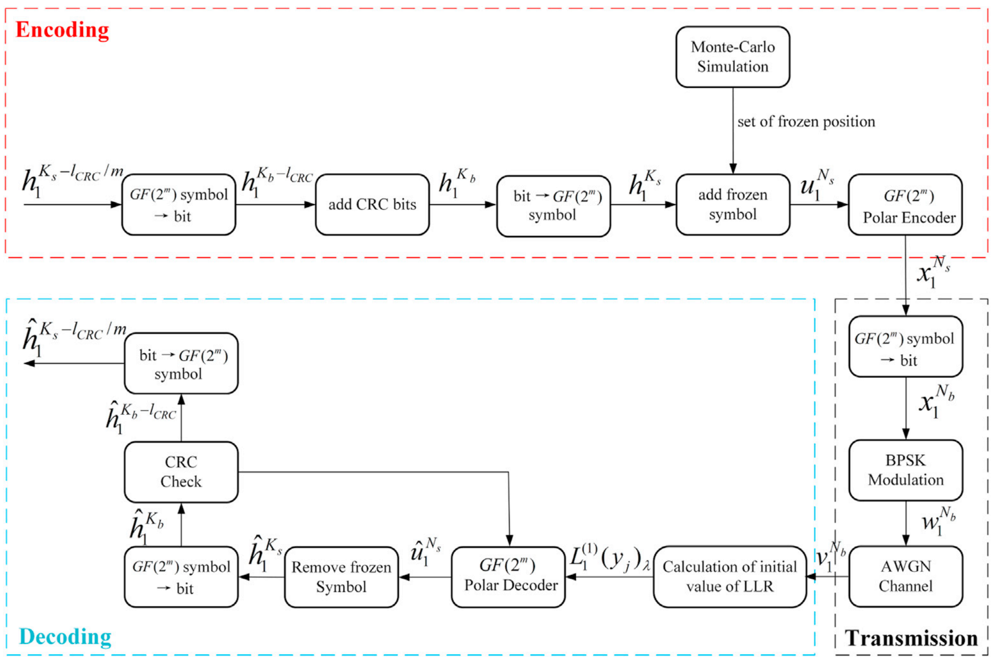


Fig. 1. System model of NBPCs over BI-AWGN channels.

The calculation of the initial LLR value and the operation of the decoder will be further explained in Section 4.

3. Encoding of Non-Binary Polar Codes

3.1. Galois field elements

Let α be the primitive element of Galois field $GF(q)$ with $q=p^m$, where p is a prime number and m is a positive integer. Then all the elements over $GF(q)$ can be expressed as $\alpha^{-\infty}=0$, $\alpha^0=1$, α, α^2, …, α^{q-2} with the power of α. These q elements can be generated by primitive polynomial $f(\alpha) = a_0+a_1\alpha+\cdots+a_{m-1}\alpha^{m-1}+a_m\alpha^m$, where α, α^2, …, $\alpha^m \in GF(p)$. Thus, the addition operation between elements can be expressed as the coefficient sum of polynomials, while the multiplication operation can be transformed into power addition. In this paper, we consider the NBPC based on the finite field of characteristic 2, i.e., $GF(2^m)$.

Similar to non-binary LDPC codes, the multiple bits of NBPCs are mapped into a symbol through the Galois field elements, which improves the resistance to burst errors. Importantly, latency is also reduced since the multiple bits are decoded jointly as a symbol.

Suppose a q-ary element α^i can be transformed into an m-bit binary sequence $y_1^m=\{y_1,\dots y_m\}$ by mapping function $\xi(y_1,\dots,y_m) = \sum_{k=1}^m y_k\alpha^{k-1}$. Taking $GF(4)$ as an example, the mapping relationship among field elements, bits and symbols is shown in Table 1. Note that it is common to convert the mapping between decimal and binary to bits and symbols for convenience.

According to the mapping relationship of Table 1 and operations of the Galois field, we get the addition and multiplication rules of symbols in Table 2 and Table 3, where $p(x)=x^2+x+1$ is adopted as the primitive polynomial.

Table 1
The mapping among $GF(4)$ elements, bits and symbols.

Element	Bit (Binary m-tuples)	Symbol (Decimal)
0	$\{0,0\}$	0
1	$\{0,1\}$	1
α	$\{1,0\}$	2
α^2	$\{1,1\}$	3

Table 2
Symbol addition over $GF(4)$.

Addition	**0**	**1**	**2**	**3**
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

Table 3
Symbol multiplication over $GF(4)$.

Multiplication	**0**	**1**	**2**	**3**
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

In general, given the m of $GF(2^m)$, we first formulate the mapping relationship. Then, the addition and multiplication rules of symbols are identified according to the operation over $GF(2^m)$. In this way, all operations involving vectors and matrices over $GF(2^m)$ can be realized by looking up tables, which avoids round-trip mapping for each operation.

3.2. Kernel structure and $GF(2^m)$ encoder

The traditional Arikan's kernel $G_2=\begin{bmatrix}1 & 0\\ 1 & 1\end{bmatrix}$ for BPCs can be regarded as the operation over $GF(2)$. For NBPCs, we generalize the concept of Arikan's kernel to $GF(q)$. In order to give a q-ary kernel that satisfies the polarization condition, we have the following definition.

Definition 1. *For matrix G_T, $T\geq 2$, ε_k, $k\in\{1,\dots,T\}$ represents the number of non-zero elements in the k-th row of G_T. Let $t=\underset{k\in\{1,\dots,T\}}{\arg\max}\,\varepsilon_k$, and then we call $G_{t,t}$ the tag -element of the kernel G_T.*

Obviously, the tag-element must be located on the diagonal of the matrix. In Ref. [20], it is set to 1. In this paper, let $G_{t,t}\in GF(q)$ and $G_{t,t}\neq 0$ for the generality of the definition. Next, we use the notation for the q-ary channel polarization theory in Ref. [20] and the tag-element in Definition 1 to illustrate the kernel's selection conditions.

Corollary 1. *Let $G_{t,t}$ be the tag-element of kernel G_T, $T \geq 2$. If there is a primitive element $G_{t,j}$ for $\forall j \in \{1, \ldots, T-1\}$, then $P(I_\infty \in \{0, 1\}) = 1$.*

Corollary 1 shows that at least one primitive element in the row where the kernel's tag-element is located can satisfy the channel polarization theory. The specific proof process can be seen in Ref. [20].

In order to distinguish it from Arikan's binary 2×2 kernel G_2, we use H_2 to represent the non-binary 2×2 kernel we built. At the same time, since this work mainly discusses NBPCs constructed by H_2, it can be known from Corollary 1 that $H_{2,2}$ must be a tag-element and $H_{1,2}$ is a primitive element. Then H_2 can be expressed as

$$H_2 = \begin{bmatrix} \gamma & 0 \\ \alpha & \delta \end{bmatrix} (\gamma, \delta \in GF(q), \gamma, \delta \neq 0, q = 2^m) \tag{2}$$

where α is the primitive element over $GF(q)$, that is, the element set over $GF(q)$ can be expressed as $\{0, 1, \alpha, \alpha^2, \ldots, \alpha^{q-2}\}$.

The process of $GF(q)$ polar encoder in Fig. 1 can be described by

$$x_1^{N_s} = u_1^{N_s} B_{N_s} H_2^{\otimes n} \tag{3}$$

where B_{N_s} is a permutation matrix and $n = \log{}_2 N_s$. All the addition and multiplication operations are defined over $GF(q)$.

The kernel can be represented by the basic unit, where the binary kernel constructed by Arikan can be expressed as $(u + v, v)$ kernel shown in Fig. 2. The relationship between input and output of the basic unit of the binary version is written as

$$\begin{cases} x_1 = u_1 + u_2 \\ x_2 = u_2 \end{cases} \tag{4}$$

Fig. 3 shows the basic unit corresponding to the kernel H_2 of proposed NBPCs, where the relationship between input and output is described as follows

$$\begin{cases} x_1 = \gamma u_1 + \alpha u_2 \\ x_2 = \delta u_2 \end{cases} \tag{5}$$

In Fig. 4, we take $N_s = 8$ as an example to show equation (3) completed by the encoder with a Tanner graph. There are n stages in the Tanner graph, which includes $N_s/2$ boxes at each stage. Each box represents the basic unit shown in Fig. 3. The dotted box between every two stages represents the "reverse shuffle" operator, which can be represented by R_{N_s}. The function of R_8 shown in Fig. 4 is expressed as

$$\begin{aligned} &(v_{000}, v_{001}, v_{010}, v_{011}, v_{100}, v_{101}, v_{110}, v_{111}) \times R_8 \\ &= (v_{000}, v_{010}, v_{100}, v_{110}, v_{001}, v_{011}, v_{101}, v_{111}) \end{aligned} \tag{6}$$

The relation between permutation matrix B_{N_s} and reverse shuffle operator R_{N_s} can be written as

$$B_{N_s} = R_{N_s}(I_2 \otimes B_{N_s/2}) \tag{7}$$

where I_2 is a two-dimensional identity matrix.

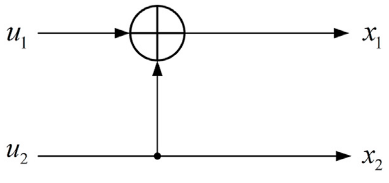


Fig. 2. The basic unit of BPC encoder.

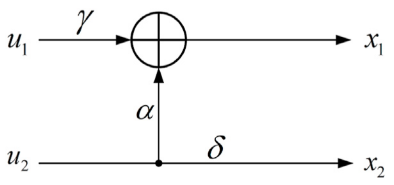


Fig. 3. The basic unit of NBPC encoder over $GF(2^m)$.

3.3. *Monte-Carlo Simulation for NBPCs*

Motivated by Ref. [25], the implementation steps of the Monte-Carlo simulation are detailed for the design of NBPCs. The Monte-Carlo simulation block diagram for proposed NBPCs is shown in Fig. 5.

For each Monte-Carlo simulation, all symbol values in the information vector $u_1^{N_s}$ are randomly generated from the elements over the Galois field, which guarantees the fairness of the results. The encoding and transmission procedures are similar to that in Fig. 1, where the operation of adding CRC bits and frozen symbols is removed. Especially, the Genie-aided SC decoder is performed for Monte-Carlo simulation, which is a special SC decoder with u_1^{i-1} in advance. The decoder ensures that the estimated result of the previous symbol is correct. Thus, the Genie-aided SC decoder is not affected by the previous decoding result $\widehat{u}_1^{i-1}$. Note that the process of Genie-aided SC decoding is detailed in Section 4.1.

Suppose that M simulations are performed, where M should be at least 10^3 for the accuracy of the results. Note that for the universality of the construction, SNR should not be set too high or low in the Monte-Carlo simulation, where SNR $\in [-3, 3]$ in the unit of dB is adopted in this paper. To evaluate the polarized channel reliability, let E_i $(i = 1, 2, \ldots, N_s)$ denote the error number of decoded symbol $\widehat{u}_i$, and $e_i = E_i/M$ denote the symbol error rate. Given a bit code length N_b, the $N_b Rm$ channel indexes with a lower e_i are selected as information symbol positions according to the sorting of e_i, while the remaining positions are placed with frozen symbols, i.e., typically 0. As such, $N_b R$ information bits, namely payloads, are transformed into the field symbol by the bit-to-symbol converter, which will then be placed at the selected information symbol positions. Thus, the flexible configuration of code rate R or payload size K_b is available in the Monte-Carlo simulation. Note that the sorting results of e_i for the same N_b and different R are not identical since the Gaussian noise variance σ^2 varies with R according to equation (1).

In this paper, Monte-Carlo simulations with $M = 10^4$ are fixed to construct NBPCs, which are typically chosen in the literature as an appropriate trade-off of construction complexity and accuracy. Fig. 6 describes the polarized channel performance results of $N_s = 128$ and $N_s = 512$ by 10,000 Monte-Carlo simulations at SNR $= 2$ dB. The kernel's relevant parameters are set to $\gamma = 1$, $\delta = 1$, and $m = 2$. It can be seen that the so-called pure noise channels have a high e_i of about 0.75. Besides, polarized channels with $e_i = 0$ cannot be found in Fig. 6 due to the log plot, which is regarded as noiseless channels approximately. Intuitively, we can estimate channel performance and pick out the good channels and bad channels through the Monte-Carlo simulation, where information symbols and frozen symbols are placed respectively. The impact of this process on the performance of the constructed NBPCs is very critical. The results of the Monte-Carlo simulation are also employed in PPB-SCL decoding, which is detailed in Section 4.3.

4. Decoding of Non-Binary Polar Codes

In this paper, we focus on the SCL decoding algorithm for NBPCs. Since the decoder of NBPCs is a parallel process with multi-bit decoded simultaneously, we take the filed symbol as the main unit for the convenience of expression in the following description. Specifically, the

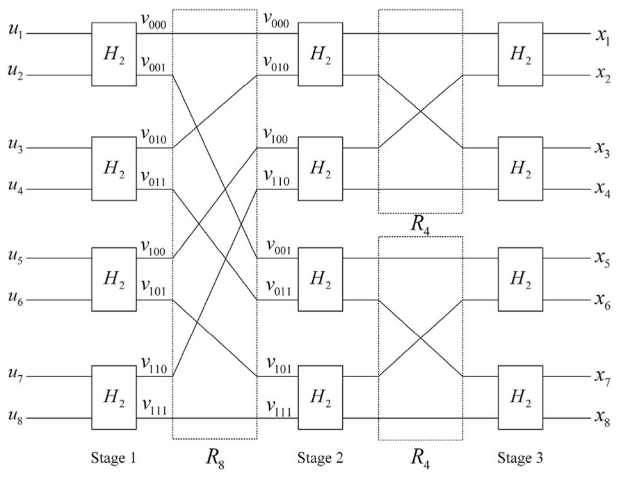


Fig. 4. Tanner graph of NBPC encoder with code length $N_s = 8$.

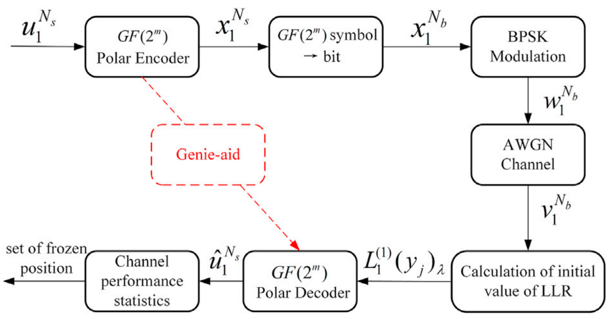


Fig. 5. System model of Monte-Carlo simulation.

proposed LLR recursive function for NBPCs is illustrated in Section 4.1, while the SCL and PPB-SCL decoding algorithms based on LLRs are presented in Section 4.2 and Section 4.3, respectively.

4.1. LLR recursive function for NBPCs

In the process of channel polarization, the transition probability of the i-th synthetic channel is defined as $W_{N_s}^{(i)}(y_1^{N_s}, \hat{u}_1^{i-1}|u_i)$, where $y_1^{N_s}$ is the channel output sequence. The LLR corresponding to the symbol u_i with λ can be defined as

$$L_{N_s}^{(i)}(y_1^{N_s}, \hat{u}_1^{i-1})_\lambda = \ln \frac{W_{N_s}^{(i)}(y_1^{N_s}, \hat{u}_1^{i-1}|0)}{W_{N_s}^{(i)}(y_1^{N_s}, \hat{u}_1^{i-1}|\lambda)} \tag{8}$$

where $\lambda \in GF(2^m)$.

For BPCs, λ in equation (8) is taken as 0 or 1, i.e., two LLRs for each symbol u_i. However, when $\lambda = 0$, the value of LLRs is always equal to 0. Hence, only the case of $\lambda = 1$ needs to be discussed for BPCs. $\hat{u}_i$, the estimated value of u_i, can be obtained by comparing LLRs with 0. For NBPCs, λ in equation (8) can be taken as 0, 1,…, and $q-1$. Similarly, when $\lambda = 0$, we have $L_{N_s}^{(i)}(y_1^{N_s}, \hat{u}_1^{i-1})_0 \equiv 0$. We need to calculate the remaining (q-1) LLRs in each decoding stage, and then compare all the LLR values to get the decoding result. According to the definition of equation (8), in order to obtain the results when the transition probability $W_{N_s}^{(i)}(y_1^{N_s}, \hat{u}_1^{i-1}|\lambda)$ takes the maximum value, the hard decision function for $\hat{u}_i$ can be expressed as

$$\hat{u}_i = \begin{cases} \arg\min\limits_{\lambda \in GF(2^m)} L_{N_s}^{(i)}(y_1^{N_s}, \hat{u}_1^{i-1})_\lambda & i \in \mathcal{A} \\ u_i & i \in \mathcal{A}^c \end{cases} \tag{9}$$

For NBPCs, the recursive function in Ref. [1] is not applicable. According to the partial distances of the kernel [20] and the probability domain-based decoding of NBPCs [27], we generally provide the LLR recursive function for the proposed NBPCs with arbitrary kernel size, which can be described as

$$\hat{R}_{t,\lambda}^{(i)} = \ln \frac{\sum_{\omega_{t+1}^T} \exp\left(-\sum_{r=1}^{T} R_{r,x_{1r}}^{(i)}\right)}{\sum_{\omega_{t+1}^T} \exp\left(-\sum_{r=1}^{T} R_{r,x_{2r}}^{(i)}\right)} \tag{10}$$

where $\hat{R}_{t,\lambda}^{(i)}$ is the updated LLR with the symbol value λ and $R_{r,x}$ represents the input LLR with the symbol value x. Note that we have $\hat{R}_{t,0} = 0$. To be specific, $\hat{R}_{t,\lambda}^{(i)}$ and $R_{r,x}$ can be expressed as

$$\hat{R}_{t,\lambda}^{(i)} = L_{N_s}^{(Ti-(T-t))}\left(y_1^{N_s}, \hat{u}_1^{Ti-(T-t)-1}\right)_\lambda \tag{11}$$

$$R_{r,x}^{(i)} = \begin{cases} L_{N_s/T}^{(i)}\left(y_1^{N_s/T}, g_1(\hat{u})\right)_x, r = 1 \\ L_{N_s/T}^{(i)}\left(y_{(r-1)N_s/T+1}^{rN_s/T}, g_r(\hat{u})\right)_x, r > 1 \end{cases} \tag{12}$$

where $t \in \{1, 2,\ldots, T\}$, $i \in \{1, 2, \ldots, N_s/T\}$.

Equation (11) shows that NBPCs constructed by kernel G_T have a total of T LLR update functions, where $\hat{R}_{1,\lambda}^{(i)}$ is calculated to obtain $L_{N_s}^{(1)}, L_{N_s}^{(1+T)}, \ldots, L_{N_s}^{(1+N_s-T)}$, and $\hat{R}_{2,\lambda}^{(i)}$ is calculated to obtain $L_{N_s}^{(2)}, L_{N_s}^{(2+T)}, \ldots, L_{N_s}^{(2+N_s-T)}$, and so on. The values of symbol $x_{1,r}$ and $x_{2,r}$ of the input LLRs can be obtained by

$$(x_{11}, x_{12}, \ldots, x_{1T}) = g(\hat{\mu}_1^{t-1}, 0, \omega_{t+1}^T) \tag{13.a}$$

$$(x_{21}, x_{22}, \ldots, x_{2T}) = g(\hat{\mu}_1^{t-1}, \lambda, \omega_{t+1}^T) \tag{13.b}$$

where $g(\cdot)$ refers to the kernel function with $g(u_1^T) = u_1^T G_T$ and $\hat{\mu}_1^{t-1}$ represents (t-1) symbol feedback from the decoder, which can be indicated explicitly as $\hat{\mu}_1^{t-1} = \hat{u}_{Ti-(T-1)}^{Ti-(T+1-t)}$. The calculation of $\hat{R}_{t,\lambda}^{(i)}$ requires not only the input LLRs but also the previous (t-1) decision feedback. ω_{t+1}^T is an arbitrary vector over $GF(q)$, which represents all possible symbol values that have not been decoded.

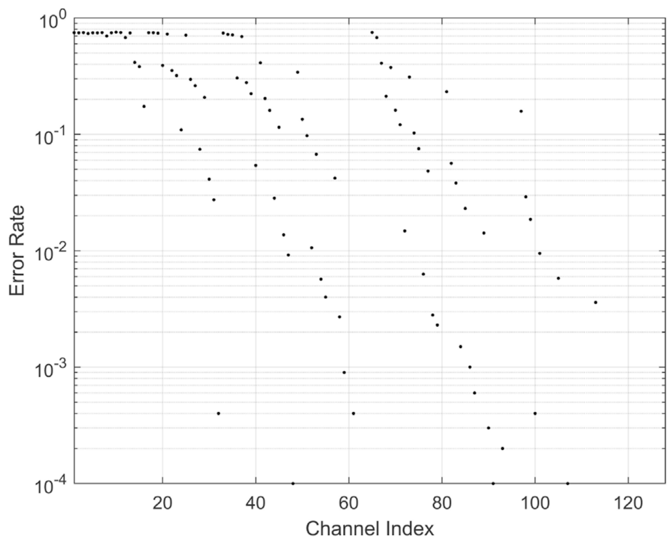
Equations (12)-(13) show that the input LLRs in equation (10) also have T forms, and each one contains 2^m different symbol values. In other words, if we want to calculate the LLRs of length N_s with all symbol values, we need to input T groups of "LLR families"of length N_s/T, each of which contains 2^m LLRs with different estimated symbol values, namely, a total of $T \cdot 2^m$ input LLRs are considered. As λ goes through all possible symbol values, each x calculated by $g(\cdot)$ also traverses all values, thus the input LLRs with every symbol value are needed.

Additionally, $g_r(\hat{u})$ in equation (12) represents an update to the feedback $\hat{u}_1^{Ti-T}$, which can be expressed as

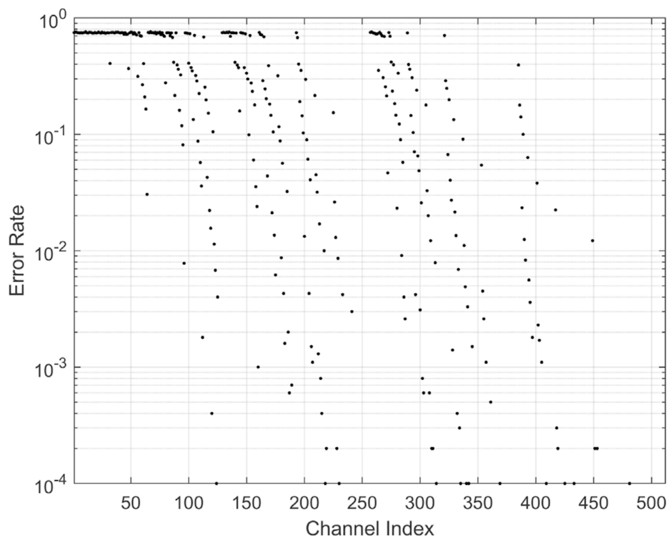
$$g_r(\hat{u}) = \hat{u} G_{T(r)}, r \in \{1, 2, \ldots, T\} \tag{14}$$

where $\hat{u} = (\hat{u}_{1,1}^{Ti-T}, \hat{u}_{1,2}^{Ti-T}, \ldots, \hat{u}_{1,T-1}^{Ti-T}, \hat{u}_{1,0}^{Ti-T})$.

Let $\hat{u}_{1,z}^{Ti-T}$ denote a sub-vector composed of all elements whose index in vector $\hat{u}_1^{Ti-T}$ is equal to z after a mod-T operation. In particular, when T is equal to 2, we use $\hat{u}_{1,o}^{2i-2}$ and $\hat{u}_{1,e}^{2i-2}$ to represent $\hat{u}_{1,1}^{2i-2}$ and $\hat{u}_{1,0}^{2i-2}$,



(a) $N_s = 128$



(b) $N_s = 512$

Fig. 6. The performance of each channel under Genie-aided SC decoding with SNR = 2 dB.

respectively, that is, the odd index sub-vector and the even index sub-vector. Note that $\widehat{u}$ is just a vector symbol here to be involved in the calculation of $g_r(\widehat{u})$ and has no practical meaning. $\widehat{u}_{1,z}^{Ti-T}$ is treated as an independent element in the vector $\widehat{u}$. In short, they only have a symbolic meaning in the operation of $g_r(\widehat{u})$ and do not represent vectors.

The termination condition for recursion is $N_s = 1$. The LLRs at this time can be expressed as

$$L_1^{(1)}(y_j)_\lambda = \ln\frac{W(y_j|0)}{W(y_j|\lambda)} \tag{15}$$

Unfortunately, equation (15) is not operational in practice because the transition probability W is unknown. Due to the BPSK modulation and BI-AWGN channels are considered in this paper, we extend the Gaussian approximation calculation method of the initial LLR value for BPCs in Refs. [32,33] under the same transmission mode to this work. As a result, $L_1^{(1)}(y_j)_\lambda$ can be calculated by

$$L_1^{(1)}(y_j)_\lambda = \frac{2\times f(v_{jm-m+1},\ldots,v_{jm})}{\sigma^2} \tag{16}$$

Before applying equation (16), we need to write the symbol λ into a form of m-bit binary number according to the conversion relationship established in advance, i.e., there is a transformation $\lambda \rightarrow \{\lambda_1,\lambda_2,\ldots,\lambda_m\}$. The function f is defined as

$$\begin{aligned} f(v_{jm-m+1},\ldots,v_{jm}) = \lambda_1 v_{jm-m+1} + \lambda_2 v_{jm-m+2} \\ +\cdots+\lambda_m v_{jm} \end{aligned} \tag{17}$$

The detailed steps of Genie-aided SC decoding are given in Algorithm 1. Note that u_1^{i-1} is substituted into $\widehat{u}_1^{i-1}$ before calculating LLRs in order to eliminate the impact of previous decoding results on subsequent decisions.

Algorithm 1 Genie-aided SC decoding for Monte-Carlo Simulation in NBPCs

```
Input: Uncoded sequence $u_1^i$, LLR initial value $L_1^{(1)}(y_j)_\lambda$
Output: Decoded sequence $\hat{u}_1^i$
1: for $i \in \{1,2,\ldots,N_s\}$ do
2:     $\hat{u}_1^{i-1} \leftarrow u_1^{i-1}$;
3:     $\hat{u}_i \leftarrow \arg\min_{\lambda\in GF(2^m)} L_{N_s}^{(i)}(y_1^{N_s},\hat{u}_1^{i-1})_\lambda$;
4: end for
5: Return $\hat{u}_1^i$
```

The operation of equations (16) and (17) is used to calculate the LLR initial value for the proposed system model. Until now, we have summarized the whole process of LLR recursive operation. However, the exponential and logarithmic calculations of equation (10) are difficult to be implemented for hardware in practice. Therefore, we give an improved function based on a hardware-friendly version. By the defined hardware-friendly equation (18), we rewrite equation (10) into equation (19) to obtain the hardware-friendly LLR update function.

$$\ln\left\{\sum_i \exp[\psi_i(\cdot)]\right\} \approx \max[\psi_i(\cdot)] \tag{18}$$

$$\widehat{R}_{t,\lambda}^{(i)} = \max_{\omega_{t+1}^T}\left(-\sum\nolimits_{r=1}^T R_{r,x_{1r}}^{(i)}\right) - \max_{\omega_{t+1}^T}\left(-\sum\nolimits_{r=1}^T R_{r,x_{2r}}^{(i)}\right) \tag{19}$$

Especially, for $\forall i \in \{1, 2, \ldots, N_s/2\}$ and $\lambda, \omega \in GF(2^m)$, the LLR update function of NBPCs with 2×2 non-binary kernel H_2 can be written as

$$\begin{cases} \widehat{R}_{1,\lambda}^{(i)} = \max\limits_{\omega}\,(-R_{1,\omega\alpha}^{(i)} - R_{2,\omega\delta}^{(i)}) - \max\limits_{\omega}\,(-R_{1,\lambda\gamma\oplus\omega\alpha}^{(i)} - R_{2,\omega\delta}^{(i)}) \\ \widehat{R}_{2,\lambda}^{(i)} = (R_{1,\gamma\hat{u}_{2i-1}\oplus\lambda\alpha}^{(i)} + R_{2,\lambda\delta}^{(i)}) - (R_{1,\gamma\hat{u}_{2i-1}}^{(i)} + R_{2,0}^{(i)}) \end{cases} \tag{20}$$

where LLRs on the left side of the equation represent $\widehat{R}_{1,\lambda}^{(i)} = L_{N_s}^{(2i-1)}(y_1^{N_s},\widehat{u}_1^{2i-2})_\lambda$, $\widehat{R}_{2,\lambda}^{(i)} = L_{N_s}^{(2i)}(y_1^{N_s},\widehat{u}_1^{2i-1})_\lambda$, respectively and LLRs on the right side of the equation are $R_{1,x}^{(i)} = L_{N_s/2}^{(i)}(y_1^{N_s/2},\gamma\widehat{u}_{1,o}^{2i-2} \oplus \alpha\widehat{u}_{1,e}^{2i-2})_x$, $R_{2,x}^{(i)} = L_{N_s/2}^{(i)}(y_{N_s/2+1}^{N_s},\delta\widehat{u}_{1,e}^{2i-2})_x$, respectively. Both addition and multiplication for symbols are defined over $GF(2^m)$.

For $m = 2$, there is $L_1^{(1)}(y_j)_\lambda = \frac{2\times(\lambda_1 v_{2j-1}+\lambda_2 v_{2j})}{\sigma^2}$ at the end of the recursion, where $\lambda \rightarrow \{\lambda_1,\lambda_2\}$ represents the binary transformation of λ. The derivation process is illustrated below.

Here, we have the global variable $T = 2$. Then, the following equation is obtained by substituting $t = 1$ into equation (19)

$$\widehat{R}_{1,\lambda}^{(i)} = \max_{\omega_2^2}\,(-R_{1,x_{11}}^{(i)} - R_{2,x_{12}}^{(i)}) - \max_{\omega_2^2}\,(-R_{1,x_{21}}^{(i)} - R_{2,x_{22}}^{(i)}) \tag{21}$$

We use ω to replace the only element in ω_2^2, and substitute $t = 1$ into equations (13.a) and (13.b), then

$$(x_{11},x_{12})=g(0,\omega)=(0,\omega)\begin{bmatrix}\gamma & 0\\ \alpha & \delta\end{bmatrix}=(\omega\alpha,\omega\delta) \tag{22.a}$$

$$(x_{21},x_{22})=g(\lambda,\omega)=(\lambda,\omega)\begin{bmatrix}\gamma & 0\\ \alpha & \delta\end{bmatrix}=(\lambda\gamma\oplus\omega\alpha,\omega\delta) \tag{22.b}$$

When $t=1$, it is not necessary to input the previous decoding feedback. By substituting the above results into equation (21) and using equations (11-12) to express the updated LLRs and input LLRs respectively, the first update function can be written as equation (23). Similarly, we put $t=2$ into equation (19) and repeat the above steps. The second update function can be obtained from equation (24).

$$\begin{aligned}\widehat{R}^{(i)}_{1,\lambda}&=L^{(2i-1)}_{N_s}(y_1^{N_s},\widehat{u}_1^{2i-2})_\lambda\\&=\max_\omega(-R^{(i)}_{1,\omega\alpha}-R^{(i)}_{2,\omega\delta})-\max_\omega(-R^{(i)}_{1,\lambda\gamma\oplus\omega\alpha}-R^{(i)}_{2,\omega\delta})\\&=\max_\omega\left[-L^{(i)}_{N_s/2}(y_1^{N_s/2},\gamma\widehat{u}_{1,o}^{2i-2}\oplus\alpha\widehat{u}_{1,e}^{2i-2})_{\omega\alpha}-L^{(i)}_{N_s/2}(y_{N_s/2+1}^{N_s},\delta\widehat{u}_{1,e}^{2i-2})_{\omega\delta}\right]\\&\quad-\max_\omega\left[-L^{(i)}_{N_s/2}(y_1^{N_s/2},\gamma\widehat{u}_{1,o}^{2i-2}\oplus\alpha\widehat{u}_{1,e}^{2i-2})_{\lambda\gamma\oplus\omega\alpha}-L^{(i)}_{N_s/2}(y_{N_s/2+1}^{N_s},\delta\widehat{u}_{1,e}^{2i-2})_{\omega\delta}\right]\end{aligned} \tag{23}$$

$$\begin{aligned}\widehat{R}^{(i)}_{2,\lambda}&=L^{(2i)}_{N_s}(y_1^{N_s},\widehat{u}_1^{2i-1})_\lambda\\&=(R^{(i)}_{1,\gamma\hat{u}_{2i-1}\oplus\lambda\alpha}+R^{(i)}_{2,\lambda\delta})-(R^{(i)}_{1,\gamma\hat{u}_{2i-1}}+R^{(i)}_{2,0})\\&=\left[L^{(i)}_{N_s/2}(y_1^{N_s/2},\gamma\widehat{u}_{1,o}^{2i-2}\oplus\alpha\widehat{u}_{1,e}^{2i-2})_{\gamma\hat{u}_{2i-1}\oplus\lambda\alpha}+L^{(i)}_{N_s/2}(y_{N_s/2+1}^{N_s},\delta\widehat{u}_{1,e}^{2i-2})_{\lambda\delta}\right]\\&\quad-\left[L^{(i)}_{N_s/2}(y_1^{N_s/2},\gamma\widehat{u}_{1,o}^{2i-2}\oplus\alpha\widehat{u}_{1,e}^{2i-2})_{\gamma\hat{u}_{2i-1}}+L^{(i)}_{N_s/2}(y_{N_s/2+1}^{N_s},\delta\widehat{u}_{1,e}^{2i-2})_{0}\right]\end{aligned} \tag{24}$$

Note that ω_{t+1}^T is an empty vector when $t=2$, hence the maximum function can be removed. Moreover, the decoding feedback $\widehat{u}_{2i-1}$ needs to be input into equations (13.a) and (13.b), which leads to the second update function with two input LLRs and an estimate of the previous symbol.

If $m=2$, $L_1^{(1)}(y_j)_\lambda=\frac{2\times(\lambda_1 v_{2j-1}+\lambda_2 v_{2j})}{\sigma^2}$ can be easily obtained from equations (16-17). The above has proved the conclusion.

Fig. 7 shows a general example of SC decoding for NBPCs over $GF(4)$ with $N_s=8$. The whole decoding process is carried out from the right to the left. The preparation stage is the calculation process of the LLR initial values in the system model. Stages 1–3 are LLR recursive processes, and stage 4 is the decision part using equation (9). In stages 1–3, $\widehat{R}^{(i)}_{1,\lambda}$ and $\widehat{R}^{(i)}_{2,\lambda}$ are computed by the first and second update function, which are represented in gray and white, respectively. It can be seen that the update of all white nodes requires additional input of feedback from the previous decoding. Besides, the symbol value λ of each LLR is not specifically given in the figure. In fact, each node stores the LLR value with $\lambda\in\{0, 1, 2, 3\}$. In other words, each update needs to calculate the LLR with four symbol values in the calculation process from the right to the left.

$$\begin{aligned}\Omega^{(i)}_\lambda&=\max_\omega\left\{-L^{((i+1)/2)}_{N_s/2}(y_1^{N_s/2},\gamma\widehat{u}_{1,o}^{2i-2}\oplus\alpha\widehat{u}_{1,e}^{2i-2}[j])_{\lambda\gamma\oplus\omega\alpha}-L^{((i+1)/2)}_{N_s/2}(y_{N_s/2+1}^{N_s},\delta\widehat{u}_{1,e}^{2i-2}[j])_{\omega\delta}\right\}\\&\quad-\max_{\omega,\theta}\left\{-L^{((i+1)/2)}_{N_s/2}(y_1^{N_s/2},\gamma\widehat{u}_{1,o}^{2i-2}\oplus\alpha\widehat{u}_{1,e}^{2i-2}[j])_{\theta\gamma\oplus\omega\alpha}-L^{((i+1)/2)}_{N_s/2}(y_{N_s/2+1}^{N_s},\delta\widehat{u}_{1,e}^{2i-2}[j])_{\omega\delta}\right\}\end{aligned} \tag{28.a}$$

4.2. SCL decoding based on LLR

For SCL decoding, the maximum width of search paths is termed as the list size L. After completing the path splitting, we select the L paths with the smallest PMs, which are saved in a list for the next stage. According to Ref. [29], if the estimated value of u_i is not identical to the decision result of SC, the PMs of this level add a "penalty value" to the result of the previous level.

Corollary 2. *For BPCs, the LLR-based penalty value ϕ for any bit u_i in the j-th path can be expressed as*

$$\phi=\left|\ln\frac{W_N^{(i)}(y_1^N,\widehat{u}_1^{i-1}[j]|\widehat{u}_i[j]=\lambda)}{W_N^{(i)}(y_1^N,\widehat{u}_1^{i-1}[j]|\widehat{u}_i[j]=t)}\right| \tag{25}$$

where $t\in\mathcal{I}\setminus\{\lambda\}$ and $\mathcal{I}=\{0,1\}$ are the symbol alphabet.

Proof: When $\lambda=1$, we have $t=0$

and $\phi=\left|\ln\frac{W_N^{(i)}(y_1^N,\widehat{u}_1^{i-1}[j]|1)}{W_N^{(i)}(y_1^N,\widehat{u}_1^{i-1}[j]|0)}\right|=\left|-L_N^{(i)}[j]\right|=\left|L_N^{(i)}[j]\right|$;

then when $\lambda=0$, we have $t=1$

and $\phi=\left|\ln\frac{W_N^{(i)}(y_1^N,\widehat{u}_1^{i-1}[j]|0)}{W_N^{(i)}(y_1^N,\widehat{u}_1^{i-1}[j]|1)}\right|=\left|L_N^{(i)}[j]\right|$, where $L_N^{(i)}[j]$ represents the LLR of a given channel output y_1^N and the past trajectory of the path $\widehat{u}_1^{i-1}[j]$ for the bit u_i.

From Corollary 2, ϕ is determined by the LLR between the current estimated value and other possible values. Based on this, then we get the following Corollary 3 for NBPCs.

Corollary 3. *For NBPCs, the LLR-based penalty value ϕ_λ that takes the estimated value λ for any symbol u_i in the j-th path can be expressed as*

$$\phi_\lambda=\left|\ln\frac{W_{N_s}^{(i)}(y_1^{N_s},\widehat{u}_1^{i-1}[j]|\widehat{u}_i[j]=\lambda)}{W_{N_s}^{(i)}(y_1^{N_s},\widehat{u}_1^{i-1}[j]\widehat{u}_i[j]\in\Lambda)}\right| \tag{26}$$

where $\Lambda=\mathcal{I}\setminus\{\lambda\}$ and $\mathcal{I}=\{0,1,\dots,q-1\}$ are the symbol alphabet.

Theorem 1. *For NBPCs generated by the kernel H_2, the PM $p_j^{(i)}$ corresponding to any path $j\in\{1, 2, \dots, L\}$ and any symbol u_i is defined as*

$$p_j^{(i)}=\begin{cases}p_j^{(i-1)}, \text{if } \widehat{u}_i[j]=\arg\min\limits_{\lambda\in GF(2^m)} L_{N_s}^{(i)}(y_1^{N_s},\widehat{u}_1^{i-1}[j])_\lambda\\ p_j^{(i-1)}+\left|\Omega_\lambda^{(i)}\right|, \text{otherwise}\end{cases} \tag{27}$$

where $\left|\Omega_\lambda^{(i)}\right|$ represents the penalty value ϕ_λ of NBPCs and the initial value of $p_j^{(0)}=0$. $\Omega_\lambda^{(i)}$ is defined by equation (28.a) for an odd i or equation (28.b) for an even i, where $\theta\in\Lambda$.

$$\begin{aligned}\Omega^{(i)}_\lambda=&\max_\theta\left\{L^{(i/2)}_{N_s/2}(y_1^{N_s/2},\gamma\widehat{u}_{1,o}^{2i-2}\oplus\alpha\widehat{u}_{1,e}^{2i-2}[j])_{\gamma\widehat{u}_{2i-1}\oplus\theta\alpha}+L^{(i/2)}_{N_s/2}(y_{N_s/2+1}^{N_s},\delta\widehat{u}_{1,e}^{2i-2}[j])_{\theta\delta}\right\}\\&-\left\{L^{(i/2)}_{N_s/2}(y_1^{N_s/2},\gamma\widehat{u}_{1,o}^{2i-2}\oplus\alpha\widehat{u}_{1,e}^{2i-2}[j])_{\gamma\widehat{u}_{2i-1}\oplus\lambda\alpha}+L^{(i/2)}_{N_s/2}(y_{N_s/2+1}^{N_s},\delta\widehat{u}_{1,e}^{2i-2}[j])_{\lambda\delta}\right\}\end{aligned} \tag{28.b}$$

Proof: From Corollary 3 and the definition of LLRs, ϕ_λ can be rewritten as (29). Here $L_{N_s}^{(i)}(y_1^{N_s},\widehat{u}_1^{i-1}[j])_x$ is the LLR of the given channel output $y_1^{N_s}$ and the past trajectory of the path $\widehat{u}_1^{i-1}[j]$, which takes the symbol value x. By using equation (20), we can further get the results of equations (28.a) and (28.b).

$$\begin{aligned}\phi_\lambda &= \left|\ln\frac{W_{N_s}^{(i)}\left(y_1^{N_s},\hat{u}_1^{i-1}[j]|\hat{u}_i[j]=\lambda\right)}{W_{N_s}^{(i)}\left(y_1^{N_s},\hat{u}_1^{i-1}[j]|\hat{u}_i[j]\in\Lambda\right)}\right| \\ &= \left|\ln\frac{W_{N_s}^{(i)}\left(y_1^{N_s},\hat{u}_1^{i-1}[j]|0\right)}{W_{N_s}^{(i)}\left(y_1^{N_s},\hat{u}_1^{i-1}[j]|\hat{u}_i[j]\in\Lambda\right)} - \ln\frac{W_{N_s}^{(i)}\left(y_1^{N_s},\hat{u}_1^{i-1}[j]0\right)}{W_{N_s}^{(i)}\left(y_1^{N_s},\hat{u}_1^{i-1}[j]\hat{u}_i[j]\in\lambda\right)}\right| \\ &= \left|L_{N_s}^{(i)}\left(y_1^{N_s},\hat{u}_1^{i-1}[j]\right)_\Lambda - L_{N_s}^{(i)}\left(y_1^{N_s},\hat{u}_1^{i-1}[j]\right)_\lambda\right|\end{aligned} \tag{29}$$

Fig. 8 shows an example of an SCL decoding tree over $GF(4)$, where information symbol position set is $\mathcal{A}=\{1,3,4\}$. The number adjacent to the visited node indicates the PM while the red line indicates the reserved paths so far. The decoding tree of NBPCs over $GF(q)$ with a length of N_s is a q-ary tree with a depth N_s. The root node indicates an empty state while the child nodes at the i-th level represent symbol values over $GF(q)$ of $\hat{u}_i$. Thus, the complete decoding path is represented by the vector $\hat{u}_1^{N_s}$ from the root node to the leaf node. The steps of SCL decoding for NBPCs are detailed in Algorithm 2. The decoding process shown in Fig. 8 can be embodied in Algorithm 2.

Algorithm 2 LLR-based SCL decoding for NBPCs

Input: maximum search width L, $I(I\le L)$ decoding paths from the previous level and PMs $p_j^{(i-1)}$, information symbol position set $\mathcal{A}$, LLR initial value $L_1^{(1)}(y_j)_\lambda$

Output: $\min(2^m I, L)$ decoding paths and their updated PMs $p_j^{(i)}$

```
1:  for $i \in \{1,2,\ldots,N_s\}$ do
2:      for $j \in \{1,2,\ldots,I\}$ do
3:          if $i \in \mathcal{A}^c$ then
4:              $\hat{u}_i[j] = 0$;
5:          else
6:              Let $\hat{u}_i[j]$ take a value over $GF(2^m)$ in turn;
7:          end if
8:          Get and compute the LLRs in $\Omega_\lambda^{(i)}$;
9:          Compute recursively $L_{N_s}^{(i)}(y_1^{N_s},\hat{u}_1^{i-1}[j])_\lambda$, for $\forall\lambda\in GF(2^m)$;
10:         Put the result in step 9 into equation (28) to get $\Omega_\lambda^{(i)}$, and use equation (27) to calculate $p_j^{(i)}$;
11:     end for
12:     if $2^m I \le L$ then
13:         Return all decoding paths and $p_j^{(i)}$;
14:     else
15:         Sort $2^m I$ paths according to the size of $p_j^{(i)}$;
16:         Return $L$ decoding paths with smaller $p_j^{(i)}$;
17:     end if
18: end for
```

4.3. PPB-SCL decoding based on LLR

From equation (27), two types of LLRs, i.e., $\Omega_\lambda^{(i)}$ and $L_{N_s}^{(i)}(y_1^{N_s},\hat{u}_1^{i-1}[j])_\lambda$, are required for calculations, which lead to the high complexity for hardware implementation. Moreover, the calculation of $\Omega_\lambda^{(i)}$ is different from the general equation (20), which includes at least $q(q-2)$ more cases. Enlightened by the hybrid decoder [34,35], we propose an LLR-based PPB-SCL decoding to reduce decoding complexity of NBPCs, which SC performs decoding at highly reliable positions and SCL decoding at less reliable positions.

For PPB-SCL decoding, we first divide the information sub-channels into perfect polarized (i.e., highly reliable) channels and imperfect polarized (i.e., less reliable) channels based on the results of channel reliability in the Monte-Carlo simulation. The definition of the perfect polarized channel position set is detailed as follows.

Definition 2. *Sort the symbol error rate e_i of each channel in the Monte-Carlo simulation and select τ $(1\le\tau\le K_s)$ channels with lower e_i. We term the set $\mathcal{B}$ composed of the indexes of highly reliable channels as the perfect polarized channel position set, where $\tau=|\mathcal{B}|$ and $\mathcal{B}\subseteq\mathcal{A}$. Then, the perfect polarization ratio is defined as $\beta=\tau/K_s$. If only τ_0 channels with $e_i=0$ are selected for $\mathcal{B}$, let $\beta_0=\tau_0/K_s$ denote the strictly perfect polarization ratio. Then, we have the range of β, i.e., $\beta_0\le\beta<1$.*

Note that the perfect polarized channel position set $\mathcal{B}$ is not fixed for different code lengths and code rates. Given a code length N_s, code rate R, and appropriate perfect polarization ratio β, the information symbol position set $\mathcal{A}$ and perfect polarized channel position set $\mathcal{B}$ $(\mathcal{B}\subseteq\mathcal{A})$ are uniquely determined via Monte-Carlo simulations. As shown in Table 4, when the code length is fixed, β_0 varies with different code rates, where $GF(4)$ based-NBPCs are considered. Thus, the available set $\mathcal{B}$ is different according to $\beta_0\le\beta<1$. Even though let $\beta=\beta_0$, the selected perfect polarized channel positions are not identical for different code rates. Moreover, the larger the code length and the code rate, the larger τ_0 and β_0. For a large β_0, it is beneficial to adjust β to a larger value to achieve a higher reduction in complexity.

In the proposed PPB-SCL decoding, only one class of LLR calculation is required just like BPCs. In other words, the calculation method in the decoding process is unified when the path is updated. Next, we derive the proposed PM for the PPB-SCL algorithm.

Lemma 1. *If U_i is uniformly distributed in $GF(q)$, then*

$$\frac{W_{N_s}^{(i)}\left(y_1^{N_s},u_1^{i-1}u_i\right)}{P\left(U_1^i=u_1^iY=y_1^{N_s}\right)} = qP\left(Y=y_1^{N_s}\right) \tag{30}$$

Proof: For $\forall u_i\in GF(q)$, we have $P(U_i=u_i)=\frac{1}{q}$. Then

$$\begin{aligned}\frac{W_{N_s}^{(i)}\left(y_1^{N_s},u_1^{i-1}|u_i\right)}{P\left(U_1^i=u_1^i|Y=y_1^{N_s}\right)} &= \frac{P\left(Y=y_1^{N_s},U_1^{i-1}=u_1^{i-1},U_i=u_i\right)}{P(U_i=u_i)P\left(U_1^i=u_1^i|Y=y_1^{N_s}\right)} \\ &= \frac{P\left(Y=y_1^{N_s},U_1^i=u_1^i\right)}{P(U_i=u_i)P\left(U_1^i=u_1^i|Y=y_1^{N_s}\right)} \\ &= \frac{P\left(Y=y_1^{N_s}\right)P\left(U_1^i=u_1^i|Y=y_1^{N_s}\right)}{P(U_i=u_i)P\left(U_1^i=u_1^i|Y=y_1^{N_s}\right)} \\ &= qP\left(Y=y_1^{N_s}\right)\end{aligned} \tag{31}$$

Thus, we complete the proof of Lemma 1.

Lemma 1 can be understood as a q-ary generalization of the binary case in Ref. [29].

Theorem 2. *For PPB-SCL decoding, the PM $p_j^{(i)}$ corresponding to any path $j\in\{1,2,\ldots,L\}$ and any symbol u_i can be calculated as*

$$\begin{aligned}p_j^{(i)} = & \ p_j^{(i-1)} + L_{N_s}^{(i)}\left(y_1^{N_s},\hat{u}_1^{i-1}[j]\right)_\eta \\ & -\min_{\lambda\in GF(q)} L_{N_s}^{(i)}\left(y_1^{N_s},\hat{u}_1^{i-1}[j]\right)_\lambda\end{aligned} \tag{32}$$

where $\eta\in GF(q)$ refers to the estimation of the current path for the i-th symbol.

Proof: Let us change the recursive form of equation (32) to another direct one, which can be described as

$$p_j^{(i)} = \sum_{k=1}^{i}\left[L_{N_s}^{(k)}\left(y_1^{N_s},\hat{u}_1^{k-1}[j]\right)_\eta - \min_{\lambda\in GF(q)} L_{N_s}^{(k)}\left(y_1^{N_s},\hat{u}_1^{k-1}[j]\right)_\lambda\right] \tag{33}$$

Suppose $\partial = \arg\min_{\lambda\in GF(q)} L_{N_s}^{(k)}(y_1^{N_s},\hat{u}_1^{k-1}[j])_\lambda$, and then the above equation

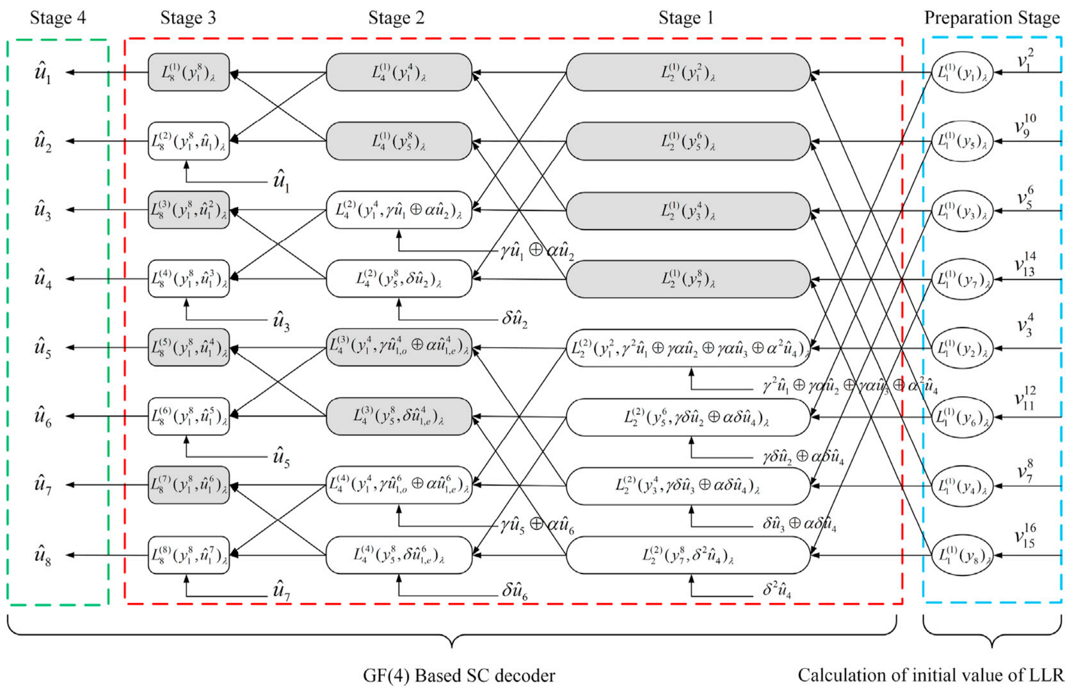


Fig. 7. Butterfly computation graph of SC decoding for a NBPC over $GF(4)$ with $N_s = 8$.

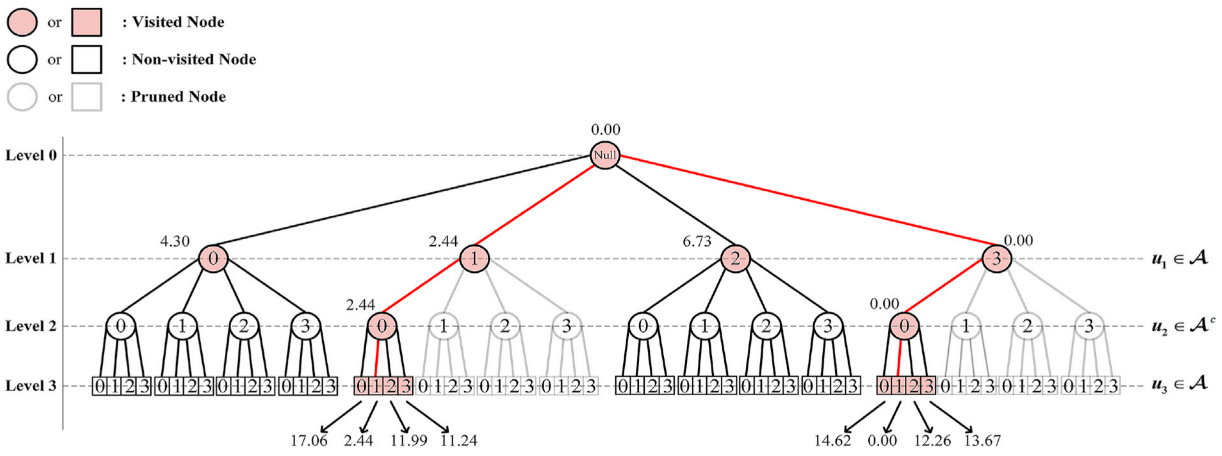


Fig. 8. SCL decoding tree over $GF(4)$ with $N_s = 4$, $K_s = 3$, $L = 2$, where the first three levels are considered.

can continue to be simplified to

$$p_j^{(i)} = \sum_{k=1}^{i}\left[L_{N_s}^{(k)}\left(y_1^{N_s}, \widehat{u}_1^{k-1}[j]\right)_\eta - L_{N_s}^{(k)}\left(y_1^{N_s}, \widehat{u}_1^{k-1}[j]\right)_\partial\right]$$
$$= \sum_{k=1}^{i}\left[\ln\frac{W_{N_s}^{(i)}\left(y_1^{N_s}, \widehat{u}_1^{i-1}[j]|\widehat{u}_i[j]=\partial\right)}{W_{N_s}^{(i)}\left(y_1^{N_s}, \widehat{u}_1^{i-1}[j]|\widehat{u}_i[j]=\eta\right)}\right] \tag{34}$$

Obviously, $\ln\frac{W_{N_s}^{(i)}\left(y_1^{N_s}, \hat{u}_1^{i-1}[j]|\hat{u}_i[j]=\partial\right)}{W_{N_s}^{(i)}\left(y_1^{N_s}, \hat{u}_1^{i-1}[j]|\hat{u}_i[j]=\eta\right)} \geq 0$ always holds in equation (34). Since $x \geq 0$, there is an approximate formula $x \approx \ln(1 + e^x)$, we can further get

$$p_j^{(i)} \approx \sum_{k=1}^{i}\ln\left\{1+\exp\left[\ln\frac{W_{N_s}^{(i)}\left(y_1^{N_s}, \widehat{u}_1^{i-1}[j]|\widehat{u}_i[j]=\partial\right)}{W_{N_s}^{(i)}\left(y_1^{N_s}, \widehat{u}_1^{i-1}[j]|\widehat{u}_i[j]=\eta\right)}\right]\right\} \tag{35}$$

Let $\mathrm{M}_{N_s}^{(i)}[j]$ denote the reciprocal of the ratio between the likelihood information and the maximum likelihood information of the estimated value η for the i-th symbol under the j-th candidate path (the last equation holds if u is uniformly distributed in $GF(q)$), which is defined as

$$\mathrm{M}_{N_s}^{(i)}[j] = \left[\frac{W_{N_s}^{(i)}\left(y_1^{N_s}, \widehat{u}_1^{i-1}[j]|\widehat{u}_i[j]=\eta\right)}{W_{N_s}^{(i)}\left(y_1^{N_s}, \widehat{u}_1^{i-1}[j]|\widehat{u}_i[j]=\partial\right)}\right]^{-1}$$
$$= \frac{P\left(Y = y_1^{N_s}, U_1^{i-1} = \widehat{u}_1^{i-1}[j], U_i = \partial\right)}{P\left(Y = y_1^{N_s}, U_1^{i-1} = \widehat{u}_1^{i-1}[j], U_i = \eta\right)} \tag{36}$$

Table 4
Strictly Perfect Polarized Ratio β_0 with Different Code Lengths and Code Rates.

	$N_s = 16$			$N_s = 64$			$N_s = 256$			$N_s = 1024$		
R	1/3	1/2	2/3	1/3	1/2	2/3	1/3	1/2	2/3	1/3	1/2	2/3
τ_0	1	1	2	8	14	21	53	82	112	280	423	567
β_0	0.200	0.125	0.182	0.381	0.438	0.488	0.624	0.641	0.655	0.821	0.826	0.830

If we simplify equation (35) by employing equation (36), we get the following result:

$$\begin{aligned} p_j^{(i)} &= \sum_{k=1}^{i} \ln\left\{1+\exp\left[\ln \mathrm{M}_{N_s}^{(k)}[j]\right]\right\} \\ &= \sum_{k=1}^{i} \ln\left(1+\mathrm{M}_{N_s}^{(k)}[j]\right) \end{aligned} \tag{37}$$

According to Ref. [29], the PM can be expressed by the logarithm of the posterior probability, namely

$$p_j^{(i)} = -\ln\left(P(U_1^i = \hat{u}_1^i[j] | Y = y_1^{N_s})\right) \tag{38}$$

By comparing equations (37) and (38), it is not difficult to find that showing (33) is equivalent to proving

$$P(U_1^i = \hat{u}_1^i[j] | Y = y_1^{N_s}) = \prod_{k=1}^{i}\left(1+\mathrm{M}_{N_s}^{(k)}[j]\right)^{-1} \tag{39}$$

Since we have

$$\begin{aligned} &P(Y = y_1^{N_s}, U_1^{i-1} = \hat{u}_1^{i-1}[j]) \\ &= \sum_{\hat{u}_i \in GF(q)} P(Y = y_1^{N_s}, U_1^i = \hat{u}_1^i[j]) \\ &= P(Y = y_1^{N_s}, U_1^i = \hat{u}_1^i[j]) \\ &\cdot \left\{1+\sum_{\lambda\in GF(q)} \frac{P(Y = y_1^{N_s}, U_1^{i-1} = \hat{u}_1^{i-1}[j], U_i = \lambda)}{P(Y = y_1^{N_s}, U_1^{i-1} = \hat{u}_1^{i-1}[j], U_i = \eta)}\right\} \\ &\approx P(Y = y_1^{N_s}, U_1^i = \hat{u}_1^i[j])\left(1+\mathrm{M}_{N_s}^{(k)}[j]\right) \end{aligned} \tag{40}$$

the following result is obtained by repeatedly applying the recursive equation (40).

$$P(Y = y_1^{N_s}, U_1^i = \hat{u}_1^i[j]) = P(Y = y_1^{N_s}) \prod_{k=1}^{i}\left(1+\mathrm{M}_{N_s}^{(k)}[j]\right)^{-1} \tag{41}$$

Divide both sides of the equation by $P(Y = y_1^{N_s})$. Equation (39) then can be obtained. Thus, Theorem 2 has been proved.

In Theorem 2, the penalty value of PMs is

$$\phi = L_{N_s}^{(i)}(y_1^{N_s}, \hat{u}_1^{i-1}[j])_\eta - \min_{\lambda\in GF(q)} L_{N_s}^{(i)}(y_1^{N_s}, \hat{u}_1^{i-1}[j])_\lambda \tag{42}$$

which reflects the result of the SC decision.

For any two candidate paths j_1 and j_2, assuming that

$$W_{N_s}^{(i)}(y_1^{N_s}, \hat{u}_1^{i-1}[j_1] | \hat{u}_i[j_1]) < W_{N_s}^{(i)}(y_1^{N_s}, \hat{u}_1^{i-1}[j_2] | \hat{u}_i[j_2]) \tag{43}$$

then $P\cdot(U_1^i = \hat{u}_1^i[j_1] | Y = y_1^{N_s}) < P\cdot(U_1^i = \hat{u}_1^i[j_2] | Y = y_1^{N_s})$ can be obtained according to Lemma 1. Thus, $p_{j_1}^{(i)} > p_{j_2}^{(i)}$ is obtained by the definition of equation (38). Thus, a path with a smaller $p_j^{(i)}$ still shows higher reliability.

Algorithm 3 LLR-based PPB-SCL decoding for NBPCs

Input: maximum search width L, $I (I \le L)$ decoding paths from the previous level and PMs $p_j^{(i-1)}$, information symbol position set $\mathcal{A}$, perfect polarized channel position set $\mathcal{B}$, LLR initial value $L_1^{(1)}(y_j)_\lambda$

Output: $\min(2^m I, L)$ decoding paths and their updated PMs $p_j^{(i)}$

```
1:  for $i \in \{1, 2, \ldots, N_s\}$ do
2:      for $j \in \{1, 2, \ldots, I\}$ do
3:          Compute recursively $L_{N_s}^{(i)}(y_1^{N_s}, \hat{u}_1^{i-1}[j])_\lambda$, for $\forall\lambda \in GF(2^m)$;
4:          if $i \in \mathcal{A}^c$ then
5:              $\hat{u}_i[j] = 0$;
6:          else if $i \in \mathcal{B}$ then
7:              $\hat{u}_i[j] = \arg\min_{\lambda\in GF(2^m)} L_{N_s}^{(i)}(y_1^{N_s}, \hat{u}_1^{i-1}[j])_\lambda$;
8:          else
9:              Let $\hat{u}_i[j]$ take a value over $GF(2^m)$ in turn;
10:         end if
11:         Use equation (32) to calculate $p_j^{(i)}$;
12:     end for
13:     if $2^m I \le L$ then
14:         Return all decoding paths and $p_j^{(i)}$;
15:     else
16:         Sort $2^m I$ paths according to the size of $p_j^{(i)}$;
17:         Return $L$ decoding paths with smaller $p_j^{(i)}$;
18:     end if
19: end for
```

Algorithm 3 details the process of LLR-based PPB-SCL decoding for NBPCs. The main steps are similar to Algorithm 2, where the calculation of $\Omega_\lambda^{(i)}$ is discarded. In PPB-SCL decoding, if the current decoding symbol position $i \in \mathcal{A} \setminus \mathcal{B}$, the path list is enlarged and the PMs are determined by the penalty value in Theorem 2. Otherwise, namely $i \in \mathcal{B}$ or $i \in \mathcal{A}^c$, the

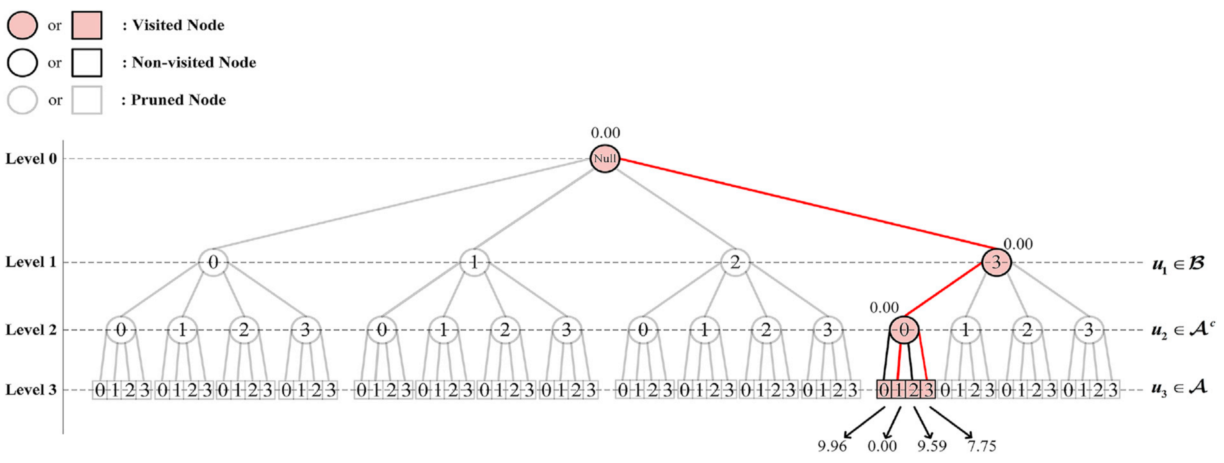


Fig. 9. PPB-SCL decoding tree over $GF(4)$ with $N_s = 4$, $K_s = 3$, $L = 2$, where the first three levels are considered.

redundant split paths are pruned and PMs keep unchanged without calculating the penalty value.

Fig. 9 shows an example of the first three levels of the PPB-SCL decoding tree with $N_s = 4$, $K_s = 3$, $L = 2$. NBPCs over $GF(4)$ are constructed by kernel H_2. In this case, we have $\mathcal{A} = \{1, 3, 4\}$ and $\mathcal{B} = \{1\}$. Due to $\mathcal{B} = \{1\}$, the subtree is pruned at the first level, leaving only one subtree that conforms to the SC decision path. Comparing Fig. 9 with Fig. 8, it is found that the number of visited nodes has been reduced from 14 to 6 only for the first three levels (except for the root node at level 0). Moreover, the survived paths up to the first three levels include the same optimal path.

5. Results and discussions

In this section, the performance of the proposed NBPCs is evaluated. In the simulations, the frozen symbol positions of NBPCs are designed by the Monte-Carlo method with $M = 10^4$ and SNR = 2 dB to trade-off the accuracy and complexity of the construction. Considering the impact of CRC length on SCL decoding performance under different code lengths, we choose two kinds of CRC lengths for medium short code lengths and long code lengths [36], respectively, where the generator polynomials of length $l_{CRC} = 8$ and $l_{CRC} = 16$ are represented as

$$g(x) = x^8 + x^7 + x^6 + x^4 + x^2 + 1 \tag{44}$$

$$g(x) = x^{16} + x^{15} + x^2 + 1 \tag{45}$$

To be specific, the BER performance and computational complexity are analyzed in Sections 5.1 and 5.2, respectively.

5.1. BER performance

First, we examine the BER performance of proposed NBPCs compared with the RS4-based NBPCs proposed in Ref. [25] and the traditional BPCs. Fig. 10(a) and Fig. 10(b) show the numerical simulation results of the proposed NBPCs over $GF(4)$ and their counterparts with different list sizes, where $N_b = 512$ and $N_b = 1024$ are considered, respectively. The kernel parameters for the proposed NBPCs are set to $\gamma = 1$ and $\delta = 1$.

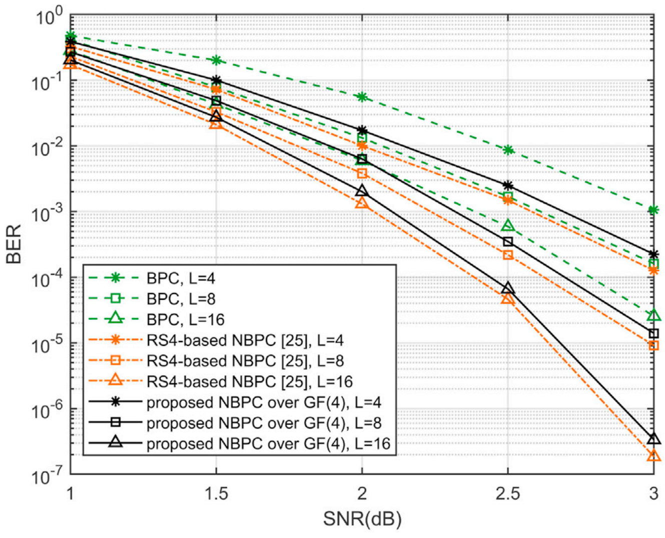
From Fig. 10, we find that the BER performance of NBPCs over $GF(4)$ outperforms the BPCs when the bit code length and list size are identical. For $N_b = 512$ and $L = 16$, a 0.36 dB gain can be observed at a BER of 10^{-4} when proposed 4-ary NBPCs are employed, compared with that of BPCs. Note that when $N_b = 1024$, the proposed NBPCs over $GF(4)$ with $L = 4$ achieve the performance of the BPCs with $L = 8$. Moreover, it can be observed that the proposed 4-ary NBPCs have a slight performance loss of 0.06 dB–0.09 dB compared with that of RS4-based NBPCs. This is because RS4-based NBPCs have a large kernel exponent by employing a high-order kernel matrix, which leads to the performance gains [24]. However, the encoding and decoding structures of RS-based NBPCs vary with the field size q, i.e., the kernel size, which cannot be implemented as easily as BPCs. With the fixed structure of non-binary 2×2 kernel, the proposed NBPCs can further obtain performance gains by increasing q, which improves flexibility and universality.

To better describe the performance potential of the proposed NBPCs, the influence of q on NBPCs is discussed, where $GF(2^m)$-based NBPCs with $m = 2, 3, 4$ are considered and CRC-8 is employed for SCL decoding. For a fair comparison, the symbol code length for $GF(4)$, $GF(8)$, and $GF(16)$ are set to 128, 64, and 64, respectively, thus the bit lengths are 256, 192, and 256, respectively. Here, the results with different values of tag element δ are exhibited, where γ is all set to 1.

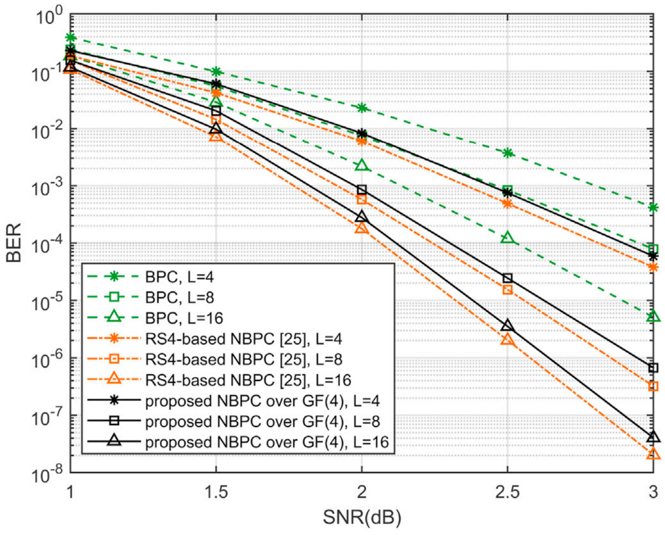
As shown in Fig. 11, NBPCs have lower BER for larger m regardless of the tag element δ, especially in the high SNR area (SNR $\geq$ 2.5 dB). Thus, a superior BER performance can be obtained for the proposed NBPC scheme by a higher field order q. It can be also seen that the BER performance of $\delta = \alpha^2$ is very close to that of $\delta = 1$ while the performance of $\delta = \alpha$ is quite slightly lower than that of $\delta = 1$. It can be said that the kernel H_2 we constructed is still polarized according to the results.

Before comparing PPB-SCL decoding with SCL decoding, we analyze the performance of PPB-SCL decoding with different code rates since a higher β is available with the increase of R according to Table 4. Simulation parameters for PPB-SCL decoding are shown in Table 5, where a reasonable β is selected on the basis of β_0. Fig. 12 exhibits the decoding performance of PPB-SCL with different R, where a $GF(4)$-based NBPC with $N_s = 256$ and 8-bit CRC is considered. It can be observed that the increase of the code rate leads to the degradation of performance. A higher code rate may achieve a higher complexity reduction since a larger β can be selected. However, PPB-SCL decoding performance with a high R is poorer than the counterpart with a low R. Therefore, a trade-off between complexity and performance is required.

Fig. 13 and Fig. 14 show the comparisons between SCL decoding and PPB-SCL decoding, where a $GF(4)$-based NBPC with $N_s = 256$ and $N_s = 1024$ is considered, respectively. All parameters for PPB-SCL decoding are set according to Table 5. Note that the result of $L = 1$

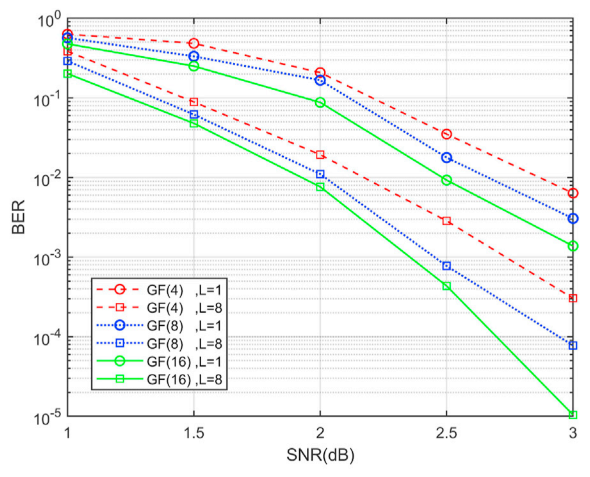


(a) $N_b = 512$ with $l_{CRC} = 8$

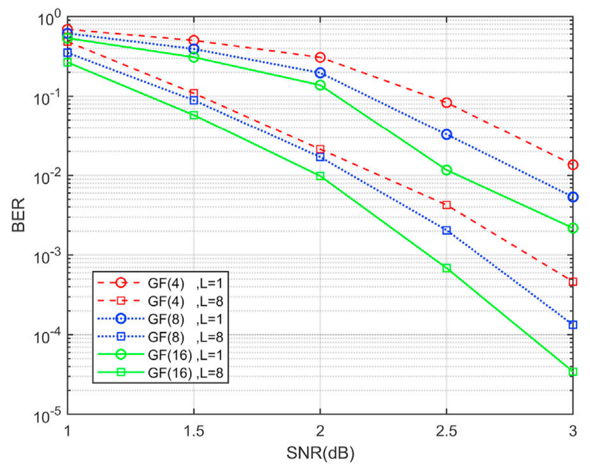


(b) $N_b = 1024$ with $l_{CRC} = 16$

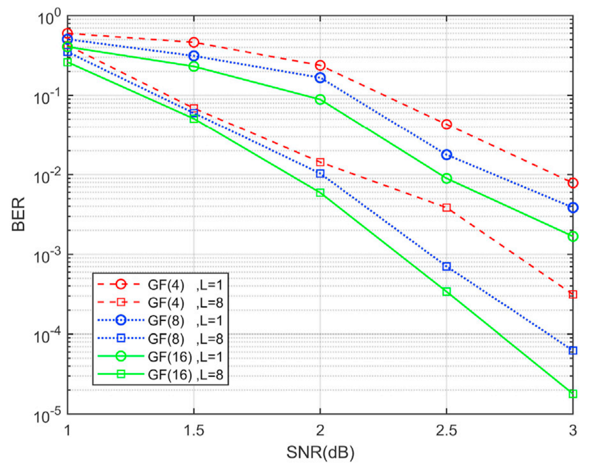
Fig. 10. Comparison of BER performance among BPCs, RS4-based NBPCs in Ref. [25] and proposed NBPCs over $GF(4)$ with different L, where $R = 0.5$ is considered.



(a) Tag-element $\delta = 1$



(b) Tag-element $\delta = \alpha$



(c) Tag-element $\delta = \alpha^2$

Fig. 11. Comparison of BER performance for NBPCs over $GF(2^m)$ with $R = 1/2$, where $m = 2, 3, 4$.

(i.e., PPB-SCL with $\beta = 1$) is also presented as a control to facilitate the analysis.

It can be seen that no matter how the code length and code rate change, the performance of PPB-SCL with $\beta = \beta_0$ is at the same level as that of traditional SCL. When $N_s = 256$, the BER of the two is exactly the same, as shown in Fig. 13. Especially in Fig. 14, we also compare the results of probability domain-based non-binary SCL decoding with $L = 8$ in Ref. [27]. The result shows that the difference between the PPB-SCL decoding with $\beta = \beta_0$ and the SCL decoding used in Ref. [27] is negligible. Hence we can conclude that the decoding performance of PPB-SCL will not be affected if path splitting is no longer performed at positions where $e_i = 0$ in Monte-Carlo simulation.

Table 5
Parameter settings.

	l_{CRC}	R	τ_0	β_0	τ	β
$N_s = 256$	8	1/3	53	0.624	62	0.729
		1/2	82	0.641	95	0.742
		2/3	112	0.655	128	0.749
$N_s = 1024$	16	1/3	280	0.821	305	0.894
		1/2	423	0.826	462	0.902
		2/3	567	0.830	611	0.895

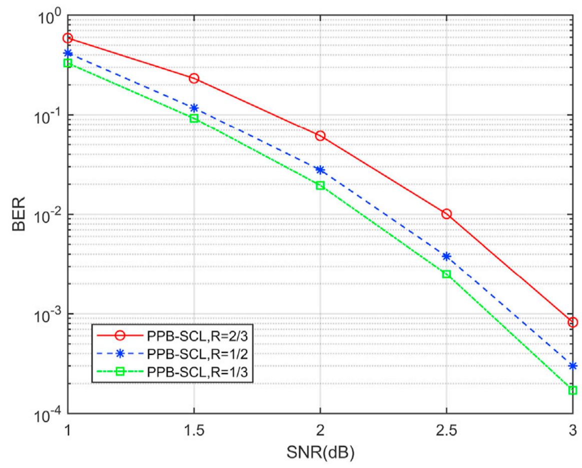


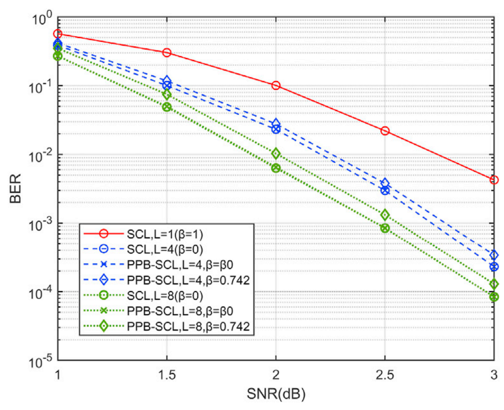
Fig. 12. BER performance of PPB-SCL decoding with different code rates, where $L = 4$.

Although the performance of PPB-SCL at $\beta = \beta_0$ fluctuates slightly with the increase of code length, we can still consider it to be equivalent to the performance of SCL decoding. When β is greater than β_0 and even approaches 1, PPB-SCL decoding gradually shows a performance degradation compared with SCL decoding. Especially when L is large, the phenomenon is more pronounced. As L increases, the number of search paths at each level of SCL decoding is more significant, which leads to more pruned paths in PPB-SCL decoding. Fig. 13(a) shows that when $N_s = 256$ and $R = 1/2$, the performance of PPB-SCL with $L = 4$ at $\beta = 0.742$ is relatively close to that of SCL. As a consequence, it is essential to choose a suitable β in the application of PPB-SCL decoding.

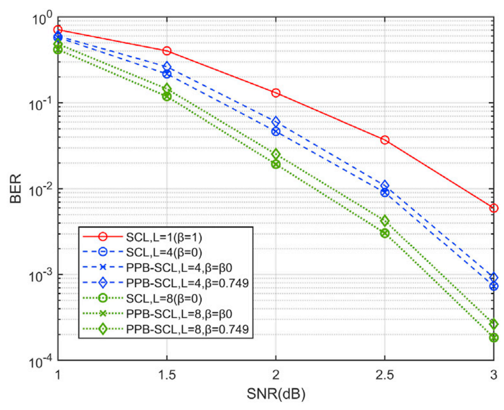
5.2. Computational complexity

Finally, we evaluate the decoding complexity of the proposed PPB-SCL scheme. In hardware design, computational complexity is an essential metric. In this paper, the computational complexity in the decoding process is derived from the basic operations used by the hardware technology. More specifically, addition, multiplication, exclusive or, and comparison operations are considered, which are denoted as ADD, MUL, XOR, and CMP, respectively.

The decoding complexity between polar codes and existing codes, e.g., LDPC codes, and turbo codes, have been compared and discussed in the literature. Polar codes with list decoding have only 8%–16% of the computational complexity of turbo codes and 22%–16% of LDPC codes [37], which varies with code rate. Here, we focus on the complexity analysis of list decoding for BPCs and the proposed NBPCs. The decoding complexity of SCL scheme can be observed by two operation parts. On the

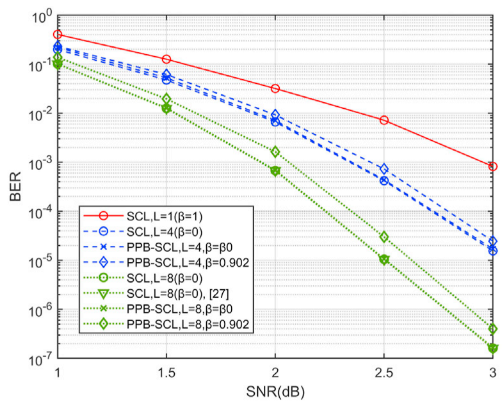


(a) $N_s = 256, R = 1/2$

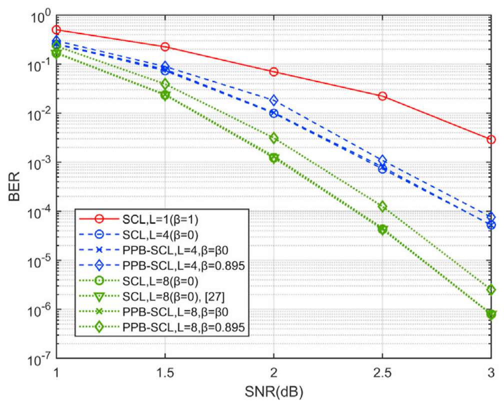


(b) $N_s = 256, R = 2/3$

Fig. 13. PPB-SCL decoding performance of NBPCs over $GF(4)$ with code length $N_s = 256$.



(a) $N_s = 1024, R = 1/2$



(b) $N_s = 1024, R = 2/3$

Fig. 14. PPB-SCL decoding performance of NBPCs over $GF(4)$ with code length $N_s = 1024$.

Table 6
A comparison of decoding complexity in terms of basic operations.

	ADD	MUL	XOR	CMP
BPC with SCL	$\frac{1}{2}LN_b\log_2 N_b + \ell$	$LN_b \log_2 N_b$	$\frac{1}{2}LN_b\log_2 N_b$	$2LN_b \log_2 N_b + \ell$
NBPC with SCL	$3qLN_s \log_2 N_s + \ell$	–	$LN_s \log_2 N_s$	$4(q-1)LN_s\log_2 N_s + \ell q$
NBPC with PPB-SCL	$(2q+1)LN_s\log_2 N_s + 2\ell$	–	$\frac{1}{2}LN_s\log_2 N_s$	$2(q-1)LN_s\log_2 N_s + \ell(q-1)$

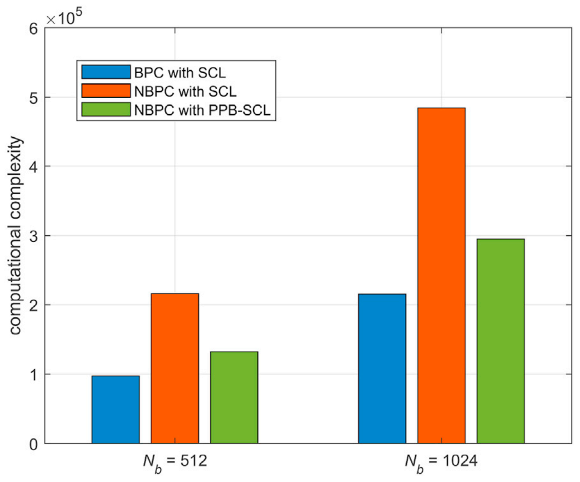


Fig. 15. Comparison of computational complexity for different algorithms, where $L = 4$ and $R = 1/2$ are exemplified.

one hand, the basic operations of LLR computation in the basic unit shown in Figs. 2 and 3 are counted. On the other hand, the basic operations of PM calculation during path splitting are also considered. Table 6 presents a comparison of the computational complexity, where ℓ denotes the number of visited nodes in the path splitting process for SCL decoding. Note that MUL operations of field elements are tallied as ADD in Table 6 as they can be represented as the addition of exponents. The LLR recursive calculation of NBPCs is done by the maximum function instead of the f function of BPCs, and thus MUL operation is not required for NBPCs.

Fig. 15 shows the decoding complexity for different schemes, where $GF(4)$-based NBPCs are considered for comparison. Note that the statistics are obtained by weighting ADD, MUL, XOR, and CMP, where MUL is weighted by 2 while the rest are weighted by 1. It can be seen that the proposed PPB-SCL scheme obtains about 40% complexity reduction of SCL decoding for NBPCs. Moreover, the computational complexity of NBPCs with SCL decoding is about 2.2 times that of BPCs. However, if the PPB-SCL decoding is implemented, it can be observed that the complexity increases by only 35%, which is a great improvement for the decoding cost of NBPCs.

Along with the BER performance analysis in Section 5.1, when $GF(4)$-based NBPCs are considered, the PPB-SCL decoding obtains a 0.3 dB–0.5 dB gain with a 35% additional complexity cost compared to BPCs. The proposed NBPC scheme with PPB-SCL decoding achieves a reasonable trade-off between computational complexity and reliability. Furthermore, NBPCs also yield significant latency gains since multiple bits are decoded simultaneously as one symbol. Therefore, the proposed NBPC shows high potential for applications in future wireless communication systems, especially ultra-reliable low-latency communications.

6. Conclusion

In this paper, we design an NBPC scheme with a general structure over $GF(2^m)$, where the popular LLR-based SCL decoding is employed flexibly without considering the field size. Specifically, a general LLR recursive function for NBPCs is presented for the arbitrary kernel size. Then, we propose a non-binary SCL decoding based on LLRs, which can be simply implemented as that of BPCs. Moreover, a PPB-SCL algorithm based on LLRs is proposed to reduce the complexity of original non-binary SCL decoding. Simulation results show that the BER performance of the proposed NBPCs significantly outperforms BPCs and has a slight loss than that of RS-based NBPCs but with high flexibility and low implementation cost. In the future, we will consider the impact of high-order non-binary kernels on performance and an adaptive Monte-Carlo simulation to further reduce the computational complexity for the construction.

Declaration of interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grant 61401407, and in part by the Fundamental Research Funds for the Central Universities under Grant CUC2019B067.

Appendix A. Supplementary data

Supplementary data related to this article can be found at https://doi.org/10.1016/j.dcan.2022.01.005.

References

[1] E. Arikan, Channel polarization: a method for constructing capacity-achieving codes for symmetric binary-input memoryless channels, IEEE Trans. Inf. Theor. 55 (7) (2009) 3051–3073.
[2] J. Wang, A. Jin, D. Shi, L. Wang, H. Shen, D. Wu, L. Hu, L. Gu, L. Lu, Y. Chen, J. Wang, Y. Saito, A. Benjebbour, Y. Kishiyama, Spectral efficiency improvement with 5G technologies: results from field tests, IEEE J. Sel. Area. Commun. 35 (8) (2017) 1867–1875.
[3] J. Wang, A. Jin, D. Shi, L. Wang, L. Hu, L. Gu, A. Benjebbour, Field trials on spectral efficiency improvement in massive MIMO systems, in: 2018 IEEE 87th Vehicular Technology Conference (VTC Spring), 2018, pp. 1–5.
[4] W. Wang, W. Liang, B. Li, L. Gu, J. Sheng, P. Qiu, J. Wang, Y. Wang, Field trial on TDD massive MIMO system with polar code, in: 2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), 2017, pp. 1–6.
[5] S. Li, Y. Deng, L. Lu, J. Liu, T. Huang, A low-latency simplified successive cancellation decoder for polar codes based on node error probability, IEEE Commun. Lett. 22 (12) (2018) 2439–2442.
[6] M. Hanif, M. Ardakani, Fast successive-cancellation decoding of polar codes: identification and decoding of new nodes, IEEE Commun. Lett. 21 (11) (2017) 2360–2363.
[7] I. Tal, A. Vardy, List decoding of polar codes, IEEE Trans. Inf. Theor. 61 (5) (2015) 2213–2226.
[8] A. Balatsoukas-Stimming, A.J. Raymond, W.J. Gross, A. Burg, Hardware architecture for list successive cancellation decoding of polar codes, IEEE Transactions on Circuits and Systems II: Express Briefs 61 (8) (2014) 609–613.
[9] J. Lin, Z. Yan, An efficient list decoder architecture for polar codes, IEEE Trans. Very Large Scale Integr. Syst. 23 (11) (2015) 2508–2518.
[10] Q. Zhang, A. Liu, X. Pan, K. Pan, CRC code design for list decoding of polar codes, IEEE Commun. Lett. 21 (6) (2017) 1229–1232.
[11] C. Yang, M. Zhan, Y. Deng, M. Wang, X.H. Luo, J. Zeng, Error-correcting performance comparison for polar codes, LDPC codes and convolutional codes in high-performance wireless, in: 2019 6th International Conference on Information, Cybernetics, and Computational Social Systems (ICCSS), IEEE, 2019, pp. 258–262.
[12] P. Popovski, C. Stefanovic, J.J. Nielsen, E. de Carvalho, M. Angjelichinoski, K.F. Trillingsgaard, A. Bana, Wireless access in ultra-reliable low-latency communication (URLLC), IEEE Trans. Commun. 67 (8) (2019) 5783–5801.
[13] L. Chettri, R. Bera, A comprehensive survey on internet of things (IoT) toward 5G wireless systems, IEEE Internet Things J. 7 (1) (2020) 16–32.
[14] X. Ge, L. Pan, Q. Li, G. Mao, S. Tu, Multipath cooperative communications networks for augmented and virtual reality transmission, IEEE Trans. Multimed. 19 (10) (2017) 2345–2358.
[15] B. Yuan, K.K. Parhi, Low-latency successive-cancellation list decoders for polar codes with multibit decision, IEEE Trans. Very Large Scale Integr. Syst. 23 (10) (2015) 2268–2280.
[16] B. Yuan, K.K. Parhi, LLR-based successive-cancellation list decoder for polar codes with multibit decision, IEEE Transactions on Circuits and Systems II: Express Briefs 64 (1) (2017) 21–25.
[17] S. Cao, H. Zheng, T. Lin, S. Zhang, S. Xu, An unfolded pipelined polar decoder with hybrid number representations for multi-user MIMO systems, IEEE Transactions on Circuits and Systems II: Express Briefs 67 (11) (2020) 2472–2476.
[18] T. Hong, T. Tang, X. Dong, R. Liu, W. Zhao, Future 5G mmwave TV service with fast list decoding of polar codes, IEEE Trans. Broadcast. 66 (2) (2020) 525–533.
[19] E. Sasoglu, E. Telatar, E. Arikan, Polarization for arbitrary discrete memoryless channels, in: 2009 IEEE Information Theory Workshop, IEEE, 2009, pp. 144–148.
[20] R. Mori, T. Tanaka, Channel polarization on q-ary discrete memoryless channels by arbitrary kernels, in: 2010 IEEE International Symposium on Information Theory, IEEE, 2010, pp. 894–898.
[21] M. Chiu, Non-binary polar codes with channel symbol permutations, in: 2014 International Symposium on Information Theory and its Applications, IEEE, 2014, pp. 433–437.
[22] T.C. Gulcu, M. Ye, A. Barg, Construction of polar codes for arbitrary discrete memoryless channels, IEEE Trans. Inf. Theor. 64 (1) (2018) 309–321.
[23] W. Park, A. Barg, Polar codes for q-ary channels, $q = 2^r$, IEEE Trans. Inf. Theor. 59 (2) (2013) 955–969.
[24] R. Mori, T. Tanaka, Non-binary polar codes using Reed-Solomon codes and algebraic geometry codes, in: 2010 IEEE Information Theory Workshop, 2010, pp. 1–5.
[25] N. Cheng, R. Zhang, Y. Ge, W. Shi, Q. Zhang, X.S. Shen, Encoder and list decoder of Reed-Solomon kernel based polar codes, in: 2016 8th International Conference on Wireless Communications Signal Processing (WCSP), 2016, pp. 1–6.
[26] S.C. Byun, G. Kim, W.J. Kim, H.Y. Song, A construction of non-binary polar codes with 4 by 4 kernels, in: 2019 Ninth International Workshop on Signal Design and its Applications in Communications (IWSDA), 2019, pp. 1–5.
[27] P. Yuan, F. Steiner, Construction and decoding algorithms for polar codes based on 2×2 non-binary kernels, in: 2018 IEEE 10th International Symposium on Turbo Codes Iterative Information Processing (ISTC), 2018, pp. 1–5.
[28] S. Cayci, T. Koike-Akino, Y. Wang, Nonbinary polar coding for multilevel modulation, in: 2019 Optical Fiber Communications Conference and Exhibition (OFC), 2019, pp. 1–3.
[29] A. Balatsoukas-Stimming, M.B. Parizi, A. Burg, LLR-based successive cancellation list decoding of polar codes, IEEE Trans. Signal Process. 63 (19) (2015) 5165–5179.
[30] J. Liu, H. Wang, C. Shen, J. Lee, Low-complexity LDPC decoder for 5G URLLC, in: 2018 IEEE Asia Pacific Conference on Postgraduate Research in Microelectronics and Electronics (PrimeAsia), 2018, pp. 43–46.
[31] M. Dhuheir, S. Ozturk, Polar codes analysis of 5G systems, in: 2018 6th International Conference on Control Engineering Information Technology (CEIT), 2018, pp. 1–6.
[32] P. Trifonov, Efficient design and decoding of polar codes, IEEE Trans. Commun. 60 (11) (2012) 3221–3227.
[33] D. Wu, Y. Li, Y. Sun, Construction and block error rate analysis of polar codes over AWGN channel based on Gaussian approximation, IEEE Commun. Lett. 18 (7) (2014) 1099–1102.
[34] B. Li, H. Shen, D. Tse, W. Tong, Low-latency polar codes via hybrid decoding, in: 2014 8th International Symposium on Turbo Codes and Iterative Information Processing (ISTC), 2014, pp. 223–227.
[35] S. Choi, H. Yoo, Hybrid decoding for polar codes, in: 2018 International SoC Design Conference (ISOCC), 2018, pp. 121–122.
[36] Y. Shen, L. Li, J. Yang, X. Tan, Z. Zhang, X. You, C. Zhang, Low-latency segmented list-pruning software polar list decoder, IEEE Trans. Veh. Technol. 69 (4) (2020) 3575–3589.
[37] H. Gamage, N. Rajatheva, M. Latva-aho, Channel coding for enhanced mobile broadband communication in 5G systems, in: 2017 European Conference on Networks and Communications (EuCNC), 2017, pp. 1–6.