Projeto de Graduação

28 de Junho de 2021

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

Decodificação Belief Propagation Orientada a Q-Learning para Códigos Polares

Lucas Marques de Oliveira

PUC RIO

Projeto de Graduação

DEPARTAMENTO
DE ENGENHARIA
ELÉTRICA

Decodificação Belief Propagation Orientada a Q-Learning para Códigos Polares

Aluno: Lucas Marques de Oliveira

Orientador: Rodrigo C. De Lamare
Coorientador: Robert M. Oliveira

Trabalho apresentado como requisito parcial à conclusão do curso de Engenharia Elétrica na Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brasil.

Agradecimentos

À Deus, fonte de vida.

Para meus pais, José Milton e Severina de Lourdes, por todo amor ,carinho e suporte.

Para meu professor Rodrigo C. De Lamare e ao doutorando Robert de Oliveira por acreditarem no meu potencial.

Para meus amigos do CETUC que me ajudaram e apoiaram nos meus momentos mais adversos.

Para os meus amigos no Labaroratório de Sistemas que me guiaram e ensinaram a cada dia que trabalhamos juntos.

Decodificação Belief Propagation Orientada a Q-Learning para Códigos Polares

Resumo

O código polar tem se provado como um forte benchmark no que se diz respeito à codificação de canal para próxima geração de comunicação móvel, o 5G. Introduzidos por Arikan [1], os códigos polares são os primeiros tipos de códigos de canal na qual sua capacidade pode ser atingida através de uma prova teórica, sob as condições de comprimento de código infinito e decodificação de cancelamento sucessivo (SC). Entretanto, códigos de comprimento finito apresentam um baixo desempenho em termos de taxa de erro sob o decodificador SC, além de apresentarem uma alta latência devido à natureza sequencial do decodificador, na qual os bits de informação são decodificados serialmente.

Como a decodificação SC para comprimentos de códigos finitos é sub-ótima, Tal e Vardy [2] propuseram um decodificador de cancelamento sucessivo em lista (SCL) no qual se aproxima do limite de máxima verossimilhança (ML) para um tamanho de lista suficientemente grande ao custo de uma maior complexidade. Além disso, códigos de verificação de redundância cíclica (CRC) [3] podem ser facilmente introduzidos a fim de aprimorarem o algoritmo SCL por meio do aumento de sua distância mínima. Esta combinação torna os códigos polares um poderoso esquema de codificação, apesar de apresentarem uma alta complexidade de decodificação SCL e uma natureza de decodificação inerentemente serial. Dessa forma, diversas variantes do decodificador foram propostas como forma de reduzir a complexidade computacional [4], [5] e [6].

Como alternativa à natureza serial do decodificador baseado em SC, Arikan propôs um algoritmo de decodificação iterativo com alto potencial de paralelismo baseado na propagação de crenças (BP) [7] sob o diagrama de codificação dos códigos polares. No entanto, apesar do algoritmo superar a decodificação SC, ser um forte candidato em aplicações que demandam uma alta taxa de dados e mais adequado para implementações em hardware, sua performance não é ainda equiparável ao decodificador SCL com CRC.

Dessa forma, uma série de esforços têm sido feitos na tentativa de melhorar a performance do decodificador. Em [8] e [9], foi mostrado que os códigos polares de comprimento finito sob decodificação BP podem ser aprimorados quando canais semi polarizados são adicionalmente protegidos pelos nós de verificação ou através de um código polar aumentado [10]. Infelizmente, essas abordagens ainda apresentam uma performance inferior quando comparada à decoficação SCL. Além disso, elas exigem uma estrutura de código ajustada e, portanto, não são compatíveis com o código polar consolidado na literatura.

Nosso projeto consiste, portanto, em desenvolver um novo decodificador BP, baseado em Q-Learning, que preserva a estrutura do código polar, e que obtém uma performance consistentemente superior aos decodificadores BP e SC convencionais.

Palavras-chave: Decodificador BP para Códigos Polares, Q-Learning, Decodificador BP orientada a Q-learning para Códigos Polares

Q-Learning Driven BP Decoding for Polar Codes

Abstract

Polar codes have been proved as a strong benchmark when it comes to channel polarization of the next generation of mobile communication, 5G. Introduced by Arikan, polar codes are the first type of channel codes in which their capacity can be achieved by means of theoretical proof, assuming an infinite code length and successive-cancellation decoder (SC). However, finite-length codes have a lower performance in terms of error rate under the SC decoder, in addition to presenting a higher latency due to the sequential nature of the decoder, in which the information bits are decoded serially.

Considering that the SC decoder for finite length codes is sub-optimal, Tal and Vardy proposed a list-based Successive Cancellation Decoder (SCL) in which it approaches the maximum likelihood (ML) performance for a list size large enough at the cost of greater complexity. Furthermore, Cyclic Redundancy Check Codes (CRC) can be effortlessly implemented in order to improve the SCL algorithm by increasing its minimum distance. This combination makes polar codes a powerful decoding scheme, despite their high complexity and a serial inherent nature. Therefore, several variants of the decoder have been proposed in order to reduce the computational complexity.

As an alternative to the serial nature of the SC decoder, Arikan proposed an iterative decoder algorithm with great potential for parallelism based on Belief Propagation (BP) under the codification diagram of the polar codes. Nevertheless, despite the algorithm outperforming the SC decoder, being a strong candidate in applications which demand a high data rate and more suitable to hardware implementations, your performance is not comparable to the CRC-aided SCL decoder.

Therefore, several efforts have been made to enhance the decoder performance. It has been shown that finite-length polar codes under BP decoding can be improved when semi-polarized channels are additionally protected by check nodes, or with an augmented polar code. Unfortunately, these approaches have inferior performance when compared to SCL decoding. Besides, they require an adjusted code structure and are thus not compatible with the standardized polar code in the literature.

Therefore, our project consists of developing a new BP decoder, based on Q-Learning, which maintains the polar code structure, and obtains a performance consistently superior to the standard BP and SC decoders.

Keywords: BP Decoder for Polar Codes, Q-Learning, Q-Learning Driven BP decoding for Polar Codes

Lista de Figuras

Lista de Tabelas

Sumário

1 Introdução

Os códigos polares são códigos corretores de erros capazes de atingir a capacidade de canais simétricos, sem memória, de entrada binária, possibilitando a transmissão de informação na maior taxa possível. Além disso, as operações de codificação e decodificação podem ser realizadas com baixa complexidade, graças à natureza recursiva na qual o código é construído.

Formalmente, um código polar é completamente definido por uma tupla-4 (N,R,A,$\boldsymbol{u}_{\mathbb{A}^c}$) no qual:

- N é o comprimento do bloco, N = 2^n, n $\geq$ 0, isto é, o número de bits transmitido pelo canal;
- R é a taxa, R $\in [0,1]$, isto é, a quantidade de informação contida em um bit;
- $\mathbb{A}$ é o conjunto de informação, $\mathbb{A} \subset \{1,...,N\}$, isto é, o conjunto de posições na qual contém bits de informação;
- $\boldsymbol{u}_{\mathbb{A}^c}$ são os bits congelados, $\boldsymbol{u}_{\mathbb{A}^c} \in \{0,1\}^{\mathrm{N(1-R)}}$, isto é, bits que possuem um valor fixo a priori, compartilhado entre o codificador e o decodificador.

A figura 1 mostra o diagrama de bloco de um sistema de codificação polar.

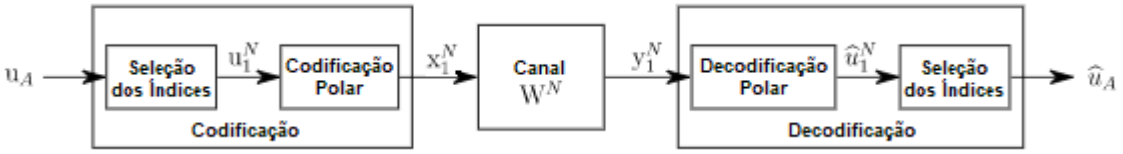


Figura 1: Esquema de Codificação Polar

Neste sistema, a mensagem é transmitida da seguinte forma:

- Através de um processo denominado polarização, escolhemos o conjunto de informação $\mathbb{A}$, cuja escolha depende de um particular canal sobre o qual a transmissão ocorre.
- Define-se os N·R bits de informação contidas no vetor $\boldsymbol{u}_{\mathbb{A}}$.
- N·(1-R) bits congelados contidos no vetor $\boldsymbol{u}_{\mathbb{A}^c}$ são fixos.
- $\boldsymbol{u}_{\mathbb{A}}$ e $\boldsymbol{u}_{\mathbb{A}^c}$ são combinados de forma a obter $\boldsymbol{u}_1^{\mathrm{N}}$.
- $\boldsymbol{u}_1^{\mathrm{N}}$ é codificado em $\boldsymbol{x}_1^{\mathrm{N}}$ usando a codificação recursiva polar.
- $\boldsymbol{x}_1^{\mathrm{N}}$ é transmitida no canal W^N, que corresponde aos N usos do canal. $\boldsymbol{y}_1^{\mathrm{N}}$ é recebido.
- Através de $\boldsymbol{y}_1^{\mathrm{N}}$, o decodificador polar produz $\hat{\boldsymbol{u}}_1^{\mathrm{N}}$, que é uma estimativa de $\boldsymbol{u}_1^{\mathrm{N}}$.
- Finalmente, somente as componentes de $\hat{\boldsymbol{u}}_1^{\mathrm{N}}$ correspondentes aos bits de informação são mantidos, gerando $\hat{\boldsymbol{u}}_{\mathrm{A}}$.

2 Fundamentos dos Códigos Polares

a Preliminares

Seja $W : \mathcal{X} \to \mathcal{Y}$ um canal sem memória discreto binário (B-DMC) com alfabeto de entrada $\mathcal{X}$, alfabeto de saída $\mathcal{Y}$, e probabilidade de transição $W(y|x)$, $x \in \mathcal{X}$, $y \in \mathcal{Y}$. O alfabeto de entrada $\mathcal{X}$ será sempre {0,1}. Seja ainda W^N o canal correspondente ao N usos de W, assim, $W^N : \mathcal{X}^N \to \mathcal{Y}^N$ com $W^N(\boldsymbol{y_1^N}|\boldsymbol{x_1^N}) = \prod_{i=1}^{N} W(y_i|x_i)$.

Dado um B-DMC W, dois parâmetros importantes do canal, a saber, a capacidade simétrica e o parametro de Bhattacharyya, medem a taxa e a confiabilidade do canal, respectivamente. A capacidade simétrica é definida como:

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

sendo $I(W)$ a maior taxa na qual uma comunicação confiável é garantida através de W.

Já o parâmetro Bhattacharyya, por sua vez, é dado por:

$$Z(W) \triangleq \sum_{y\in\mathcal{Y}} \sqrt{W(y|0)W(y|1)} \tag{2}$$

sendo $Z(W)$ o limite superior da probabilidade de erro de decisão com máxima verossimilhança (ML) quando W é usado somente uma vez para transmitir 0 ou 1.

Além disso, pode ser provado que o relacionamento entre esses parâmetros é dado por:

$$I(W) \geq \log \frac{2}{1 + Z(W)} \tag{3}$$

$$I(W) \leq \sqrt{1 - Z(W)^2} \tag{4}$$

Por fim, é importante salientar que para os canais simétricos, ou seja, aqueles sobre os quais existe uma permutação π no alfabeto de saída $\mathcal{Y}$ tal que i) $\pi^{-1} = \pi$ e ii) $W(y|1) = W(\pi(y)|0)$ para todo $y \in \mathcal{Y}$, a capacidade simétrica I(W) é igual a capacidade de Shannon. São exemplos de canais simétricos: canal simétrico binário (BSC) e canal de eliminação binária (BEC).

b Polarização

Esse fenômeno é uma operação no qual se fabrica, a partir de N cópias independentes de um dado B-DMC, um segundo conjunto de canais $\{W_N^{(i)} : 1 \leq i \leq N\}$ que apresentam um efeito de polarização, de maneira que, conforme N se torna maior, os termos de capacidade simétrica $I(W_N^{(i)})$ tendem para 0 ou 1, exceto para uma seleta fração de índice i. Essa operação é construído por meio das fases de combinação e divisão de canal.

1 Combinação de canal

Essa etapa combina cópias de um dado B-DMC W de maneira recursiva para produzir um vetor de canais $W_N : \mathcal{X}^N \to \mathcal{Y}^N$, onde N = 2^n, n $\geq$ 0. A recursão se inicia no estágio 0 quando n=0, com somente uma cópia de W e definimos $W_1 \triangleq W$. O primeiro estágio, n=1, da recursão combina duas cópias independentes de W_1, conforme mostrado na figura 2 e obtém-se $W_2 : \mathcal{X}^2 \to \mathcal{Y}^2$ com probabilidades de transição:

$$W_2(y_1, y_2|u_1, u_2) = W(y_1|u_1 \oplus u_2)W(y_2|u_2) \tag{5}$$

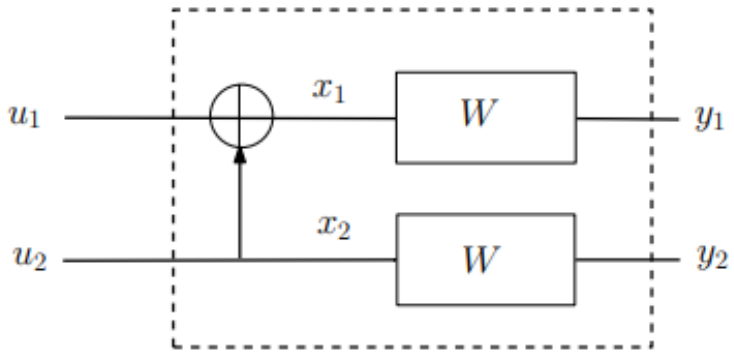


Figura 2: Canal W_2. Retirado de [11]

sendo $\oplus$ a soma módulo-2 das componentes do vetor $\boldsymbol{u_1^2}$.

O próximo estágio de recursão é mostrado na figura 3 na qual duas cópias independentes de W_2 são combinadas para criar um canal $W_4 : \mathcal{X}^4 \rightarrow \mathcal{Y}^4$ com probabilidades de transição $W_4(\boldsymbol{y_1^4}|\boldsymbol{u_1^4}) = W_2(\boldsymbol{y_1^2}|u_1 \oplus u_2, u_3 \oplus u_4)W_2(\boldsymbol{y_3^4}|u_2, u_4)$.

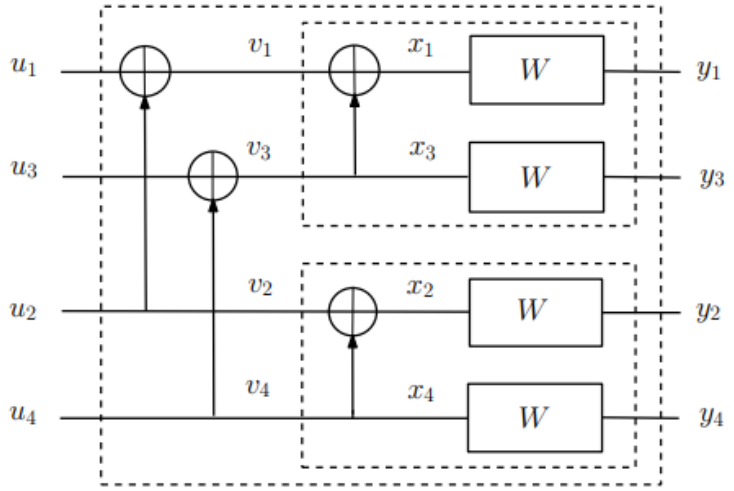


Figura 3: Canal W_4. Retirado de [11]

De maneira simular, W_N pode ser construido recursivamente a partir de $W_{\frac{N}{2}}$,$W_{\frac{N}{4}}$,..., W_2,W em l etapas, onde n=2^l.

2 Divisão de Canais

Tendo sintetizado o vetor de canais W_N em W^N, o próximo passo da polarização de canal é dividir novamente W_N em um conjunto de N canais de entrada binária indexados por i, $W_N^{(i)} : \mathcal{X} \rightarrow \mathcal{Y}^N$ x $\mathcal{X}^{i-1}$, 1 $\leq$ i $\leq$ N, definidos pelas probabilidades de transição: ´

$$W_N^{(i)}(\boldsymbol{y_1^N}, \boldsymbol{u_1^{(i-1)}}|u_i) = \sum_{\boldsymbol{u_{i+1}^N} \in \mathcal{X}^{N-i}} \frac{1}{2^{N-1}} W_N(\boldsymbol{y_1^N}|\boldsymbol{u_1^N}) \tag{6}$$

no qual $(\boldsymbol{y_1^N}, \boldsymbol{u_1^{i-1}})$ denota a saída de $W_N^{(i)}$ e u_i sua entrada.

Assim, para N=2, temos que:

$$W_2^{(1)}(y_1^2|u_1) = \sum_{u_2^2 \in X^1} \frac{1}{2^{2-1}} W(y_1|x_1)W(y_2|x_2) \tag{7a}$$

$$= \sum_{u_2} \frac{1}{2} W(y_1|u_1 \oplus u_2)W(y_2|u_2) \tag{7b}$$

$$W_2^{(2)}(y_1^2, u_1|u_2) = \frac{1}{2} W(y_1|x_1)W(y_2|x_2) \tag{7c}$$

$$= \frac{1}{2} W(y_1|u_1 \oplus u_2)W(y_2|u_2) \tag{7d}$$

A partir dessas operações, pode-se mostrar que para qualquer B-DMC W, os canais $W_N^{(i)}$ se polarizam de maneira que, para qualquer $\delta \in (0,1)$ fixo, quando N tende a infinito através de uma potência de 2, a fração de índices i $\in$ {1,...,N} no qual $I(W_N^{(i)}) \in (1-\delta, 1]$ tende a $I(W)$ e a fração no qual $I(W_N^{(i)}) \in [0, \delta)$ tende a $1 - I(W)$. Esse teorema é mostrado na seção IV de [1].

O efeito da polarização para o caso em que W é um BEC com probabilidade de erro $\epsilon = 0.5$ é mostrado na figura 4. Os valores para $I(W_N^{(i)})$ são computados a partir das seguintes relações recursivas:

$$I(W_N^{(2i-1)}) = I(W_{N/2}^{(i)})^2 \tag{8a}$$

$$I(W_N^{(2i)}) = 2I(W_{N/2}^{(i)}) - I(W_{N/2}^{(i)})^2 \tag{8b}$$

com $I(W_1^{(1)}) = 1 - \epsilon$. Essa recursão é válida apenas para BECs e é provada na seção III de [1].

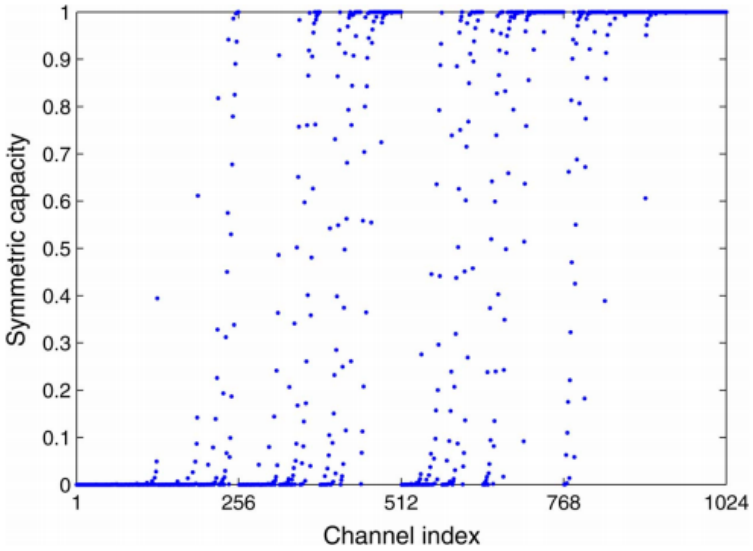


Figura 4: Efeito da Polarização: $I(W_N^{(i)})$ para N=2^{10} considerando um BEC com $\epsilon = 0.5$. Retirado de [1].

A partir da figura 4, observe que $I(W_N^{(i)})$ tende para 0 quanto menor for i, e próximo de 1 quanto maior for i. Entretanto, $I(W_N^{(i)})$ apresenta um comportamento errático para um intervalo intermediário de i.

Por fim, em relação ao parâmetro de Bhattacharyya, a relação recursiva é determinada por:

$$Z(W_N^{(2i-1)}) = 2Z(W_{N/2}^{(i)}) - Z(W_{N/2}^{(i)})^2 \tag{9a}$$

$$Z(W_N^{(2i)}) = Z(W_{N/2}^{(i)})^2 \tag{9b}$$

com $Z(W_1^{(1)}) = \epsilon$.

3 Construção do Código

A partir do efeito da polarização, constrói-se códigos de comprimento 2^n, onde $n = \log N$, que atingem a capacidade do canal simétrico, $I(W)$, através de um método denominado codificação polar. Sua ideia básica consiste em criar um sistema de codificação em que os k canais com menores valores de $Z(W_N^{(i)})$ dentro do vetor de canais, $W_N^{(i)}$, são selecionados para transmitirem bits de informação. Portanto, definem-se os conjuntos $\mathbb{A}$, com os k menores valores de $Z(W_N^{(i)})$, e $\mathbb{A}^{\mathbb{C}}$ com os demais, responsáveis por transmitirem bits congelados. A construção de códigos polares pode ser generalizado para qualquer comprimento de código a partir de técnicas de shortening [12], puncturing [13] e usando polarização de canal não uniforme [14].

Vale ressaltar ainda que os códigos polares são códigos específicos do canal, ou seja, parâmetros como σ para AWGN e ϵ para BEC são entradas para o método de construção do código. Considerando W um BEC, a construção do código polar para N=8 e R=$\frac{1}{2}$ com probabilidade de erro $\epsilon = 0.3$ é mostrada na figura 5.

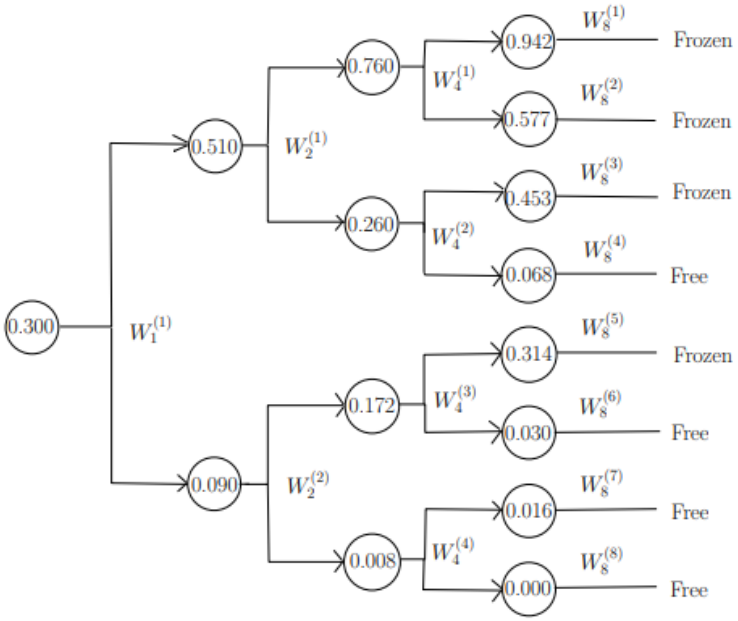


Figura 5: Construção do código para BEC com N=8, R=$\frac{1}{2}$ e ϵ=0.3. Retirado de [11]

.

Inicialmente, a confiabilidade do primeiro canal $W_1^{(1)}$ é 0.3. Em seguida, $W_2^{(1)}$ é calculado como $Z(W_2^{(1)}) = 2Z(W_1^{(1)}) - Z(W_1^{(1)})^2$, sendo $Z(W_N^{(i)})$ a probabilidade de apagamento do i-ésimo canal. Em paralelo, o segundo canal é calculado como $Z(W_2^{(2)}) = Z(W_1^{(1)})^2$. De maneira análoga, os demais

valores de confiabilidade de canais podem ser determinados recursivamente a partir das equações (9).

Ao final da etapa n, sendo $n = \log N$ (neste caso, n=3), tem-se a probabilidade de apagamento de todos os N=8 canais. Nesta etapa, os canais que possuem os 4 menores valores de $Z(W_N^{(i)})$ são selecionados como livres, e os demais como congelados.

c Codificação

Códigos polares podem ser codificados por meio de um simples mapeamento linear. Para um comprimento de código N=2^n com $n \geq 1$, a matriz geradora, G_N, é definida [1] como

$$\boldsymbol{G_N} = \boldsymbol{B_N F^{\oplus n}} \tag{10}$$

sendo $\boldsymbol{B_N}$ uma matriz de reversão de bits e $F^{\oplus n}$ a n-ésima potência de Kronecker da matriz $\boldsymbol{F} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$. A título de exemplo, mostraremos a construção da matriz de Kronecker $\boldsymbol{F}^{\oplus 3}$.

$$\mathbf{F}^{\oplus 3} = \begin{bmatrix} \mathbf{F}^{\oplus 2} & 0 \\ \mathbf{F}^{\oplus 2} & \mathbf{F}^{\oplus 2} \end{bmatrix} = \begin{bmatrix} \mathbf{F} & 0 & 0 & 0 \\ \mathbf{F} & \mathbf{F} & 0 & 0 \\ \mathbf{F} & 0 & \mathbf{F} & 0 \\ \mathbf{F} & \mathbf{F} & \mathbf{F} & \mathbf{F} \end{bmatrix} \tag{11}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{12}$$

Ademais, como a permutação de bits, dada pela matriz $\boldsymbol{B_N}$, serve apenas para reordenar os índices de $(x_1, ..., x_N)$ sem afetar as propriedades dos códigos polares, não trataremos essa reversão ao longo deste projeto por uma questão de simplicidade.

O codificador polar de comprimento N funciona como um Sistema Linear Invariante no Tempo (SLIT) cujo vetor de entrada é $\boldsymbol{u_1^N}$ e o vetor de saída é $\boldsymbol{x_1^N}$. O mapeamento $\boldsymbol{u_1^N} \rightarrow \boldsymbol{x_1^N}$ é linear sob o corpo binário finito, GF(2), de modo que $\boldsymbol{x_1^N} = \boldsymbol{u_1^N G_N}$. Além disso, as linhas de G_N são linearmente independentes, constituindo assim uma base para o espaço de código. Em termos de complexidade, um codificador de n bits inclui $(\frac{N}{2} \log_2 N)$ operações XOR.

A representação do diagrama da estrutura de um codificador de 8 bits é mostrado na figura 6, onde o símbolo $\oplus$ representa uma operação binária XOR. Note que a qualidade do canal define quais canais transmitirão dados e quais estarão congelados.

Como resultado, as seguintes equações são obtidas na saída do codificador de 8 bits.

$$\begin{aligned} x_1 &= u_1 \oplus u_1 \oplus u_2 \oplus u_3 \oplus u_4 \oplus u_5 \oplus u_6 \oplus u_7 \oplus u_8 \\ x_2 &= u_5 \oplus u_6 \oplus u_7 \oplus u_8 \\ x_3 &= u_3 \oplus u_4 \oplus u_7 \oplus u_8 \\ x_4 &= u_7 \oplus u_8 \\ x_5 &= u_2 \oplus u_4 \oplus u_6 \oplus u_8 \\ x_6 &= u_6 \oplus u_8 \\ x_7 &= u_4 \oplus u_8 \\ x_8 &= u_8 \end{aligned} \tag{13}$$

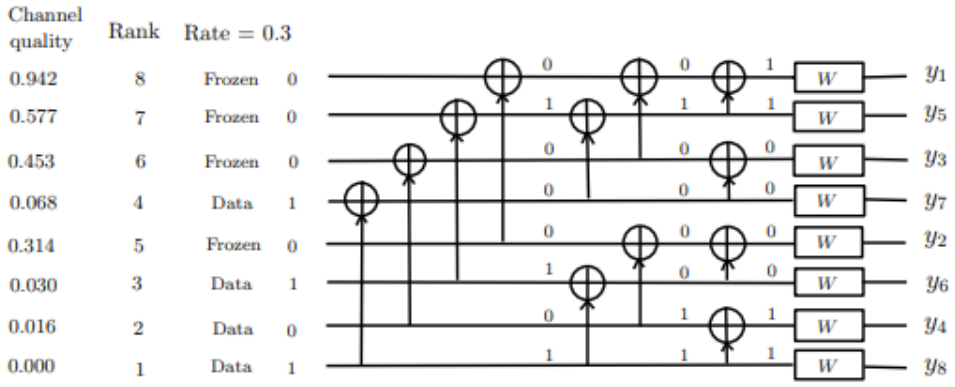


Figura 6: Processo de codificação para um código polar N=8, k=4, e BEC com ϵ=0.3. Retirado de [11].

1 Codificação Sistemática

Esquemas de codificação para códigos polares podem ser não sistemáticos, conforme mostrado na figura 6, ou sistemáticos, conforme discutido em [15]. Em geral, os códigos sistemáticos oferecem um melhor Bit Error Rate (BER) do que sua contraparte não sistemática devido ao aumento na distância média efetiva de Hamming na palavra código gerada [16]. Além disso, eles permitem o uso de técnicas de adaptação de taxas com baixa complexidade tal como o método de encurtamento de código proposto em [17]. Codificação sistemática de baixa complexidade flexível de códigos polares é discutida em detalhes em [18].

O esquema de codificação proposto anteriormente, em que o vetor de entrada $\boldsymbol{u}_1^N$ é multiplicado pela matriz geradora $\boldsymbol{G}$, resulta em uma palavra código polar não sistemática $\boldsymbol{x}_1^N$, no qual os bits de informação não aparecem explicitamente na palavra código. Em contrapartida, no caso sistemático, os bits de informação e os congelados permanecem inalterados.

O esquema de codificação sistemático para códigos polares foi introduzido em [15] e apresenta uma performance superior em termos BER quando comparada com o caso não sistemático. Seu algoritmo se inicia com os bits de informação, denotados como $\boldsymbol{x}_{\mathbb{A}}$, e calculamos os bits congelados, de paridade, de acordo com

$$\boldsymbol{x}_{\mathbb{A}^c} = \boldsymbol{x}_{\mathbb{A}}(\boldsymbol{G}_{\mathbb{A}\mathbb{A}})^{-1}G_{\mathbb{A}\mathbb{A}} \tag{14}$$

em que $\boldsymbol{G}_{\mathbb{A}\mathbb{A}}$ é a submatriz de $\boldsymbol{G}$ cujas linhas e colunas correspondem aos índices dos bits de informação, e $\boldsymbol{G}_{\boldsymbol{A}\boldsymbol{A}^c}$ corresponde a submatriz de $\boldsymbol{G}$ em que as linhas e colunas diz respeito aos bits de informação e congelados, respectivamente. Os bits de informação e de paridade são, então, combinados em uma única palavra código sistemática dispostos nas posições de bits de informação e congelados, respectivamente.

Vale ressaltar ainda que quando a natureza recursiva da matriz geradora $\boldsymbol{G}$ é explorada, a implementação serial desse algoritmo apresenta uma complexidade de $O(N \log N)$, sendo N o comprimento do código. A figura 7 mostra um exemplo de um esquema de codificação sistemática de baixa complexidade proposta em [18], compreendida por duas passagens pelo esquema de codificação não sistemático com uma operação de mascaramento de bits no meio. Para um código polar N=8 e R=$\frac{1}{2}$, um vetor de 8 bits $\boldsymbol{u}_1^N = (0, 0, 0, a_0, 0, a_1, a_2, a_3)$, onde $(a_0, ..., a_3)$ representam os 4 bits de informação, são as entradas do primeiro codificador não sistemático à esquerda. Em seguida, usando o mascaramento de bits, as posições correspondentes aos bits congelados são setados como 0 antes de realizar uma nova propagação do vetor atualizado através do segundo codificador não sistemático.

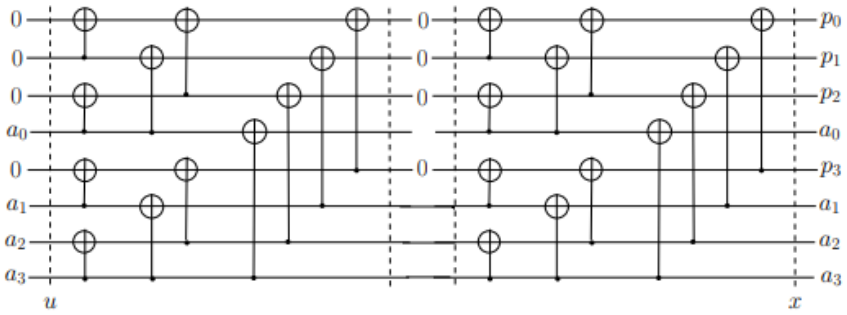


Figura 7: Codificação Sistemática de um código polar N=8, k=4. Retirado de [11].

Com isso, a codificação sistemática pode ser sintetizada através da seguinte expressão:

$$\boldsymbol{x'} = \boldsymbol{u}\boldsymbol{G_N}\boldsymbol{p}\boldsymbol{G_N} \tag{15}$$

em que $\boldsymbol{p} = [u_i(i \in \mathbb{A}^{\mathbb{C}}) = 0, u_i(i \notin \mathbb{A}^{\mathbb{C}}) = 1]$.

Por fim, a figura 8 mostra a performance dos códigos polares para N=1024 e R=$\frac{1}{2}$ comparando esquemas de codificações sistemático e não sistemático. Os resultados mostram que os códigos sistemáticos apresentam um ganho significativo quando comparado com os não sistemáticos, apesar de não haver uma prova teórica na literatura justificando tal ganho.

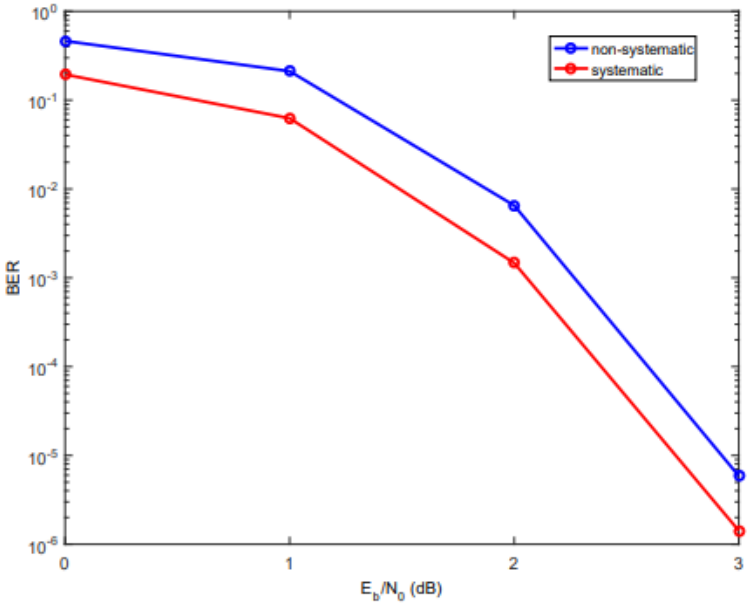


Figura 8: Comparação de esquemas sistemáticos e não sistemáticos para códigos polares com N=1024 e R=$\frac{1}{2}$. Retirado de [11].

d Decodificação

No contexto dos códigos polares, a tarefa do decodificador consiste em gerar uma estimativa $\hat{\boldsymbol{u}}_1^N$ de $\boldsymbol{u}_1^N$, dado o conhecimento de $\mathbb{A}$, $\boldsymbol{u}_{\boldsymbol{A}^c}$ e $\boldsymbol{y}_1^N$. Como o decodificador detém o conhecimento do conjunto congelado $\boldsymbol{u}_{\boldsymbol{A}^c}$, erros são evitados nesta parte. Logo, o real objetivo consiste em gerar a estimativa $\hat{\boldsymbol{u}}_{\boldsymbol{A}}$ de $\boldsymbol{u}_{\boldsymbol{A}}$. Para isso, Arikan [1] propôs um decodificador de cancelamento sucessivo (SC) capaz de atingir a capacidade dos canais B-DMCs sob condições de comprimento de código infinito. Entretanto, para códigos de comprimento finito, essa esquema de decodificação apresenta um baixo desempenho e alta latência, devido à sua natureza serial.

Tal e Vardy [2], por sua vez, propôs uma alternativa em lista do decodificador SC, o SCL, com o intuito de melhorar o desempenho para códigos de comprimento finito, se aproximando do limite de máxima verossimilhança (ML). Em seguida, códigos de redundância cíclicas (CRC) foram adicionados no decodificador SCL melhorando seu desempenho através do aumento da distância de Hamming. Sendo denominado como decodificador de cancelamento sucessivo em lista com CRC (CRC-SCL), esse algoritmo se provou como um poderoso esquema de decodificação, referência na literatura, apesar de sua alta complexidade e ainda natureza serial.

Como alternativa, Arikan [7] propôs novamente um algoritmo de decodificação iterativo com alto potencial de paralelismo baseado em propagação de crenças (BP) sob o diagrama de codificação polar. No entanto, infelizmente, sua performance ainda nao é equiparável com a decodificacao CRC-SCL.

A seguir, apresentaremos os quatros principais decodificadores presentes na literatura: Decodificador de Cancelamento Sucessivo (SC), Decodificador de Cancelamento Sucessivo em Lista (SCL), Decodificador de Cancelamento Sucessivo em Lista com CRC e o Decodificador BP.

1 Decodificação de Cancelamento Sucessivo (SC)

O decodificador SC estima os bits transmitidos $\boldsymbol{u}_1^N$ como $\hat{\boldsymbol{u}}_1^{\boldsymbol{N}}$ usando a palavra código recebida $\boldsymbol{y}_1^N \in Y$ nos canais B-DMC, $W_N : \mathcal{X}^N \to \mathcal{Y}^N$. As informações de verossimilhança do canal, LLRs, obtidas a partir do vetor $\boldsymbol{y}_1^{\boldsymbol{N}}$, são definidas por meio do log da razão das verossimilhanças.

$$LLR_{n+1,j} = \ln \frac{P_r(x_j = 0|y_j)}{P_r(x_j = 1|y_j)} = \frac{2y_j}{\sigma^2} \tag{16}$$

Enquanto computa os valores das LLRs intermediárias a partir dos valores das LLRs do canal, o decodificador executa a propagação através de uma decisão do tipo soft. Apenas após uma sequência de computações de LLRs, o decodificador SC estima $\hat{\boldsymbol{u}}_1^N$ baseado em decisões do tipo hard em uma ordem sucessiva de $\hat{\boldsymbol{u}}_1$ para $\hat{\boldsymbol{u}}_N$. Em outras palavras, $\hat{\boldsymbol{u}}_i$ é decidido de acordo com $\hat{\boldsymbol{u}}_j$ para $1 < j < i - 1$.

A descrição em alto nível da decodificação BP é ilustrada no Algoritmo 1. O algoritmo recebe a palavra recebida $\boldsymbol{y}_1^N$, o comprimento do código N, e o conjunto dos bits de informação $\mathbb{A}$ como entrada e determina a estimativa dos bits de informação $\hat{\boldsymbol{u}}_{\mathbb{A}}$ como vetor de saída. Existem N etapas de decisão no algoritmo. Se uma decisão hard pertence ao conjunto de bits congelados, $\mathbb{A}^c$, a decisão está predeterminada, uma vez que tanto o codificador quanto o decodificador detém o conhecimento deste bit. Caso contrário, o decodificador define sua decisão hard com base nas informações de decisão soft. Após que as N decisões hard são tomadas, a saída do decodificador constitui os bits de informação.

Algorithm 1 Decodificador de Cancelamento Sucessivo

Entrada: Vetor recebido, $\boldsymbol{y_1^N}$
Entrada: Comprimento de código, N
Entrada: Conjunto de bits de informação, $\mathbb{A}$
Entrada: Vetor de bits congelados, $\boldsymbol{u}_{\mathbb{A}^c}$
Saida: Bits de informação estimados, $\hat{\boldsymbol{u}}_{\mathbb{A}}$

begin
for i $\leftarrow$ 1:N **do**
  if i $\notin \mathbb{A}$ **then**
    $\hat{\boldsymbol{u}}_{\boldsymbol{i}} \leftarrow \boldsymbol{u}_{\boldsymbol{i}}$
  else
    if $\log \frac{W_N^{(i)}(y_1^N, u_1^{(i-1)}|\hat{u}_i=0)}{W_N^{(i)}(y_1^N, u_1^{(i-1)}|\hat{u}_i=1)} \geq 0$ **then**
      $\hat{\boldsymbol{u}}_{\boldsymbol{i}} \leftarrow 0$
    else
      $\hat{\boldsymbol{u}}_{\boldsymbol{i}} \leftarrow 1$
    end if
  end if
end for
Retorna $\hat{\boldsymbol{u}}_{\mathbb{A}}$

Três diferentes funções são definidas para ilustrar o comportamento do decodificador SC. Essas funções são chamadas f, g, e d. A função f é responsável pelo cálculo da operação de divisão de canal superior W^-, definido na seção 2, dado, na representação de razão de verossimilhança (L), por

$$f(L_a, L_b) = L_c \tag{17a}$$
$$= \frac{W(y_1^2|\hat{u}_c = 0)}{W(y_1^2|\hat{u}_c = 1)} \tag{17b}$$
$$= \frac{W(y_1|\hat{u}_a = 0)W(y_2|\hat{u}_b = 0) + W(y_1|\hat{u}_a = 1)W(y_2|\hat{u}_b = 1)}{W(y_1|\hat{u}_a = 0)W(y_2|\hat{u}_b = 1) + W(y_1|\hat{u}_a = 1)W(y_2|\hat{u}_b = 0)} \tag{17c}$$
$$= \frac{L_a L_b + 1}{L_a + L_b} \tag{17d}$$

pois, na equação 17c, o numerador e o denominador foram divididos por $W(y_1|\hat{u}_a = 1)W(y_2|\hat{u}_b = 1)$. Já na representação de razão do log da verossimilhança (LLR), a função f é dada por

$$f(\gamma_a, \gamma_b) = \gamma_c \tag{18a}$$
$$= 2\tanh^{-1}\left(\tanh\left(\frac{\gamma_a}{2}\right)\tanh\left(\frac{\gamma_b}{2}\right)\right) \tag{18b}$$
$$= \text{sign}(\gamma_a\gamma_b)\min(|\gamma_a|, |\gamma_b|) \tag{18c}$$

em que a aproximação de mínimo-soma foi definido para a decodificação BP dos códigos LDPC [19], usada na decodificação SC dos códigos polares pela primeira vez em [20].

Já a função g computa a operação de divisão de canal inferior W^+ no decodificador SC. Sendo a

decisão soft baseada na razão de verossimilhança, g é dado por

$$g(L_a, L_b) = \frac{W(y_1^2|\hat{u}_c = 0)}{W(y_1^2|\hat{u}_c = 1)} \tag{19a}$$

$$= \begin{cases} \frac{W(y_1|\hat{u}_a=0)W(y_2|\hat{u}_b=0)}{W(y_1|\hat{u}_a=1)W(y_2|\hat{u}_b=1)}, & \text{se } \hat{u} = 0 \\ \frac{W(y_1|\hat{u}_a=1)W(y_2|\hat{u}_b=0)}{W(y_1|\hat{u}_a=0)W(y_2|\hat{u}_b=1)}, & \text{se } \hat{u} = 1 \end{cases} \tag{19b}$$

$$= L_a^{(1-2\hat{u}_c)} L_b \tag{19c}$$

onde $\hat{u}_c$ representa a soma parcial dos bits decodificados anteriormente. Essa soma parcial corresponde a propagação dos bits decodificados da esquerda para direita no diagrama de decodificação.

A função g, em sua representação LLR, é dada por:

$$g(\gamma_a, \gamma_b, \hat{u}) = (-1)^{(\hat{u})}\gamma_a + \gamma_b \tag{20}$$

Por fim, a função d é uma função de decisão, que computa decisões hard a partir de decisões soft, definido por:

$$\hat{u}_i = \begin{cases} 0, & \text{se } \notin \mathbb{A} \\ 0, & \text{se } \in \mathbb{A} \text{ e } \frac{W(y, \hat{u}_1^{(i-1)}|\hat{u}_i=0)}{W(y, \hat{u}_1^{(i-1)}|\hat{u}_i=1)} \geq 1 \\ 1, & \text{caso contrário} \end{cases} \tag{21a}$$

A figura 9 ilustra graficamente como as razões de verossimilhança L são calculadas recursivamente nos nós de decisão, a partir do canal L, através das funções f e g.

A tabela abaixo lista as operações realizadas para decodificar cada bit $\hat{u}_i$, descrevendo o conjunto de nós ativados em cada estágio da figura 9. Esse padrão de ativação pode ser determinado pelo índice i, tratando-o como um número binário. O bit l desse número corresponde ao estágio l, e f (ou g) é usado se seu bit é 0 (ou 1). Note que todo nó do diagrama de decodificação deve ser ativado para decodificar uma palavra código recebida.

Estágio, i	0	1	2	3	4	5	6	7
l=2	f	f	f	f	g	g	g	g
l=1	f	f	g	g	f	f	g	g
l=0	f	g	f	g	f	g	f	g

Tabela 1: Operações realizadas por um decodificador SC com N=8

O processo de decodificação de cancelamento sucessivo (SC) para N=4 é mostrado na figura 10. Inicialmente, $c_1 = f(L_{y0,y2})$ e $c_1 = f(L_{y1,y3})$ correspondem a operações f nos nós de verificação no estágio $l = 1$. Com isso, $\hat{u}_0 = f(c_0, c_1)$ é estimado a partir da operação f no nó variável no estágio $l = 0$.

Em seguida, $\hat{u}_1$ pode ser estimado a partir de c_0, c_1 e $\hat{u}_0$. Como os dois nós de verificação já foram processo anteriormente, essa etapa contém apenas uma única operação g no nó variável no estágio $l = 0$, $\hat{u}_1 = g(c_0, c_1, \hat{u}_0)$.

Usando as relações $c_2 = g(L_{y0}, L_{y2}, \hat{u}_0 \oplus \hat{u}_1)$ e $c_3 = g(L_{y1}, L_{y3}, \hat{u}_0)$, a estimação de $\hat{u}_2$ envolve duas operações g nos nós de verificação no estágio $l = 1$ usando a informação de $\hat{u}_0$ e $\hat{u}_1$ e uma operação f no nó variável no estágio $l = 0$, $\hat{u}_2 = f(c_2, c_3)$.

Finalmente, u_3 é obtido pela operação g no nó variável no estágio $l = 0$. Como na segunda etapa, somente esse nó precisa ser processado, $\hat{u}_3 = g(c_2, c_3, \hat{u}_2)$.

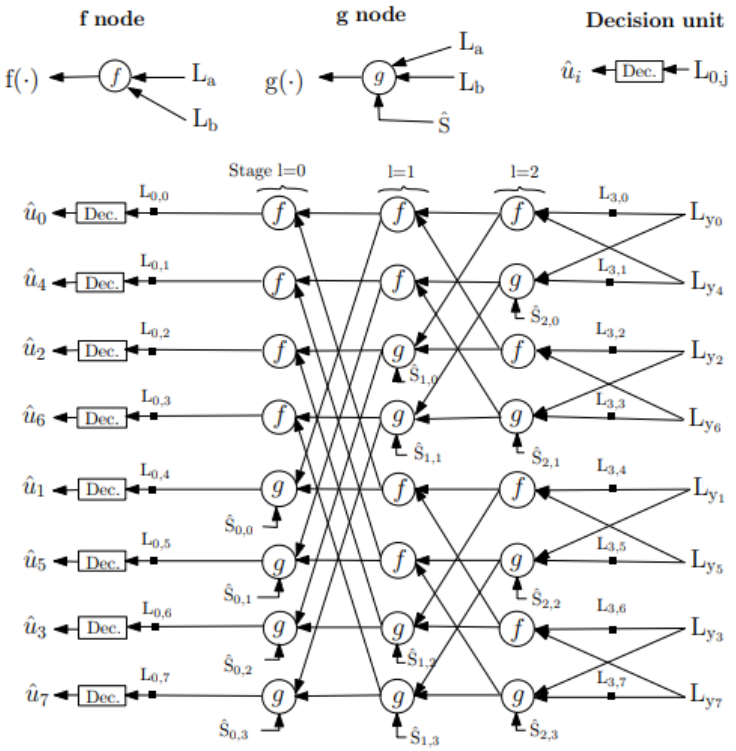


Figura 9: Estrutura do diagrama de decodificação polar para N=8. Retirado de [11]

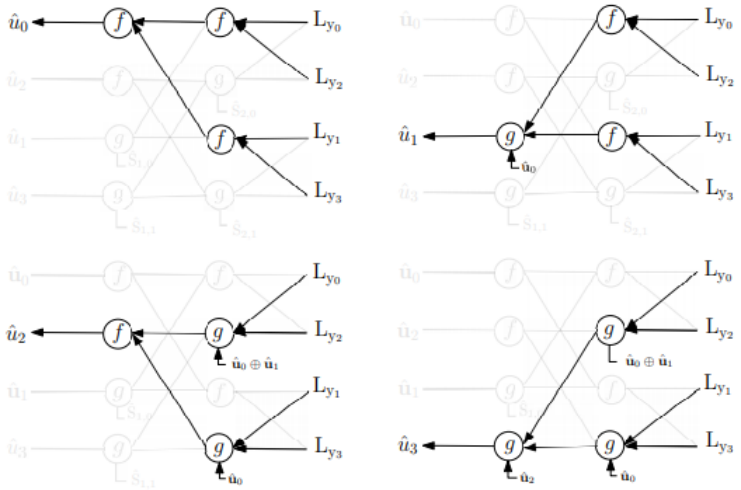


Figura 10: Decodificação de Cancelamento Sucessivo (SC) para código polar com N=4. Retirado de [11]

2 Decodificação de Cancelamento Sucessivo em Lista (SCL)

O algoritmo de decodificação em lista foi introduzido por Tal e Vardy em [21] e gera L candidatos de possíveis palavras códigos, aumentando a probabilidade de uma decodificação correta quando comparado com o SC. Com o uso do CRC, esse método tem se provado bem eficiente para códigos polares. Se L é grande o suficiente, a performance da decodificação ML é alcançada dado que um número suficiente de possíveis caminhos de decodificação são visitados [3].

Diferentemente da seleção de um particular caminho tomado em cada etapa de decisão na decodificação SC, a decodificação em lista (SCL) se expande em dois caminhos, $\hat{u}_i = 0$ e $\hat{u}_i = 1$ para cada etapa de decisão. Então, os L caminhos mais confiáveis são preservados em cada etapa de decisão.

Com isso, a complexidade da decodificação SCL é limitada pelo tamanho da lista, determinado por $O(Ln \log n)$.

Seja $\hat{u}_i[\tau]$ o bit $\hat{u}_i$ estimado no τ-ésimo caminho, onde $\tau \in 1, 2, ..., L$. A métrica de caminho (PM) do $\hat{u}_i[\tau]$, mostrada a seguir, é usada para medir a confiabilidade ou verossimilhança dos caminhos.

$$PM_\tau^{(i)} = PM_\tau^{(i-1)} + c_i[\tau] \cdot |L_n^{(i)}[\tau]| \tag{22}$$

onde i é o índice da etapa de decisão, PM_τ^0=0, $c_i[\tau] = 0$ quando i $\in A^c$ e $c_i[\tau] = 1$, caso contrário.

Como u_i possuem dois valores possíveis, os L caminhos com menores métricas são selecionados na etapa de decisão. Após o último bit decodificado, o caminho com menor métrica é considerado como resultado da decodificação.

Uma descrição em alto nível do algoritmo é mostrado no Algoritmo 2. O algoritmo SCL possui como entradas o vetor recebido $\mathbf{y_1^n}$, o tamanho do bloco N, o conjunto de informação $\mathbb{A}$, o vetor de bits congelados $\mathbf{u}_{\mathbb{A}^c}$ e o tamanho máixmo de lista L e calcula os bits de informação estimados $\hat{u}_\mathbb{A}$. A variável do tamanho da lista atual, cL, é definida como 1 na inicialização do algoritmo. Se a i-ésima decisão do tipo hard pertence ao conjunto congelado $\mathbb{A}^c$, a i-ésima decisão do tipo hard de todas as L listas são atualizadas com a decisão congelada, u_i. No caso de uma decisão livre, o decodificador verifica se o tamanho da lista atual é igual ao tamanho máximo da lista. Caso não forem iguais, o tamanho da lista dobra e o decodificador rastreia as verossimilhanças de ambas decisões. No caso em que todas as L listas estão ocupadas, o decodificador classifica a verossimilhança das 2L listas para continuar com os L melhores caminhos da decodificação. Ao final da última etapa de decisão, o decodificador emite os bits livres do melhor caminho na lista como $\hat{u}_\mathbb{A}$.

Algorithm 2 Decodificador de Cancelamento Sucessivo em Lista

Entrada: Vetor recebido, $\boldsymbol{y_1^N}$
Entrada: Comprimento de código, N
Entrada: Conjunto de bits de informação, $\mathbb{A}$
Entrada: Vetor de bits congelados, $\boldsymbol{u}_{\mathbb{A}^c}$
Entrada: Tamanho máximo da lista, L
Saida: Bits de informação estimados, $\hat{\boldsymbol{u}}_\mathbb{A}$

begin
cL ← 1, // Variável do tamanho da lista atual
for i ← 1:N **do**
  if i $\notin \mathbb{A}$ **then**
    for τ ← 1:cL **do**
      $\hat{u}_{\tau,i} \leftarrow u_i$
    end for
  else
    if cl $\neq$ L **then**
      for τ ← 1:cL **do**
        $\hat{u}_{\tau,i} \leftarrow 0$
        $\hat{u}_{\tau+cL,i} \leftarrow 1$
      end for
    else
      s ← sort($W_N^{(i)}(y_1^N, u_1^{(i-1)})$
      for τ ← 1:cL **do**
        $\hat{u}_{\tau,i} \leftarrow$ s
      end for
    end if
  end if
end for
Retorna $\hat{\boldsymbol{u}}_\mathbb{A}$

Um exemplo para L=4 é considerado na figura 11, onde N=4 e todos os bits estão livres. Em (a), o algoritmo se inicia e o primeiro bit pode ser 0 ou 1. Na segunda etapa, (b), o segundo bit assume 0 ou 1. Sendo assim, as palavras possíveis são 00,01,10,11. Em (c), é mostrado todos as opções possíveis para o primeiro, segundo e terceiro bit, mas note que há 8 caminhos possíveis e devemos manter apenas os L=4 caminhos mais prováveis. Em (d), é mostrado que foi mantido apenas as palavras 010,011,100,111. A decodificação com o algoritmo em lista continua para o quarto bit em (e). Entretanto, os caminhos possíveis são novamente iguais a 8 e são selecionados os L=4 melhores caminhos em (f). Finalmente, a decodificação termina e é obtido as palavras códigos {0100,0110,0111,1111}.

Infelizmente, a performance dos códigos polares é, em geral, pior do que códigos LDPC e Turbo de comprimento similar. Entretanto, estudos mostraram que a palavra código transmitida não é a palavra código mais provável, embora que esteja dentro da lista que o decodificador gera. Assim, a performance pode ser melhorada usando um CRC adicional, tornando o decodificador comparável aos códigos LDPC e Turbo.

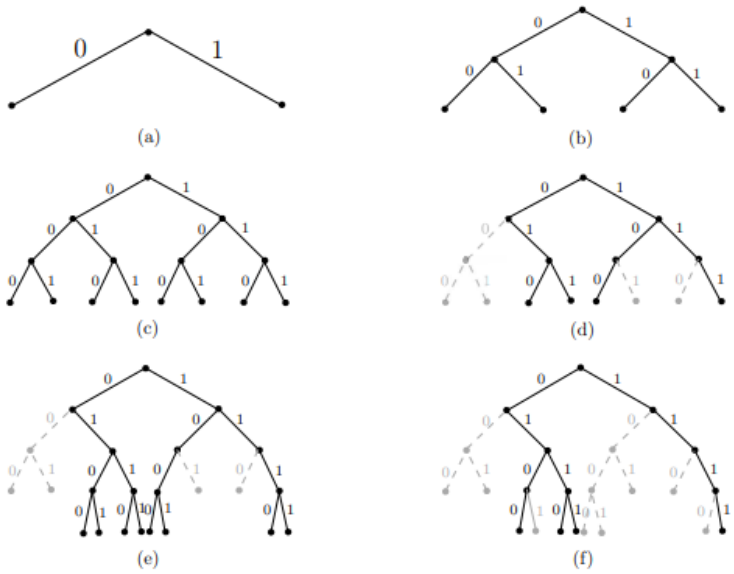


Figura 11: Evolução do caminho da decodificação para L=4. Retirado de [11]

A figura 12 mostra a performance dos decodificadores para diferentes tamanhos de listas considerando códigos polares sistemáticos em um cenário eMBB com R=$\frac{1}{2}$ e N=512. Os resultados mostram que a performance para o decodificador com L=1 é equivalente ao decodificador de cancelamento sucessivo e que há uma considerável melhora conforme o aumento do número de candidatos nas listas. Em particular, o ganho pelo uso de listas é bastante significativo quando comparado com o decodificador sucessivo (L=1), porém esse ganho perde notoriedade conforme o tamanho da lista aumenta.

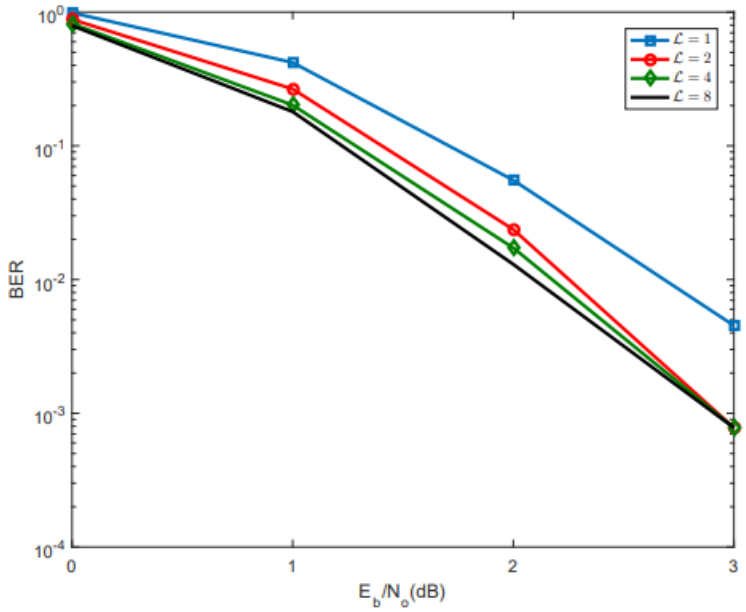


Figura 12: Comparação do decodificador de cancelamento sucessivo em lista para diferentes L. Retirado de [11]

3 Decodificação de Cancelamento Sucessivo em Lista com CRC (CRC-SCL)

O algoritmo de cancelamento sucessivo em lista com verificação de redundância cíclica (CRC-SCL) consiste na concatenação dos algoritmos de decodificação SC, SCL e CRC, cujo objetivo é aumentar a taxa de transferência do decodificador SCL [21], [22], [23] e [24].

Uma descrição em alto nível do algoritmo é mostrado no Algoritmo 3. As entradas desse decodificador são: o vetor recebido $\mathbf{y_1^n}$, o comprimento de bloco N, o conjunto de informação $\mathbb{A}$, o vetor de bits congelados $\mathbf{u_{A^c}}$ e o tamanho máximo da lista L. A saída do algoritmo é o vetor de bits livres $\hat{u}_{\mathbb{A}}$. Da mesma forma que o decodificador SCL definido anteriormente, esse algoritmo calcula os L vetores candidatos se baseando em decisões do tipo hard. Porém, se algum deles possuem um CRC válido, o algoritmo finaliza com essa saída. Caso contrário, o algoritmo termina com o vetor candidato mais provável [25].

O projeto polinomial CRC [26] depende do comprimento total máximo N do bloco. De acordo com [27], onde *d* é o grau do polinômio gerador, o comprimento total máximo do bloco é dado por 2^{d-1}. Assim, o CRC-8 deve ser usado para N $\geq$ 128 e CRC-16 é sugerido para N $\geq$ 32768.

A figura 13 mostra a performance dos decodificadores de lista para diferentes CRCs considerando códigos polares sistemáticos em um cenário eMBB com taxa R=$\frac{1}{4}$ e N=1920. Os resultados mostram que a performance do decodificador por lista melhora com o aumento do número de candidatos na lista L e o tamanho do CRC.

Algorithm 3 Decodificador de Cancelamento Sucessivo em Lista

Entrada: Vetor recebido, $\boldsymbol{y}_1^N$
Entrada: Comprimento de código, N
Entrada: Conjunto de bits de informação, $\mathbb{A}$
Entrada: Vetor de bits congelados, $\boldsymbol{u}_{\mathbb{A}^c}$
Entrada: Tamanho máximo da lista, L
Saida: Bits de informação estimados, $\hat{\boldsymbol{u}}_{\mathbb{A}}$

begin
  j $\leftarrow$ *Falso* // Vetor CRC válido da variável SC
  k $\leftarrow$ *Falso* // Vetor CRC válido da variável SCL
  $\hat{u}_A \leftarrow$ *Decodificação de Cancelamento Sucessivo*($\boldsymbol{y}_1^n$,N,A,u_{A^c})
  j $\leftarrow$ *Decodificação de Verificação de Redundância Cíclica*($\hat{u}_{\mathbb{A}}$)
  if j is *True* **then**
    Retorna $\hat{u}_A$
  else
    $\hat{u}_{l,A} \leftarrow$ *Decodificação de Cancelamento Sucessivo em Lista*($\boldsymbol{y}_1^n$,N,A,$u_{\mathbb{A}^c}$,L)
    for l $\leftarrow$ 1:L **do**
      k $\leftarrow$ *Decodificação de Verificação de Redundância Cíclica*($\hat{u}_{l,\mathbb{A}}$)
      if k is True **then**
        $\hat{u}_{\mathbb{A}} \leftarrow \hat{u}_{\tau,\mathbb{A}}$
        Retorna $\hat{u}_{\mathbb{A}}$
      end if
    end for
    $\hat{u}_{\mathbb{A}} \leftarrow \hat{u}_{\tau,\mathbb{A}}$
    Retorna $\hat{u}_{\mathbb{A}}$
  end if

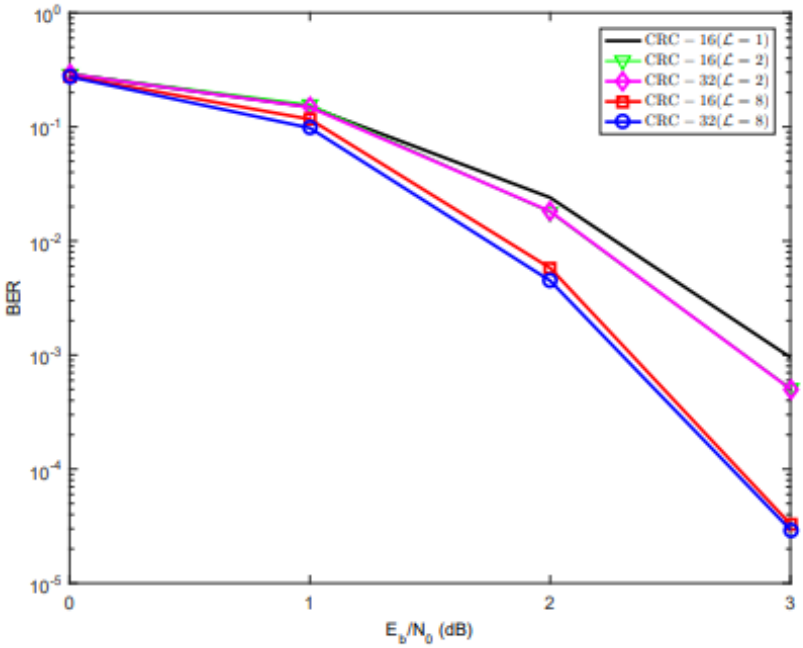


Figura 13: Comparação do decodificador de cancelamento sucessivo em lista para diferentes CRCs. Retirado de [11]

4 Decodificação BP

A decodificação por Belief Propagation (BP) foi projetado com intuito de lidar com a natureza sequencial dos decodificadores SC. Se trata de um algoritmo de troca de mensagens baseado no diagrama de codificação dos códigos polares mostrado na figura 14, sendo usado para estimar a palavra código $\mathbf{x}$ ou a mensagem $\hat{\mathbf{u}}$.

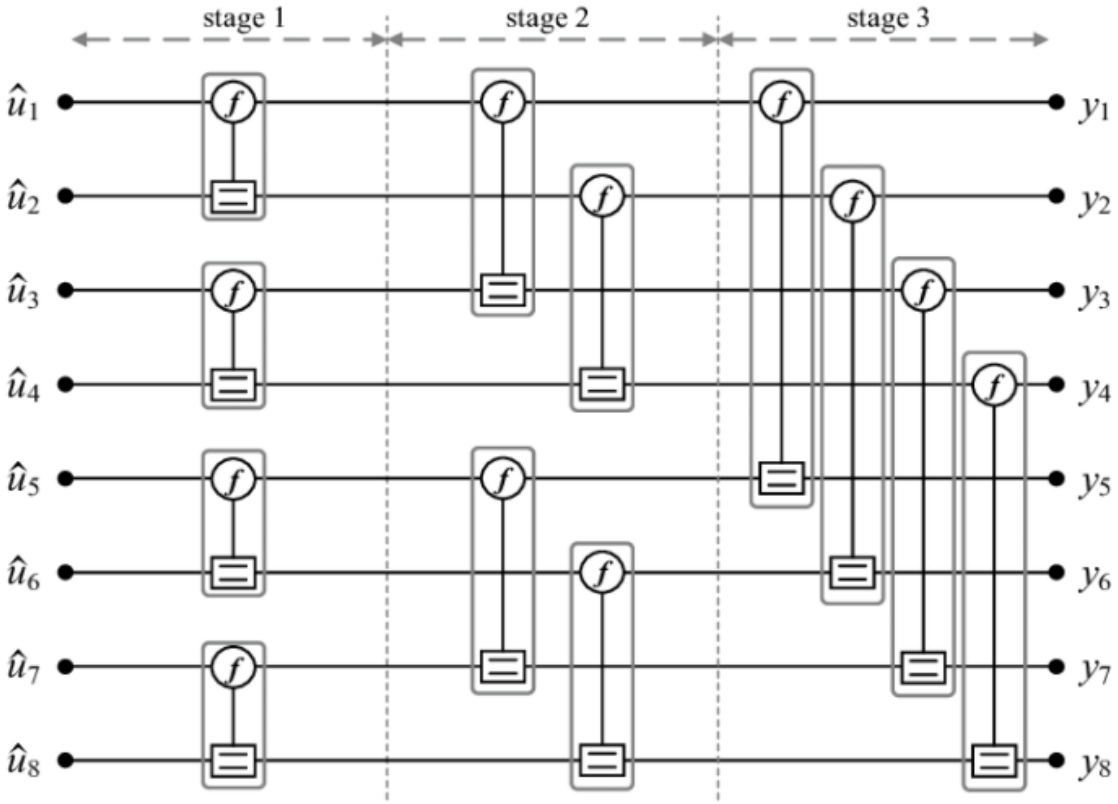


Figura 14: Diagrama de fatores para códigos polares.

Para um código polar de comprimento N, existem $\log N+1$ estágios e $N\cdot(\log N+1)$ nós. As mensagens são passadas iterativamente entre os nós adjacentes da esquerda para a direita e da direita para a esquerda até que um número máximo de iterações seja atingido. No fim, uma decisão do tipo hard é utilizada para estimar a palavra código e a mensagem. A unidade básica do diagrama de codificação é o elemento de processamento (PE), que conecta 4 nós em 2 estágios consecutivos, conforme mostrado na figura 15.

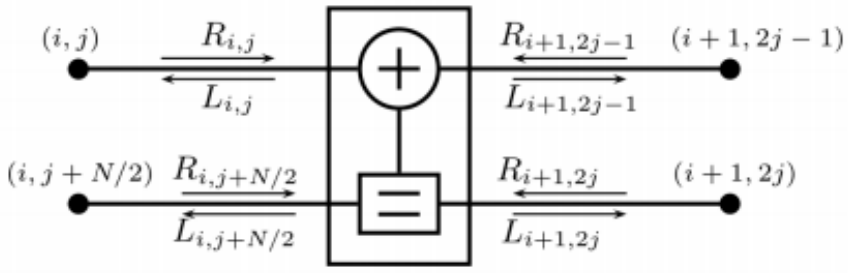


Figura 15: Elemento de processamento do decodificador BP.

onde o par (i,j), $1 \leq i \leq \log N$ e $1 \leq j \leq N/2$, indica o j-ésimo nó no i-ésimo estágio no qual possui dois tipos diferentes de mensages: L e R.

O decodificador BP propaga mensagens do tipo soft entre nós adjacentes com dois tipos de mensagens envolvidas: as mensagens da esquerda para direita, denominadas mensagens R, e as mensagens da direita para a esquerda, denominadas mensagens L. No domínio LLR (log da razão de verossimilhança), as LLRs de entrada para decodificação BP são inicializadas como:

$$L_{n+1,j}^{(0)} = \ln \frac{P_r(x_j = 0|y_j)}{P_r(x_j = 1|y_j)} = \frac{2y_j}{\sigma^2} \tag{23.1}$$

$$R_{1,j}^{(0)} = \begin{cases} 0 & \text{if j} \in \mathbb{A} \\ \infty & \text{if j} \in \overline{\mathbb{A}} \end{cases} \tag{23.2}$$

onde x_j e y_j denotam o j-ésimo bit da palavra código modulada e do vetor recebido, respectivamente.

A propagação das mensagens ao longo do diagrama de codificação é baseada na seguinte regra de atualização iterativa:

$$\begin{aligned} L_{i,j}^{(t)} &= g(L_{i+1,j}^{(t-1)}, L_{i+1,j+N/2^i}^{(t-1)} + R_{i,j+N/2^i}^{(t-1)}) \\ L_{i,j+N/2^i}^{(t)} &= g(L_{i+1,j}^{(t-1)}, R_{i,j}^{(t-1)}) + L_{i+1,j+N/2^i}^{(t-1)} \\ R_{i+1,j}^{(t)} &= g(R_{i,j}^{(t-1)}, L_{i+1,j+N/2^i}^{(t-1)} + R_{i,j+N/2^i}^{(t-1)}) \\ R_{i+1,j+N/2^i}^{(t)} &= g(L_{i+1,j}^{(t-1)} + R_{i,j}^{(t-1)}) + R_{i,j+N/2^i}^{(t-1)} \end{aligned} \tag{24}$$

onde $g(x, y)$ se refere ao seguinte operador:

$$g(x, y) = \ln \frac{1 + e^{x+y}}{e^x + e^y} \tag{25}$$

Porém, como a equação (25) é muito complexa para ser integrada em circuitos, é utilizado a seguinte aproximação para implementações em hardware:

$$g(x, y) \approx sign(x) \cdot sign(y) \cdot \min(|x|, |y|) \tag{26}$$

Com a introdução desta aproximação, é necessário levar em conta as perdas de performance. Com intuito de minimizá-las, um fator de escala, s, é, portanto, introduzido a fim de mitigar erros. Aplicando s, o conjunto de equações definido por (24) se torna:

$$\begin{aligned} L_{i,j}^{(t)} &= s \cdot g(L_{i+1,j}^{(t-1)}, L_{i+1,j+N/2^i}^{(t-1)} + R_{i,j+N/2^i}^{(t-1)}) \\ L_{i,j+N/2^i}^{(t)} &= s \cdot g(L_{i+1,j}^{(t-1)}, R_{i,j}^{(t-1)}) + L_{i+1,j+N/2^i}^{(t-1)} \\ R_{i+1,j}^{(t)} &= s \cdot g(R_{i,j}^{(t-1)}, L_{i+1,j+N/2^i}^{(t-1)} + R_{i,j+N/2^i}^{(t-1)}) \\ R_{i+1,j+N/2^i}^{(t)} &= s \cdot g(L_{i+1,j}^{(t-1)} + R_{i,j}^{(t-1)}) + R_{i,j+N/2^i}^{(t-1)} \end{aligned} \tag{27}$$

Essa correção introduz melhorias significativas na performance do decodificador, pois essa aproximação é mais fiel a função $g(x, y)$ original sem aproximação. Simulações revelaram que o melhor valor para s é 0.9375 = 1 - 2^{-4} com uma melhoria de 0.5 dB.

Por fim, quando um número máximo de iterações, T_{max}, é atingido, o bit de informação $\hat{\mathbf{u}}_j$ e a palavra código transmitida $\hat{\mathbf{x}}_j$ são estimados baseados nas LLRs a partir do seguinte critério de decisão do tipo hard:

$$\begin{aligned} \hat{u}_j &= \begin{cases} 0, & \text{if } L_{1,j}^{T_{max}} + R_{1,j}^{T_{max}} > 0 \\ 1, & \text{caso contrário} \end{cases} \\ \hat{x}_j &= \begin{cases} 0, & \text{if } L_{n+1,j}^{T_{max}} + R_{n+1,j}^{T_{max}} > 0 \\ 1, & \text{caso contrário} \end{cases} \end{aligned} \tag{28}$$

A fim de reduzir a complexidade computacional dada pela natureza iterativa do decodificador, é fundamental buscarmos critérios de parada antecipada cujo principal objetivo consiste em encontrar uma solução correta antes de atingir o número máximo de iterações. Os dois primeiros métodos que surgiram foram o G-Matrix [28] e o minLLR [28] que, embora eficientes, possuem um elevado custo computacional. O terceiro, o critério H-Matrix [29], se trata de uma melhoria do G-Matrix e possui uma performance superior usando menos recursos adicionais. Além disso, o Worst of Information Bits (WIB) [29] e o Frozen Bit Error Rate (FBER) [30] são apresentados como critério de parada antecipada de baixa complexidade. Não obstante, como apenas o método G-Matrix será utilizado como critério de parada do nosso decodificador BP proposto, apenas o mesmo será descrito em detalhes. Ao leitor interessado, referências foram apresentadas para os demais critérios.

O critério G-Matrix tem como proposta reduzir a complexidade computacional explorando a matriz geradora G, definida por:

$$G_N = F^{\oplus n} \tag{29}$$

Assim, sabendo que a palavra código $\mathbf{x}$, transmitida pelo canal, é construída pela multiplicação da mensagem $\mathbf{u}$ e a matriz geradora G, se $\hat{\mathbf{x}} = \hat{\mathbf{u}}\mathbf{G_N}$, então têm-se uma decodificação bem sucedida e o processo iterativo pode ser paralizado. Caso contrário, se essa condição não for atingida e o número de iterações máximo for atingido, a decodificação estará incorreta. Esse processo é repetido para toda iteração. A partir desse método, o número de iterações pode ser reduzido, sem perda de desempenho do FER, de 23% a 2,5 dB e 42% a 3,5 dB normalizado a 40 iterações.

3 Decodificação Orientada a Q-Learning

A fim de melhorarmos o desempenho do algoritmo de decodificação BP frente aos decodificadores CRC-SCL, propomos uma técnica de ponderação das LLRs em cada elemento de processamento, PE, baseado em seus sinais e nas suas distâncias euclidianas a partir de uma métrica heurística. Esse decodificador foi denominado BP melhorado e exige a estimativa de um fator de correção único para todos os PE de forma a maximizar a probabilidade de uma decodificação correta. Com isso, usando Q-Learning, aprimoraremos o cálculo deste fator de correção possibilitando uma escolha exclusiva por estado, a ser definido posteriormente, e por PE. Dessa forma, o algoritmo de decodificação orientado a Q-Learning proposto consiste em duas etapas: inicialmente, as mensagens são ponderadas em cada PE usando uma métrica específica e, em seguida, o algoritmo é instigado a descobrir, ao longo do aprendizado, os melhores fatores de correção em prol de uma decodificação bem sucedida.

a Decodicação BP Melhorada

Nesta seção, propomos uma técnica de ponderação para decodificadores BP. Essa técnica servirá de base para o nosso decodificador orientado a Q-Learning, discutido na seção seguinte. Conforme pode ser visto na equação (24), a propagação das mensagens exigem quatros parâmetros para cada direção, a saber $L_{i+1,j}^{(t)}$ $L_{i+1,j+N/2^i}^{(t)}$, $R_{i,j}^{(t)}$ e $R_{i,j+N/2^i}^{(t)}$, para mensagens L e $R_{i,j}^{(t)}$, $R_{i,j+N/2^i}^{(t)}$, $L_{i+1,j}^{(t)}$ e $L_{i+1,j+N/2^i}^{(t)}$ para mensagens R. Além disso, como as atualizações dos PEs podem ser sintetizadas em sucessivas operações de módulo e sinal definida pela equação (26), nossa técnica de ponderação é construída de forma a considerar como essas 4 LLRs evoluem em termos de sinal e módulo ao longo do tempo.

Assim, introduzindo quatro fatores de correção, ρ_1, ρ_2, ρ_3, ρ_4, modificaremos como as LLRs serão atualizadas, conforme pode ser visto abaixo.

$$\begin{aligned} L_{i,j}^{(t)} &= g\left(\rho_1 \cdot L_{i+1,j}^{(t-1)}, \rho_1\left(L_{i+1,j+N/2^i}^{(t-1)} + R_{i,j+N/2^i}^{(t)}\right)\right) \\ L_{i,j+N/2^i}^{(t)} &= g\left(\rho_2 \cdot L_{i+1,j}^{(t-1)}, \rho_2 \cdot R_{i,j}^{(t)}\right) + \rho_2 \cdot L_{i+1,j+N/2^i}^{(t-1)} \end{aligned} \tag{30}$$

onde

$$\begin{aligned} \rho_1 &= 1 + \beta \cdot \left[\frac{||L_{i,j}^{(t)}| - |L_{i,j}^{(t-1)}||}{(|L_{i,j}^{(t)}| + |L_{i,j}^{(t-1)}|)}\right] \cdot \Delta_1 \\ \rho_2 &= 1 + \beta \cdot \left[\frac{||L_{i,j+N/2^i}^{(t)}| - |L_{i,j+N/2^i}^{(t-1)}||}{(|L_{i,j+N/2^i}^{(t)}| + |L_{i,j+N/2^i}^{(t-1)}|)}\right] \cdot \Delta_2 \\ \Delta_1 &= sign(L_{i,j}^{(t)} + L_{i,j}^{(t-1)}) \\ \Delta_2 &= sign(L_{i,j+N/2^i}^{(t)} + L_{i,j+N/2^i}^{(t-1)}) \end{aligned} \tag{31}$$

O método de ponderação é baseado na distância entre as LLRs no tempo *t* e *t-1* e se os sinais alteraram ao longo das iterações. Note que quando $|L_{i,j}^{(t)}|$ e $|L_{i,j}^{(t-1)}|$ estão próximos um do outro, ρ_1 é aproximadamente igual a 1. Consequentemente, a atualização do nó é similar ao da equação (24). Por outro lado, conforme esses valores desviam, maior será a ponderação. Além disso, os desvios em termos de sinal são considerados por Δ_1. Por fim, observe que essa mesma ideia é aplicada aos fatores de ponderação subsequentes.

$$\begin{aligned} R_{i+1,j}^{(t)} &= g\left(\rho_3 \cdot R_{i,j}^{(t)}, \rho_3\left(L_{i+1,j+N/2^i}^{(t-1)} + R_{i,j+N/2^i}^{(t)}\right)\right) \\ R_{i+1,j+N/2^i}^{(t)} &= g\left(\rho_4\left(L_{i+1,j}^{(t-1)} + R_{i,j}^{(t)}\right)\right) + \rho_4 \cdot R_{i,j+N/2^i}^{(t)} \end{aligned} \tag{32}$$

onde

$$\rho_3 = 1 + \beta \cdot \left[\frac{||R^{(t)}_{i+1,j}| - |R^{(t-1)}_{i+1,j}||}{(|R^{(t)}_{i+1,j}| + |R^{(t-1)}_{i+1,j}|)} \right] \cdot \Delta_3$$
$$\rho_4 = 1 + \beta \cdot \left[\frac{||R^{(t)}_{i+1,j+N/2^i}| - |R^{(t-1)}_{i+1,j+N/2^i}||}{(|R^{(t)}_{i+1,j+N/2^i}| + |R^{(t-1)}_{i+1,j+N/2^i}|)} \right] \cdot \Delta_4 \quad (33)$$
$$\Delta_3 = sign(R^{(t)}_{i+1,j} + R^{(t-1)}_{i+1,j})$$
$$\Delta_4 = sign(R^{(t)}_{i+1,j+N/2^i} + R^{(t-1)}_{i+1,j+N/2^i})$$

É importante salientar ainda que, no decodificador BP melhorado, β representa um fator de correção geral para todos os elementos de ponderação, cuja simulações indicam que o mesmo deve estar no intervalo entre [-0.50, 0.50]. Assim, um problema em aberto consiste em definir o melhor β para uma entrada específica, tarefa na qual o algoritmo Q-Learning, proposto na próxima seção, tenta resolver.

Uma descrição em alto nível do decodificador BP melhorado é ilustrado no Algoritmo 4. O Algoritmo toma o vetor recebido $\mathbf{y}_1^n$, o comprimento de bloco N, o número máximo de iterações T_{max}, e o conjunto de informação $\mathbb{A}$ e estima os bits livres $\hat{\mathbf{u}}_{\mathbb{A}}$. Primeiramente, as mensagens são inicializadas de acordo com a equação (23). Na primeira iteração, as mensagens são propagadas baseado na regra de atualização do BP convencional definido pela equação (24). Por outro lado, para as iterações subsequentes, as mensagens L são atualizadas de acordo com (30) e as mensagens R pela equação (32). Além disso, no fim de cada iteração, todas as mensagens são armazenadas para tornar o processo de decodificação possível. Após T_{max} iterações, $\hat{\mathbf{u}}$ é calculado pela equação (28) e $\hat{\mathbf{u}}_{\mathbb{A}}$ corresponde as $\mathbb{A}$ posições de $\hat{\mathbf{u}}$.

Algorithm 4 Decodificação BP Melhorada

Entrada: $\mathbf{y}_1^n$, N, T_{max}, $\mathbb{A}$
Saida: $\hat{\mathbf{u}}_{\mathbb{A}}$

for cada nó (i,j) **do**
  if (i==1) & (j $\notin \mathbb{A}$) **then**
    $R^{(0)}_{1,j} \leftarrow \infty$
  else if (i==n+1) **then**
    $L^{(0)}_{n+1,j} \leftarrow \frac{2y_j}{\sigma^2}$
  else
    $L^{(0)}_{i,j} \leftarrow 0$, $R^{(0)}_{i,j} \leftarrow 0$
  end if
end for
for (1<t< T_{max}) & cada nó (i,j) **do**
  if (t==1) **then**
    Atualize $L^{(1)}_{i,j}$ e $R^{(1)}_{i,j}$ de acordo com a equação (27)
    Armazene $L^{(1)}_{i,j}$ e $R^{(1)}_{i,j}$
  else
    Atualize $L^{(t)}_{i,j}$ e $R^{(t)}_{i,j}$ de acordo com as equações (30) e (32), respectivamente
    Armazene $L^{(t)}_{i,j}$ e $R^{(t)}_{i,j}$
  end if
end for
Compute $\hat{\mathbf{u}}$ de acordo com a equação (28)
Selecione as $\mathbb{A}$ posições de $\hat{\mathbf{u}}$ para compor $\hat{\mathbf{u}}_{\mathbb{A}}$
Retorna $\hat{\mathbf{u}}_{\mathbb{A}}$

b Reinforcement Learning e Q-Learning

Um processo de decisão Markoviano [31], MDP, é um framework matemático para problemas de tomadas de decisões sequenciais totalmente observáveis em ambientes estocásticos, sendo definido como uma tupla <S, A, T, R, γ>, onde S é o conjunto de estados, A é um conjunto finito de ações, $T : SxAxS \rightarrow [0, \infty)$ é a função de transição para um estado do sistema, $R : SxAxS \rightarrow \mathbb{R}$ é a função de recompensa limitada, e $\gamma \in (0, 1)$ é um fator de desconto.

Políticas de exploração são mapeamentos entre estados e distribuições de probabilidade baseado em ações. Dado um MPD <S, A, T, R, γ>, para cada par estado-ação (s, a) e política π, definimos o retorno esperado como

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi,T}\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_0 = s, a_0 = a\right] \tag{34}$$

onde s_t é uma amostra de T, a_t é uma amostra de $\pi(\cdot, s_t)$ e $r_{t+1} = R(s_t, a_t, s_{t+1})$. A função de valor associada é $V^{\pi}(s) = \max_{a \in A} Q^{\pi}(s, a)$, e objetivo do framework consiste em aprender a política ótima π^* que maximiza $V^{\pi}(s)$ [32].

Em ambientes de aprendizado por reforço, RL, T e R não são conhecidos, então, políticas ótimas são aprendidas a partir de experiências, definidas como as sequências de transição $(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$, t=0,1,..., divididos em episódios. Em outras palavras, a cada etapa, o agente, submetido ao processo de aprendizagem, está em um estado s_t, seleciona uma ação a_t e move para um próximo estado s_{t+1}, obtendo uma recompensa r_t, conforme pode ser visto na Fig. 16. Dado uma estimativa $\widetilde{G}$ do retorno esperado no tempo t a partir do estado $s = s_t$ tomando a ação $a = a_t$, a diferença temporal, TD, atualiza Q-values da seguinte maneira

$$Q(s_{t+1}, a_{t+1}) = Q(s_t, a_t) + \alpha\left(\widetilde{G}_t - Q(s_t, a_t)\right) \tag{35}$$

onde $\alpha \in [0, 1]$ é a taxa de aprendizado.

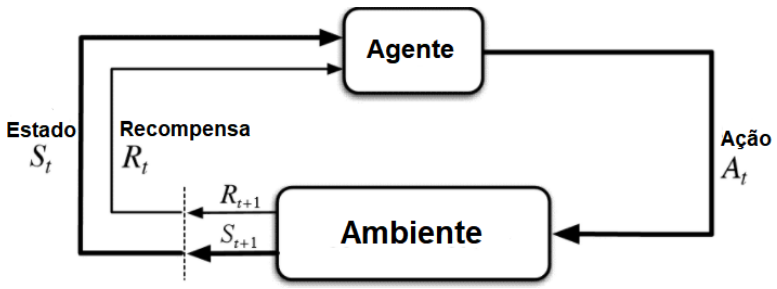


Figura 16: Interação agente-ambiente em aprendizado por reforço

Tipicamente, $\widetilde{G}$ é computado via *bootstrapping* a partir dos Q-values atuais a fim de reduzir a variância. Com isso, utilizaremos Q-Learning [33], um dos algoritmos *model-free* de aprendizado por reforço, para modelar a forma como o agente aprenderá a política ótima. Para tal,

$$\widetilde{G}_t = r_t + \gamma \max_{a' \in A} Q_t(s_{t+1}, a') \tag{36}$$

Com isso, a partir das equações (35) e (36), apresentamos a regra de atualização geral do Q-Learning

$$Q(s_{t+1}, a_{t+1}) = Q(s_t, a_t) + \alpha \left[r_t + \gamma \cdot \max_{a' \in A} Q(s_{t+1}, a') - Q(s_t, a_t) \right] \tag{37}$$

onde, novamente, α é a taxa de aprendizado, e afeta como os Q-values são alterados após a tomada de uma ação e a constante γ é o fator de desconto e determina quanta influência as recompensas futuras têm na atualização dos Q-values.

Portanto, para todo par estado-ação, um Q-value, $Q(s_t, a_t)$, mede a quantidade total de recompensas descontadas esperadas no futuro quando o agente move do estado s_t para o estado s_{t+1} tomando a ação a_t, e seguindo sua política π posteriormente.

c Decodificação Belief Propagation Orientada a Q-Learning

Conforme mencionado na seção 3.a, um problema em aberto consiste em computar o melhor valor de β. Entretanto, ao invés de ter um valor geral de β para todos os PEs no diagrama de codificação para códigos polares, avaliamos diferentes valores de β para cada PE a fim de evitar um subespaço enorme de estados. A partir dessa abordagem, nosso ambiente é composto por cada $\text{PE}_{l,m}$, l, m denota o m-ésimo PE no estágio l, onde m pertence ao intervalo $[1, \frac{N}{2}]$ e l pertence ao intervalo $[1, n]$. Assim, o agente, em dado estado, $s_{t_{l,m}}$, seleciona uma ação $a_{t_{l,m}}$ e aguarda o processo de decodificação retornar para $\text{PE}_{l,m}$ para ser finalmente recompensado.

- **Recompensa**:

  A recompensa quantifica o incentivo de escolher uma ação ao transicionar para algum estado. Pode ser positivo ou negativo, sendo este interpretado como uma penalidade para uma ação indesejável. A recompensa total com o fator de desconto que o agente atingirá de t até o final da tarefa pode ser definido por:

  $$R_t = r_t + \gamma r_{t+1} + ... + \gamma^{n-t} r_n = r_t + \gamma R_{t+1} \tag{38}$$

  Implementamos uma alta recompensa positiva para um processo de decodificação bem sucedido a fim de incentivar o agente a atingir esse objetivo. Além disso, o agente deve obter uma recompensa ligeiramente positiva se a LLR no tempo t tiver o mesmo sinal que a LLR no tempo $t-1$ em um dado PE. Por outro lado, se as LLRs nos tempos t e $t-1$ não tiverem o mesmo sinal, o agente deverá obter uma pequena penalidade. Ao fazer isso, evitamos ações indesejáveis, que nos afastam de uma decodificação bem sucedida. O fator de desconto e as recompensas utilizadas estão mostradas na tabela I.

  Tabela 2

Evento	Recompensa
Decodificação de sucesso	20/10/0
LLRs têm o mesmo sinal	1
LLRs não têm o mesmo sinal	-1
Fator de desconto	0.6

  Note que quando temos uma decodificação bem sucedida, a recompensa pode ser 20, 10 ou 0. Conforme discutido na seção 2.4 , a propagação das LLRs nos PEs envolve 4 números e geram duas LLRs de saída para cada direção de propagação. Portanto, se ambas LLRs de saída não mudaram de sinal ao longo do tempo, recompensamos o PE associado com 20. Caso contrário, se uma delas alteraram de sinal, recompensamos o PE associado com 10. Por fim, se ambas LLRs mudaram de sinal, recompensamos com 0.

- **Espaço de Estados**:

  O espaço de estados é o conjunto de todas as possíveis situações que um elemento de processamento pode ter. Para cada propagação, é necessário 4 números para computar as saídas. Assim, existem 2^4 variações de sinais e $4!$ variações de módulos, o que implica em um espaço de estados com $4! \cdot 2^4$= 384 estados possíveis. Dessa forma, um classificador de estados se faz necessário.

- **Ações**:

  No algoritmo QLBP proposto, durante o processo de decodificação, para cada PE no diagrama de codificação, o agente encontra um dos 384 estados e toma uma ação. Se o sinal da LLR no tempo t difere do sinal da LLR no tempo $t-1$, a escolha da ação mudará a forma com que a LLR é ponderada de acordo com (30) e (32). Caso contrário, é assumido $\rho_{1,2,3,4}$ = 1.

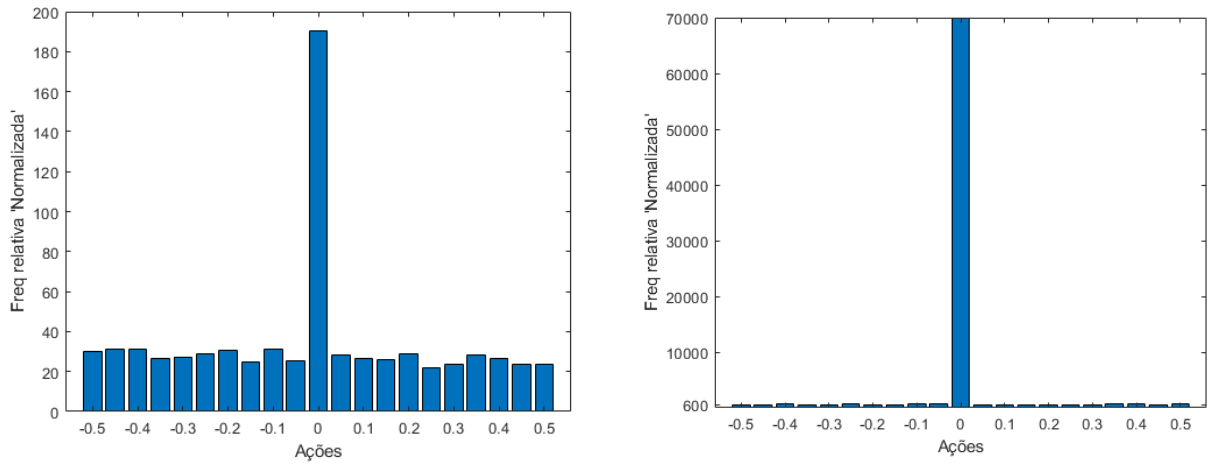
  A ação, no nosso caso, é um fator de desconto, β, e estudos de simulação mostraram que devemos atribuir um valor entre -0.5 e 0.5 de forma não uniforme, pois, a medida que a SNR aumenta, o algoritmo tende a escolher ações mais próximas de zero, ou seja, que realizam apenas um ajuste fino nas LLRs. O histograma da Fig. 17 resume a frequência relativa das ações em baixa e alta SNR, respetivamente. Note ainda que conforme a SNR aumenta, mais o algoritmo é acionado para definir uma ação ótima. Dessa forma, as ações consideradas neste projeto são valores discretos positivos e negativos cujos módulos são [0, 0.025, 0.05, 0.075, 0.1, 0.125, 0.15, 0.2, 0.3, 0.4, 0.5].

  Um agente pode interagir com o ambiente de duas formas diferentes. A primeira se baseia na interação com a matriz de pares estado-ação para armazenar e obter informações. Cada par estado-ação é associado com um Q-value, que indica a qualidade desta decisão. Assim, em um dado estado s^*, o agente seleciona a ação baseado no valor máximo de Q(s^*, a) em busca de uma maior recompensa. Este procedimento é conhecido como *exploiting*, pois utilizamos a informação que temos disponível para a tomada de decisão.

  A segunda forma de tomar uma ação é agir aleatoriamente, cujo procedimento é denominado *exploring*. Ao invés de selecionar ações baseado nas recompensas futuras máximas, selecionamos uma ação ao acaso. Agir aleatoriamente é importante, pois permite que o agente explore e descubra novos pares estados-ações que outrora não foram selecionados no processo de *exploitation*. Sendo assim, o agente melhora seu conhecimento atual acerca de cada ação, acarretando em um benefício a longo prazo, pois permite que o agente tome decisões com maior precisão.

  Portanto, quando o agente realiza *exploring*, estimativas mais precisas serão obtidas no futuro acerca dos valores dos pares estados-ações. Por outro lado, quando ele realiza *exploiting*, ele é mais recompensado. Assim, se trata de um *trade-off* cujo o equilíbrio entre *exploration* e *exploitation* é determinado por ϵ, através do método de seleção denominado ϵ-greedy [34], medida que define o quão frequente você quer realizar exploring, ao invés de exploiting. $\epsilon = 0.5$ foi utilizado em nossas simulações.

Uma descrição em alto nível do decodificador BP orientado a Q-Learning é mostrado no Algoritmo 5. O algoritmo recebe os parâmetros do BP convencional, tais como o vetor recebido $\mathbf{y}_1^n$, o comprimento do bloco N, o número máximo de iterações T_{max}, e o conjunto de informação $\mathbb{A}$, a Q-table e os parâmetros do Q-Learning, como a taxa de aprendizado α, o fator de desconto γ, e o parâmetro greedy ϵ, e tem como saída os bits livres estimados $\hat{\mathbf{u}}_{\mathbb{A}}$ e a Q-table atualizada. Inicialmente, as mensagens L e R são inicializadas e atualizadas de acordo com a decodificação BP melhorada descrita na seção 3.a. Depois, usando a abordagem ϵ-greedy, é escolhido o processo de exploitation ou exploration. Se o sinal da LLR em um nó no tempo t difere do sinal da LLR no tempo $t-1$, uma penalidade é aplicada. Caso contrário, aplicamos uma recompensa. Em seguida, o Q-value é atualizado de acordo com a equação (37). Além disso, se x é igual a $u \cdot G$, uma recompensa maior é considerada, o Q-value é computado novamente e o loop-for é rompido. Caso contrário, o loop-for dura por T_{max} iterações. Ao final, $\hat{\mathbf{u}}$ é calculado de acordo com a equação (28) e $\hat{\mathbf{u}}_{\mathbb{A}}$ corresponde a $\mathbb{A}$ posições de $\hat{\mathbf{u}}$.



(a) Frequência relativa das ações em SNR 0

(b) Frequência relativa das ações em SNR 4

Figura 17: Histograma das Frequências Relativas

Algorithm 5 Decodificador BP orientado a Q-Learning

Entrada: Parâmetros do BP Convencional, Q-Table, Parâmetros do Q-Learning
Saida: $\hat{\mathbf{u}}_{\mathbb{A}}$, Q-Table

Inicialização do BP Convencional
for (1<t< T_{max}) **do**
  for cada nó (i, j) **do**
    Atualize $L_{i,j}^{(t)}$ e $R_{i,j}^{(t)}$ de acordo com as equações (30) e (32), respectivamente
    Armazene $L_{i,j}^{(t)}$ e $R_{i,j}^{(t)}$
    if rand()< ϵ **then**
      Escolha uma ação aleatoriamente
    else
      Escolha a ação que maximiza o Q-value para o estado atual
    end if
    if $sign(LLR_{i,j}^{(t)}) \neq sign(LLR_{i,j}^{(t-1)})$ **then**
      Aplique uma penalidade
    else
      Aplique uma recompensa
    end if
    $Q(s_{t+1}, a_{t+1}) = Q(s_t, a_t) + \alpha \left[r_t + \gamma \cdot \max_{a' \in A} Q(s_{t+1}, a') - Q(s_t, a_t) \right]$
  end for
  if $(x = u \cdot G)$ **then**
    Aplique uma recompensa maior
    Compute $Q^n(s_t, a_t)$ novamente
    Pause o for-loop
  end if
end for
Compute $\hat{\mathbf{u}}$ de acordo com a equação (28)
Selecione as $\mathbb{A}$ posições de $\hat{\mathbf{u}}$ para compor $\hat{\mathbf{u}}_{\mathbb{A}}$
Retorna [$\hat{\mathbf{u}}_{\mathbb{A}}$, Q-Table atualizada]

4 Simulações

Nesta seção, os resultados das simulações são apresentados para demonstrar o desempenho dos algoritmos BP melhorado e BP orientado a Q-Learning propostos frente aos principais decodificadores presentes na literatura, como o SC original do Arikan, o SCL com Lista 4 e Lista 8 e o BP original do Arikan para N=256 e N=512 e R=$\frac{1}{2}$.

Conforme pode ser observado nas Figs, 18 e 19, para N=256, R=$\frac{1}{2}$, ambos resultados nos revelam que o decodificador BP orientado a Q-Learning supera os algoritmos SC, BP e BP Melhorado, em aproximadamente 0.5 dB e 0.4 dB em termos de BER e FER, respetivamente, em 2 dB, e se aproximando do desempenho dos decodificadores SCL. Entretanto, note que o ganho de desempenho decresce a medida que a razão sinal-ruído aumenta. Assim, o algoritmo BP com Q-Learning é mais adequado para implementações que operam em baixas SNRs neste comprimento de código.

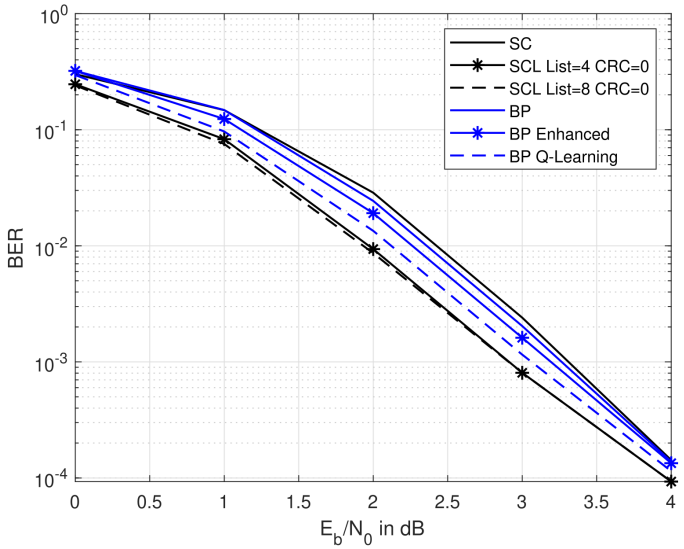


Figura 18: Comparação de BER entre o decodificador SC original de Arikan, decodificador de listas SCL com List-4 e List-8, decodificador BP de Arikan, os decodificadores propostos BP Enhanced e BP Q-learning para N=256 e K=128; sem uso de CRC.

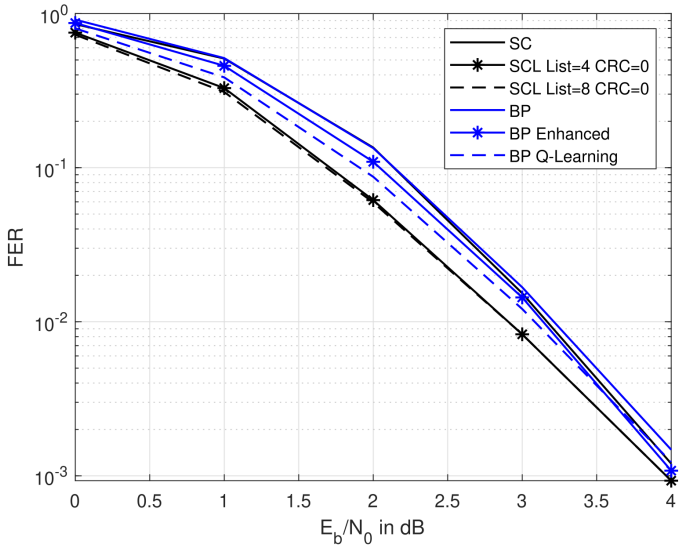


Figura 19: Comparação de FER entre o decodificador SC original de Arikan, decodificador de listas SCL com List-4 e List-8, decodificador BP de Arikan, os decodificadores propostos BP Enhanced e BP Q-learning para N=256 e K=128; sem uso de CRC.

No segundo exemplo, avaliamos os decodificadores para N=512 e K = 256 nas Figs. 20 e 21. Conforme mencionado anteriormente, o decodificador BP orientado a Q-Learning também supera os algoritmos SC, BP e BP melhorado em aproximadamente 0.4 dB e 0.25 dB em termos de BER e FER, respectivamente, em 2 dB. Além disso, mesmo que o desempenho dos decodificadores SCL não tenha sido atingido pelo decodificador proposto com Q-Learning, as técnicas propostas tem o desempenho ainda mais próximo com a vantagem de que o nosso algoritmo permite uma implementação paralela. Por fim, note que o ganho de desempenho da BER se manteve constante ao longo das SNRs, possibilitando vastas implementações neste comprimento de código.

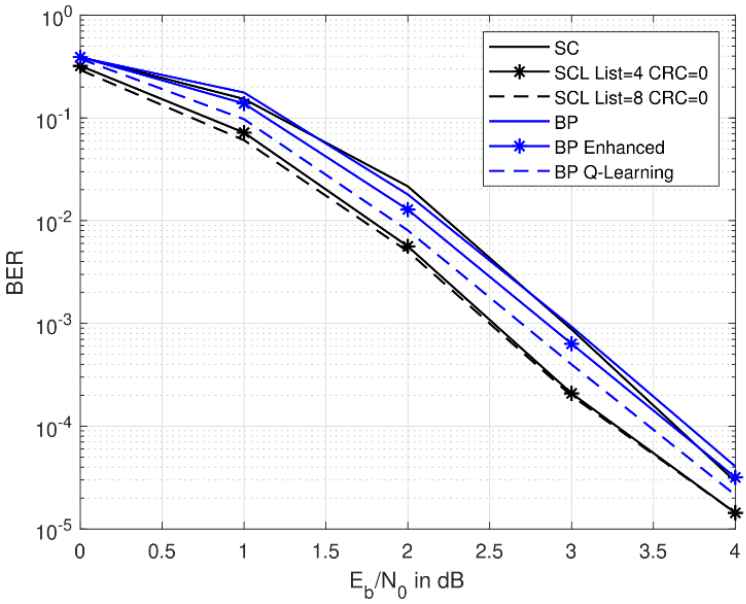


Figura 20: Comparação de BER entre o decodificador SC original de Arikan, decodificador de listas SCL com List-4 e List-8, decodificador BP de Arikan, os decodificadores propostos BP Enhanced e BP Q-learning para N=512 e K=25; sem uso de CRC.

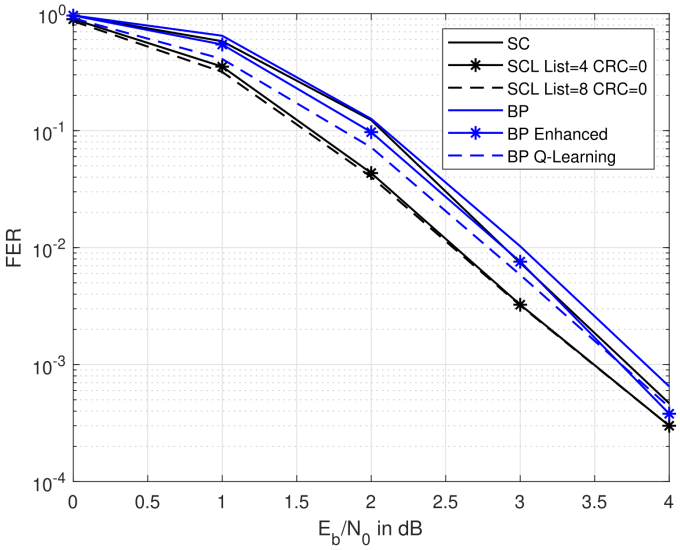


Figura 21: Comparação de FER entre o decodificador SC original de Arikan, decodificador de listas SCL com List-4 e List-8, decodificador BP de Arikan, os decodificadores propostos BP Enhanced e BP Q-learning para N=512 e K=256; sem uso de CRC.

5 Conclusão

Este trabalho desenvolveu o projeto de um decodificador BP baseado em Q-Learning, que busca a melhor ação através de um fator de ponderação para um estado específico ao processar a informação de decodificação. Especificamente, o estudo desenvolvido mostra que a técnica Q-learning aprende a estratégia ótima para maximizar a recompensa total, que equivale a uma decodificação bem sucedida. Desta forma, o decodificador aprende como ponderar cada elemento de processamento. Os resultados das simulações mostram que o desempenho das técnicas de decodificação para códigos polares propostas são superiores às técnicas SC e BP existentes e se aproximam da técnica SCL.

Referências

[1] E. Arikan, “Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels,” *IEEE Transactions on Information Theory*, vol. 55, no. 7, pp. 3051–3073, 2009.

[2] A. TAL, I.; VARDY, “How to construct polar codes,” *IEEE Transactions on Information Theory*, vol. 59, p. 6562–6582, 2013.

[3] K. Niu and K. Chen, “Crc-aided decoding of polar codes,” *IEEE Commun. Lett.*, vol. 16, no. 10, p. 1668–1671, 2012.

[4] M. B. P. A. Balatsoukas-Stimming and A. Burg, “Llr-based successive cancellation list decoding of polar codes,” *IEEE Trans. Signal Process.*, vol. 63, no. 19, p. 5165–5179, 2015.

[5] C. X. J. Lin and Z. Yan, “A high throughput list decoder architecture for polar codes,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 6, p. 2378–2391, 2016.

[6] D. Kim and I.-C. Park, “A fast successive cancellation list decoder for polar codes with an early stopping criterion,” *IEEE Trans. Signal Process.*, vol. 66, no. 18, p. 4971–4979, 2018.

[7] E. Arıkan, “Polar codes: A pipelined implementation,” *Proc. Int. Symp. Broad. Commun. (ISBC)*, p. 11–14, 2010.

[8] A. G. i. F. J. Guo, M. Qin and P. H. Siegel, “Enhanced belief propagation decoding of polar codes through concatenation,” *IEEE Inter. Symp. Inf. Theory (ISIT)*, p. 2987–2991, 2014.

[9] S. C. A. Elkelesh, M. Ebada and S. ten Brink, “Improving belief propagation decoding of polar codes using scattered exit charts,” *IEEE Inf. Theory Workshop (ITW)*, p. 91–95, 2016.

[10] ——, “Flexible length polar codes through graph based augmentation,” *IEEE Inter. ITG Conf. on Syst. Commun. and Coding (SCC)*, pp. 1–6, 2017.

[11] R. M. Oliveira and R. C. de Lamare, “Polarization-driven puncturing for polar codes in 5g systems,” Master’s thesis, Pontifical Catholic University of Rio de Janeiro, 3 2018.

[12] R. M. Oliveira and R. C. De Lamare, “Rate-compatible polar codes based on polarization-driven shortening,” *IEEE Communications Letters*, vol. 22, no. 10, pp. 1984–1987, 2018.

[13] ——, “Puncturing based on polarization for polar codes in 5g networks,” *15th International Symposium on Wireless Communication Systems (ISWCS)*, pp. 1–5, 2018.

[14] ——, “Design of rate-compatible polar codes based on non-uniform channel polarization,” *IEEE Access*, vol. 9, pp. 41 902–41 912, 2021.

[15] E. ARIKAN, “Systematic polar coding,” *IEEE Communications Letters*, vol. 15, p. 860–862, 2011.

[16] K. N. Z. LIU, K. CHEN and Z. HE, “Distance spectrum analysis of polar codes,” *IEEE Wireless Communications and Networking Conference (WCNC)*, vol. 1, p. 490–495, 2014.

[17] E. H. Y. LI, H. ALHUSSIEN and A. JIANG, “A study of polar codes for mlc nand flash memories,” *Int. Conf. on Comput., Netw. and Commun. (ICNC)*, vol. 1, p. 608–612, 2015.

[18] P. G. A. V. C. T. G. SARKIS, I. TAL and W. J.GROSS, “Flexible and low complexity encoding and decoding of systematic polar codes,” *IEEE Trans. Commun*, vol. 64, p. 2732—2745, 2016.

[19] M. I. H. FOSSORIER, M. P. C.; MIHALJEVIC, “Reduced complexity iterative decoding of low-density parity check codes based on belief propagation,” *IEEE Transactions on Communications*, vol. 47, p. 673—680, 1999.

[20] I. V. A. G. W. J. LEROUX, C.; TAL, “Hardware architectures for successive cancellation decoding of polar codes,” *IEEE Int. Conf. on Acou. Speech and Signal Process. (ICASSP)*, vol. 1, p. 1665—1668, 2011.

[21] A. TAL, I.; VARDY, “List decoding of polar codes,” *IEEE Int. Sym. Inf. Theory (ISIT)*, vol. 1, p. 1—5, 2011.

[22] H. S. B. LI and D. TSE, “An adaptive successive cancellation list decoder for polar codes with cyclic redundancy check,” *IEEE Communications Letters*, vol. 16, p. 2044—2047, 2012.

[23] A. P. X. P. K. ZHANG, Q.; LIU, “Crc code design for list decoding of polar codes,” *IEEE Communications Letters*, vol. 21, p. 1229—1232, 2017.

[24] H. MURATA, T.; OCHIAI, “On design of crc codes for polar codes with successive cancellation list decoding,” *IEEE International Symposium on Information Theory (ISIT)*, vol. 1, p. 1868—1872, 2017.

[25] P. V. A. T. C. G. W. J. SARKIS, G.; GIARD, “Increasing the speed of polar list decoders,” *Proc. Sig. Process. Sys. (SiPs) IEEE Workshop*, vol. 16, p. 1—6, 2014.

[26] D. R. WASSERMAN, “Designing polar codes to minimize the ber of crc-aided list decoding,” *Information Theory and Applications Workshop (ITA)*, vol. 1, p. 1—3, 2016.

[27] D. LIN, S.; COSTELLO, *Error Control Coding*. PRENTICE HALL, 2004.

[28] B. Yuan and K. K. Parhi, “Early stopping criteria for energy-efficient low latency belief-propagation polar code decoders,” *IEEE Transactions on Signal Processing*, vol. 62, no. 24, p. 6496—6506, 2014.

[29] H. L. H. Yu and S. Lee, “Early termination belief propagation decoder with parity check matrix for polar codes,” *27th Wireless and Optical Communication Conference (WOCC)*, p. 1—4, 2018.

[30] A. L. Qingshuang Zhang and X. Tong, “Early stopping criterion for belief propagation polar decoder based on frozen bits,” *ELECTRONICS LETTERS*, vol. 53, no. 24, p. 1576—1578, 2017.

[31] C. Courcoubetis and M. Yannakakis, “Markov decision processes and regular events,” *IEEE Transactions on Automatic Control*, vol. 43, no. 10, p. 1399—1418, 1998.

[32] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley Sons, 2014.

[33] G. H. B. Jang, M. Kim and J. W. Kim, “Q-learning algorithms: A comprehensive classification and applications,” *IEEE Access*, vol. 7, p. 133653—133667, 2019.

[34] M. Gimelfarb, S. Sanner, and C.-G. Lee, “Epsilon-bmc: A bayesian ensemble approach to epsilon-greedy exploration in model-free reinforcement learning,” *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, vol. 115, pp. 476–485, 2020.