Short-Length Linear Block Code Decoders for 5G URLLC

組別：B351　　組員：李政翰、潘玠旻　　指導教授：翁詠祿 教授

Introduction

In 5G New Radio, Ultra-Reliable Low-Latency Communication (URLLC) imposes strict demands on latency and reliability. Short-length codes are considered for URLLC in digital communication encoding due to their potential to reduce latency. However, their shorter lengths compromise reliability, necessitating decoders with improved error rate performance and lower decoding complexity. In recent years, machine learning(ML) has flourished, and ML decoding solutions have become a method to enhance reliability. This project focuses on implementing URLLC decoders, specifically exploring and optimizing short-length block code decoders using ML. The study is divided into two parts: Part I introduces the principles and implementation of ML decoders, utilizing Minimum Sum(MS) and Normalized Minimum Sum(NMS) algorithms as the foundation for low-density parity-check(LDPC) codes. ML optimizes decoder weights for enhanced error rate performance. Part II reproduces two short-length code decoding algorithms, Ordered Reliability Bits Guessing Random Additive Noise Decoding (ORBGRAND) and Turbo Decoding for Product Codes, proposing a new decoder under turbo decoding architecture to potentially reduce decoding complexity.

Part I: AI Optimized LDPC Decoder

In this part, we will base our decoders on the MS algorithm[1] and the NMS algorithm[1] and compare with two decoders models that use machine learning-trained weight parameters: the Neural NMS(NNMS)[2] and the Shared-parameter NNMS(SNNMS)[2] algorithm decoders. All four algorithms have the same structure as the MS algorithm, with the difference lying in the weights used for message passing between variable nodes(VN) and check nodes(CN).

Table 1 shows the differences in weights among the four algorithm decoders.

Table 1
Weights of Four Algorithms

Decoders	VN-to-CN weights	CN-to-VN weights
MS	1	1
NMS	0.8	1
NNMS	$\alpha_{ij}^{(l)}$	$\beta_{ij}^{(l)}$
SNNMS	$\alpha^{(l)}$	$\beta^{(l)}$

Table 2
Model Training Parameters

Training data	0dB~5dB, 0.25dB/Step, Totally 21 kinds of data/ Epoch
Loss function	Binary Cross-Entropy
Epoch & learning rate (lr)	Epoch = 10 & lr = 0.001
Iteration	$l_{max} = 10$
Activate function	LeakyReLU (r = 100)

Both the parameters in NNMS are two dimensions and their size are equivalent to the size of the parity check matrix $\mathbf{H}= [h_{i,j}]_{0\le i\le m, 0\le j\le n}$, $h_{i,j} \in$ GF(2). In this part, we use (192,96) LDPC code, whose code rate is 0.5 and $d_v = 6, d_c = 7$. The parameter l represents the iteration. Unlike NNMS, the parameters $\alpha^{(l)}$ in SNNMS are consistent in each iteration and so are $\beta^{(l)}$.

Next, we have to determine training model parameters to improve the BER performance of the decoders . Figure 1, figure 2 and figure 3 show the process of determining the parameters, and Table 2 shows the final training parameters.

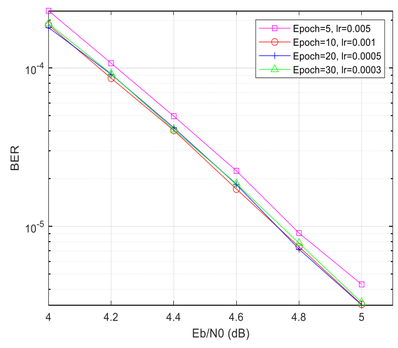

Fig.1 NNMS w/ different epoch and lr

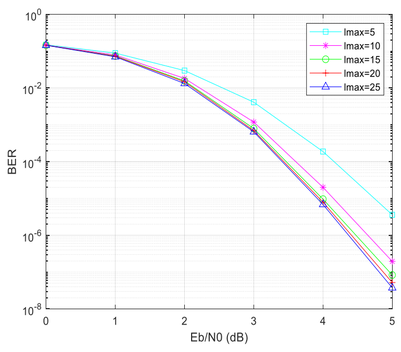

Fig. 2 NNMS w/ different l_{max}

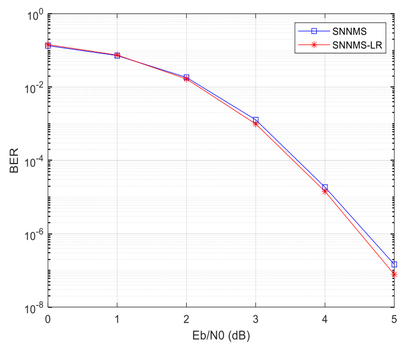

Fig. 3 SNNMS w/ & w/o LeakyReLU

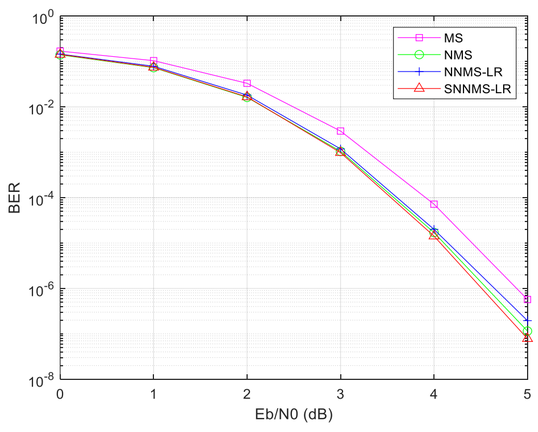

Fig. 4 BER performance of four algorithm decoders

From figure 4, it can be observed that the NNMS decoder, which has a large number of weights and is trained using a neural network, performs worse in terms of error rates compared to the NMS decoder with a single weight.

On the other hand, the SNNMS decoder, which shares parameters in each iteration and obtains weights through neural network training, not only has significantly fewer parameters than NNMS decoder but also has optimal error rate performance. This successful improve the decoding performance through machine learning.

Part II: Short-Length Code Decoders

A. ORBGRAND

In ORBGRAND[3], We attempt to derive the posterior probability of the error pattern $\boldsymbol{z^n}$ to model it. For an element Y_i in the received codeword $\boldsymbol{Y^n}$, its Log-Likelihood Ratio (LLR) is:

$$LLR(Y_i) = log\frac{f_{Y|C}(Y_i|c=1)}{f_{Y|C}(Y_i|c=0)} \quad (1)$$

Where c is the transmitted codeword $\boldsymbol{c^n}$. By further derivation, we know that the posterior probability of is related to the LLR as (2):

$$P(\boldsymbol{Z^n} = \boldsymbol{z^n}) \propto \exp(-\sum_{i=1}^{n} |LLR(Y_i)|z_i) \quad (2)$$

Reliability is defined as: $Rel(\boldsymbol{z^n}) = \sum_{i=1}^{n} |LLR(Y_i)|\, z_i$

By (2), we know that the error pattern $\boldsymbol{z^n}$ with the minimum reliability is the most probable occurrence of errors. In this way, we can identify a set of possible $\boldsymbol{z^n}$ and arrange them in ascending order based on their reliabilities. During decoding, these $\boldsymbol{z^n}s$ are sequentially added to the received codeword, and the decoded message $\boldsymbol{c^n} = \boldsymbol{y^n} \oplus \boldsymbol{z^n}$.

In basic ORBGRAND, we use a linear model $\boldsymbol{\lambda^n}$ to approximate $Rel(\boldsymbol{z^n})$:

$$Rel(\boldsymbol{z^n}) \approx \sum_{i:z_i=1} \lambda_i = \beta \sum_{i=1}^{n} iz_i = \beta * w_L(\boldsymbol{z^n}) \quad (3)$$

Define the logistic weight $w_L(\boldsymbol{z^n}) = \sum_{i=1}^{n} iz_i$ as the sum of the indices of error positions in the $\boldsymbol{z^n}$ and the Hamming Weight , $w_H(\boldsymbol{z^n})$. Then, we have the set combine all possible $\boldsymbol{z^n}$:

$$S_W = \bigcup_{w=1}^{\lfloor\sqrt{1+8W}-1/2\rfloor} \{\boldsymbol{z^n} \in \{0,1\}^n, w_H(\boldsymbol{z^n}) = w, w_L(\boldsymbol{z^n}) = W\} \quad (4)$$

Utilizing the Landslide Algorithm[3], we can easily obtain $\boldsymbol{z^n}$. The flowchart is depicted in Fig. 5, and the simulation results are presented in Fig. 6, 7.

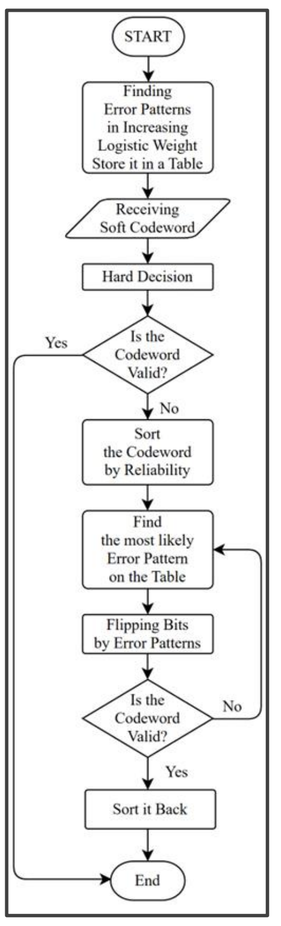

Fig. 5 ORBGRAND flowchart

B. Turbo Decoding for Product Codes

The product code turbo decoder (PTD) [4][5] includes row(column) decoders, and within the decoder, the Chase decoder is employed to find a candidate codeword $\hat{T}$. Subsequently, the decoder proceeds to calculate the extrinsic information L_i^e. Decoding is completed by hard decision on likelihood $L_{row}(L_{col})$,obtained by the exchange of extrinsic info. , in collaboration with iterations. Where α_{iter} is positively correlated with the number of iterations.

$$L_{row}(\hat{T}_i) = Y_i + \alpha_{iter}L_{rc,i}^e + \alpha_{iter}L_{cr,i}^e \quad (5)$$
$$L_{col}(\hat{T}_i) = Y_i + \alpha_{iter}L_{cr,i}^e + \alpha_{iter}L_{rc,i}^e \quad (6)$$

The simulation results are presented in Fig. 8.

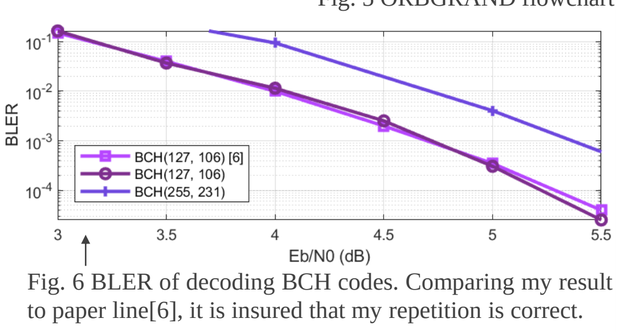

Fig. 6 BLER of decoding BCH codes. Comparing my result to paper line[6], it is insured that my repetition is correct.

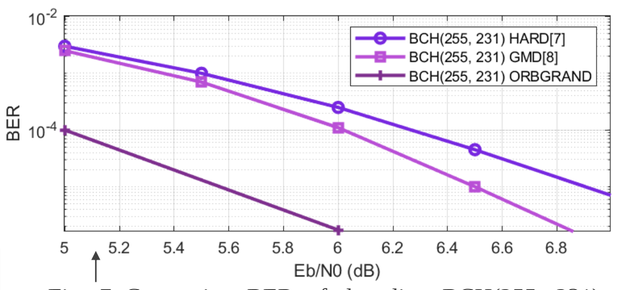

Fig. 7 Comparing BER of decoding BCH(255, 231) between Hard decoding[7], Generalized Minimum Distance decoding(GMD)[8], and ORBGRAND. Getting 1dB gain by using ORBGRAND.

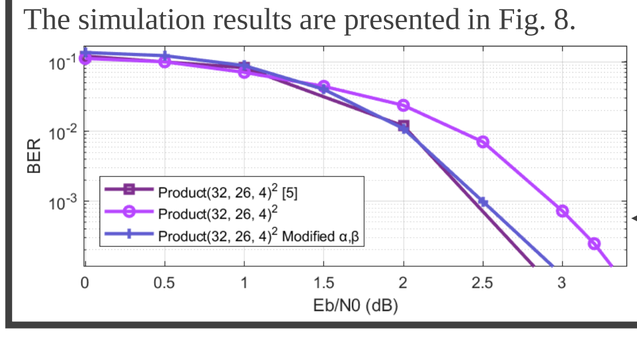

Fig. 8 Bit Error Rate of decoding Product codes.
Let p = 8, max iteration=4, comparing to paper, we have gotten the circle line. After slightly adjusting α_{iter}, β_{iter}, the result is matched the line from the paper[5].

Future Work

After reproducing two papers in Part II, it is possible and valuable to replace the two row(column) decoders in the implemented product code turbo decoder (PTD) with the ORBGRAND decoder. Through the reproduction process, we observed that both ORBGRAND and PTD have a commonality in generating error patterns. ORBGRAND uses the landslide algorithm, while the PTD employs the Chase decoder. Analyzing the two decoders: the advantage of ORBGRAND lies in the landslide algorithm, which pre-generates an error pattern table for efficient error checking. However, its drawback is the need for a large number of comparisons when dealing with low SNR. On the other hand, the advantage of PTD is the use of the Chase decoder, which iteratively decodes without the need to generate numerous error patterns as in ORBGRAND. The disadvantage is the complexity arising from generating error patterns on the fly during decoding.

By replacing the two row(column) decoders in the implemented PTD with the ORBGRAND decoder, the combined advantages of both decoders can be leveraged. This approach utilizes the landslide algorithm to generate multiple codewords, which are then decoded iteratively using the PTD. This integration is expected to reduce decoding complexity, as it combines the efficiency of the landslide algorithm with the iterative decoding process of the PTD.

Conclusion

This project explores potential machine learning(ML) improvements for decoders in URLLC. In Part I, we aimed to familiarize with the principles and implementation of ML decoders. We developed a ML-optimized short-length LDPC decoder based on the MS and NMS. Results indicated that the SNNMS outperformed the NNMS in error rate performance, potentially due to overfitting in the latter. In Part II, we reproduced two short-length code decoders: ORBGRAND decoder and PTD. Leveraging the advantages of ORBGRAND in pre-processing error patterns and compensating for its drawback of requiring numerous comparisons in the presence of low SNR, we proposed replacing row(column) decoders in the implemented PTD with the ORBGRAND decoder. The sensitivity of the error rate performance of PTD to the weights of transmitted messages suggests the need for ML to find optimal weights.

Due to time constraints, the proposed new decoder and its ML optimization were not implemented in this project. However, our analysis and literature review provide valuable insights into understanding relevant technologies and their potential impact on the decoding process. These insights serve as a valuable starting point for future research.

Reference:
[1] Jinghu Chen and Marc P. C. Fossorier, “Density Evolution for Two Improved BP-Based Decoding Algorithms of LDPC Codes,” *IEEE Trans. On Commun*. vol. 6, no.5, pp.208-210, May 2002.
[2] Qing Wang, Qing Liu, Shunfu Wang, Leian Chen, Haoyu Fang, Luyong Chen, Yuzhang Guo, and Zhiqiang Wu, “Normalized Min-Sum Neural Network for LDPC Decoding”, *IEEE Trans. On Cognitive Commun*. vol. 9, no.1, pp.70-81, Feb 2023.
[3] K. R. Duffy, W. An and M. Médard, "Ordered Reliability Bits Guessing Random Additive Noise Decoding," *IEEE Transactions on Signal Processing*, vol. 70, pp. 4528-4542, 2022, doi: 10.1109/TSP.2022.3203251.
[4] Ryan, W., & Lin, S. (2009). Turbo Codes. In *Channel Codes: Classical and Modern* (pp.298-338). Cambridge: Cambridge University Press. doi:10.1017/CBO9780511803253.008
[5] R. M. Pyndiah, "Near-optimum decoding of product codes: block turbo codes," *IEEE Transactions on Communications*, vol. 46, no. 8, pp. 1003-1010, Aug. 1998, doi: 10.1109/26.705396.
[6] K. R. Duffy, "Ordered Reliability Bits Guessing Random Additive Noise Decoding," *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Toronto, ON, Canada, 2021, pp. 8268-8272, doi: 10.1109/ICASSP39728.2021.9414615.
[7] Lin, Yi-Min, Lee, Chen-Yi, & Chang, Hsie-Chia (2010). Area-Efficient Soft BCH and RS Decoders. http://hdl.handle.net/11536/40709.
[8] G. Forney, "Generalized minimum distance decoding," in *IEEE Transactions on Information Theory*, vol. 12, no. 2, pp. 125-131, April 1966, doi: 10.1109/TIT.1966.1053873.