An Algorithm for Finding an Approximate Reliability Sequence for Polar Codes on the BEC

Saeid Ghasemi and Bartolomeu F. Uchôa-Filho

***Abstract*— One of the most important parts of encoding and decoding in polar codes is finding the bit-channel reliability sequence, i.e., the positions of frozen bits and data bits, which has a great effect on the error correction performance. In this paper, we present an algorithm with linear complexity for determining an approximate reliability sequence in small disagreement with the true reliability sequence on the binary erasure channel (BEC).**

***Keywords*— Polar codes, reliability sequence, binary erasure channel.**

I. Introduction

Polar codes, proposed by Arikan in 2009 [1], have attracted a lot of attention as they achieve the symmetric capacity on binary-input discrete memoryless channels under successive cancellation (SC) decoding. Polar coding is based on a phenomenon called channel polarization. After properly combining the input bits of several copies of the channel in a parallel concatenation scheme, the bit-channels present different reliabilities. Asymptotically, as the number of copies goes to infinity, a fraction of the bit-channels (which equals the channel capacity) becomes noiseless, while the remaining bit-channels are rendered useless. Encoding amounts to sending the data bits through the good bit-channels and the so-called frozen bits (usually zeros) through the bad bit-channels.

Deciding about the reliable and unreliable bit positions is always challenging. The reliabilities of all (say, N) bit-channels can be first computed. Then, they can be ordered in increasing order of reliability. Finally, a rate $R = K/N$ polar code can be obtaining by selecting the K most reliable (rightmost) bit-channels for the data. The sequence of bit-channel positions ordered in increasing order of reliability is called the reliability sequence (RS).

The problem is that, in general, there is no low-complexity algorithm for obtaining the bit-channel reliabilities. The exception is the binary-erasure channel (BEC), for which a recursive formula exists and the reliabilities can be obtained with linear complexity $\mathcal{O}(N)$. Still, in general, ordering has complexity $\mathcal{O}(N\log(N))$. Therefore, a procedure that produces the RS or an approximate RS, even for the simplest channel of all, i.e., the BEC, is of great interest.

A wide range of approximate construction methods are proposed in [1]-[7]. In general, the reliability order of the bit-channels depends on the channel conditions and on the code length, and therefore is not universal. Several methods to design the frozen sets on-the-fly with limited complexity have been proposed [8].

In this paper, we present an algorithm with linear complexity that specifies an approximate RS on the BEC for any arbitrary length that is a power of two, regardless of the erasure probability. Although no formal proof is presented, we show by means of examples that the number of disagreements between the approximate and the true RS is quite small. Moreover, by fixing the code rate, i.e., by setting a borderline between the data set and the frozen set, we realize that the swapped bit-channels are very close to the borderline. In other words, their bit-channel reliabilities are quite similar and little impact on the error performance is expected by adopting the approximate RS instead of the true one.

This paper is organized as follows. In Section II, we briefly review the encoding and decoding principles for polar codes. In Section III, we talk about the concept of RS. In Section IV, we present the proposed algorithm. In Section V, we discuss the performance of the proposed algorithm by evaluating the number of bit-channel swaps (between the frozen set and the information set). Finally, in Section VI, we conclude the paper.

II. Encoding and Decoding Principles for Polar Codes

In polar codes, the codeword length is $N = 2^p$, $p = 1, 2, 3, \ldots$. The length of the information block is K and, consequently, there are $N - K$ frozen bits. Coding in polar codes consists of two separated steps. The first step is the selection of the positions of frozen bits and data bits, combining them together so that the source vector $\mathbf{u} \in \mathbb{F}_2^N$ results, in which $u = (u_1, u_2, \ldots, u_N)$. In the second step, the source vector $\mathbf{u}$ is mapped to codeword $\mathbf{x}$ as follows

$$\mathbf{x} = \mathbf{u}\mathbf{G}_N = \mathbf{u}\mathbf{B}_N\mathbf{F}^{\otimes N}, \tag{1}$$

where $\mathbf{G}_N$ is the generator matrix of order N, $\mathbf{B}_N$ is the bit-reversal permutation matrix and $\mathbf{F}^{\otimes N}$ is related to the n-th Kronecker product of the kernel matrix $\mathbf{F}$:

$$\mathbf{F}^{\otimes N} = \mathbf{F} \otimes \mathbf{F}^{\otimes N-1}, \mathbf{F} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}. \tag{2}$$

The encoding process for $N = 8$ and $K = 4$ is shown in Figure 1. We have shown the positions of the frozen bits with a blue F next to the corresponding channel. In the next section we will explain how we can obtain these positions.

Decoding in polar codes is done by successive cancellation (SC) with the complexity $\mathcal{O}(N\log(N))$. SC decoding can be

Saeid Ghasemi, Department of Electrical and Electronics Engineering, Federal University of Santa Catarina, Florianopolis-SC, e-mail: saeid.ghasemi.7631@gmail.com; Bartolomeu F. Uchôa-Filho, Department of Electrical and Electronics Engineering, Federal University of Santa Catarina, Florianopolis-SC, e-mail: uchoa@eel.ufsc.br.

This work has been supported by CNPq (Brazil) under grants 141946/2019-9 and 306780/2017-8.

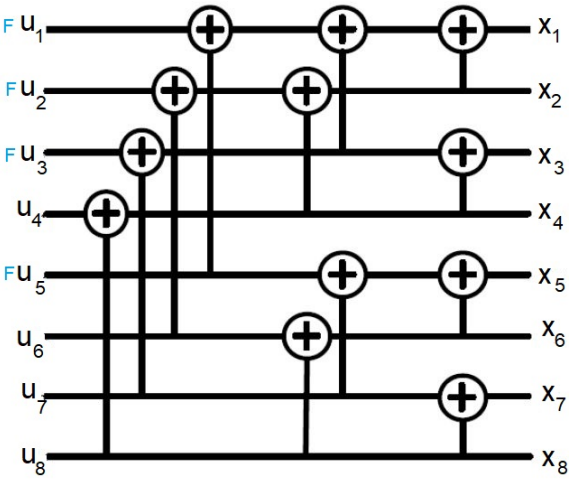

Fig. 1. Polar coding for $N = 8$, $K = 4$, and $R = \frac{1}{2}$. Frozen channels are shown in blue F.

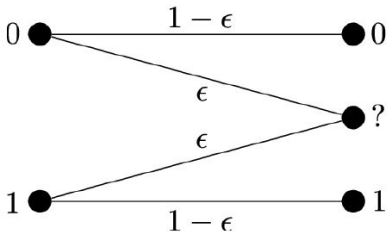

Fig. 2. Binary erasure channel with erasure probability ϵ.

viewed as a soft/hard message. passing algorithm over the trellis of polar code. More details are available in [9] and [10].

III. The RS of a BEC

Consider the BEC W shown in Figure 2, where ϵ is the erasure probability. It is well-known that this channel has capacity $I(W) = 1 - \epsilon$. Combining two independent uses of the BEC, a compound channel (W, W) is obtained. Obviously, its associated capacity is $2I(W)$. By applying the chain rule of mutual information, this compound channel can be decomposed into two synthesized channels. After channel combining and channel splitting, two independent BECs with the same reliability are transformed into two polarized channels and the sum capacity of the two channels is unchanged, that is, $I(W^-)+I(W^+) = 2I(W)$. In [1], Arikan derived the mutual information of the two channels as $I(W^+) = 2I(W)-I(W)^2$, $I(W^-) = I(W)^2$, and proved that the bad channel W^- has a smaller capacity than the given BEC W, whereas the good channel W^+ has higher capacity.

In fact, this process is the basis of channel polarization. If we do this process more and more times, we obtain more channels showing a tendency to approach either zero or unit capacity. In the end, we sort all obtained capacities from the smallest to the largest and we have the RS. In Figure 3, the different bit-channel capacities are shown after polarization, from which we can determine the RS.

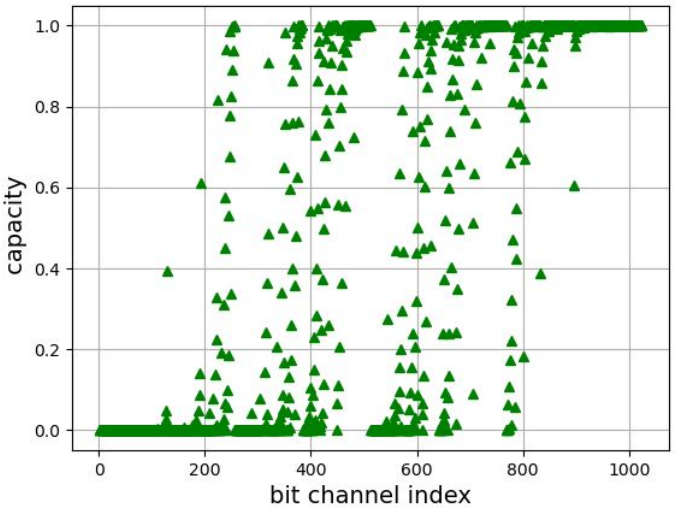

Fig. 3. All capacities after polarization over BEC ($\epsilon = 0.5$) for $N = 1024$.

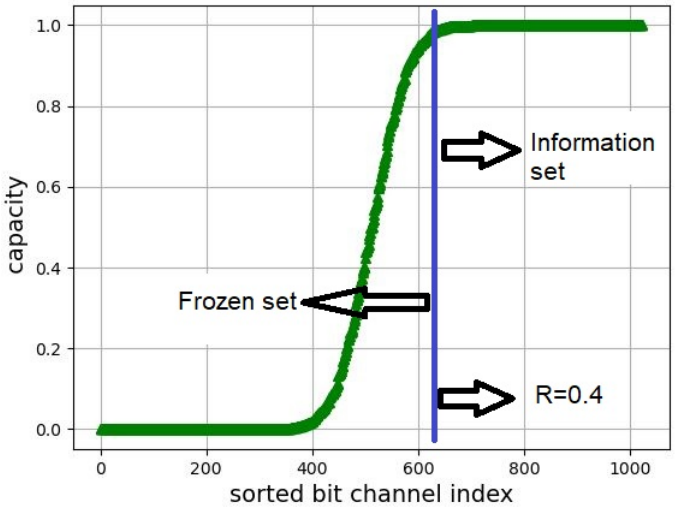

Fig. 4. Selection of frozen set and data set from the sorted capacities according to the code rate for $N = 1024$. Code rate $R = 0.4$ is shown as an example.

According to this sequence, we can determine which positions are more reliable and consequently have probabilities of erasure smaller than other positions. Therefore, we allocate data bits to these positions and frozen bits to the rest of the positions in which the probability of erasure is higher. In fact, we consider all frozen bits as zero and when we want to do decoding, we already know that frozen bits are zero and we just have to do decoding for the rest of the positions.

In Figure 4, we sorted all capacities obtained from Figure 3. The reliability of transmission depends on the code rate. For instance, by choosing a small rate, the transmission is very reliable, because we only use those channels with high capacity that are located to the right of the blue line and we consider the rest of the channels as frozen channels. As the code rate increases, we have to use some low capacity channels.

For example, suppose that we know for $N = 8$, the RS is $\{1, 2, 3, 5, 4, 6, 7, 8\}$ (channel number 1 is the worst channel

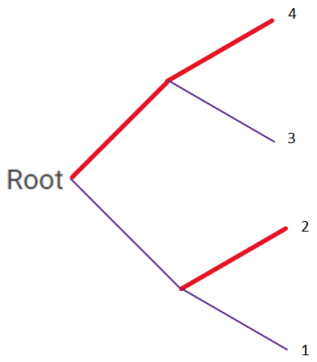

Fig. 5. Finding RS according to direction

and channel number 8 is the best one). The positions of the four noisiest channels in Fig. 1 are marked with an F. If we want to send four data bits, say $\{1, 0, 1, 1\}$, due to the frozen bits we have to do encoding for the following source vector:

$$\mathbf{u} = [\overbrace{0}^{F}, \overbrace{0}^{F}, \overbrace{0}^{F}, 1, \overbrace{0}^{F}, 0, 1, 1]. \tag{3}$$

IV. The Proposed Algorithm

How does our algorithm work? Suppose that we want to obtain the RS for a polar code with length N. We know that N is a power of 2. We let $N = 2^p$, $p = 1, 2, \dots$. Obviously, the RS for $N = 2$ is $\{1, 2\}$. We use this RS to obtain the RS for $N = 4$. By using RS for $N = 4$, we can achieve RS for $N = 8$, and so on.

In this algorithm, we assign all numbers from 1 to N to some vectors, according to how many times we need to go down (blue lines in Figure 5) from root to the leaf. As a simple example, for $N = 4$, based on Figure 5 we have

$$\begin{cases} A_0^{(4)} = [1], \\ A_1^{(4)} = [2, 3], \\ A_2^{(4)} = [4]. \end{cases} \tag{4}$$

In this example, $A_0^{(4)}$ shows all the numbers that we can achieve by going down two times, while $A_1^{(4)}$ represents all numbers that we obtain by going down one time, and $A_2^{(4)} = [4]$ is for going down zero time. Therefore, in this case, the RS from the worst to the best channel is $\{1, 2, 3, 4\}$. The basic issue in this algorithm is that, if we go down fewer times, we have a channel with higher capacity.

We now give a general description of the proposed algorithm. Let $N = 2^p$, $p = 1, 2, \dots$ be the codeword length of the polar code. The proposed approximate RS is denoted by the vector $\mathbf{A}^{(N)}$, whose elements are the claimed ordered bit-channel positions. The approximate RS is given by

$$\mathbf{A}^{(N)} = [\mathbf{A}_0^{(N)} \cdots \mathbf{A}_p^{(N)}], \tag{5}$$

where $\mathbf{A}_i^{(N)}$ for $i = 0, \dots, p$ is a sub-vector of size $|\mathbf{A}_i^{(N)}| = \binom{p}{i}$ whose elements are obtained recursively from the sub-vectors $\mathbf{A}_i^{(N/2)}$ and $\mathbf{A}_{p-i}^{(N/2)}$ as we describe next.

1) For any $N = 2^p$, the 1-st and the p-th sub-vectors are given by
$$\mathbf{A}_0^{(N)} = [1] \quad \text{and} \quad \mathbf{A}_p^{(N)} = [N]. \tag{6}$$
So, for $p = 1$, we have
$$\mathbf{A}^{(2)} = [1, 2], \tag{7}$$
where $\mathbf{A}_0^{(2)} = [1]$ and $\mathbf{A}_1^{(2)} = [2]$;
2) For $p > 1$, and for $i = 1, \dots, p-1$, we have
$$\mathbf{A}_i^{(N)} = [\mathbf{A}_i^{(N/2)} \quad \mathbf{B}_i^{(N)}], \tag{8}$$
where $|\mathbf{B}_i^{(N)}| = |\mathbf{A}_{p-i}^{(N/2)}| = \binom{p-1}{p-i} := z$.
The sub-vector $\mathbf{B}_i^{(N)}$ is given by
$$\mathbf{B}_i^{(N)} = [b_{i,1}^{(N)} \dots b_{i,j}^{(N)} \dots b_{i,z}^{(N)}], \tag{9}$$
where
$$b_{i,j}^{(N)} = (N+1) - a_{p-i,z-j+1}^{(N/2)}. \tag{10}$$

As an example, in the following, we show the method for obtaining these sequences for $N = 4, 8, 16$. We consider $[1, 2]$ as trivial RS for $N = 2$. In any length, we have shown numbers from previous length with P.L above numbers. For $N = 4$

$$N = 4 : \begin{cases} A_0^{(4)} = \overbrace{1}^{\text{P.L}} \\ A_1^{(4)} = \overbrace{2}^{\text{P.L}}, 3 \\ A_2^{(4)} = 4, \end{cases} \Rightarrow RS = [1, 2, 3, 4]. \tag{11}$$

For $N = 8$

$$N = 8 : \begin{cases} A_0^{(8)} = \overbrace{1}^{\text{P.L}} \\ A_1^{(8)} = \overbrace{2, 3}^{\text{P.L}}, 5 \\ A_2^{(8)} = \overbrace{4}^{\text{P.L}}, 6, 7 \\ A_3^{(8)} = 8 \end{cases} \Rightarrow RS = [1, 2, 3, 5, 4, 6, 7, 8]. \tag{12}$$

For $N = 16$

$$N = 16 : \begin{cases} A_0^{(16)} = \overbrace{1}^{\text{P.L}} \\ A_1^{(16)} = \overbrace{2, 3, 5}^{\text{P.L}}, 9 \\ A_2^{(16)} = \overbrace{4, 6, 7}^{\text{P.L}}, 10, 11, 13 \\ A_3^{(16)} = \overbrace{8}^{\text{P.L}}, 12, 14, 15 \\ A_4^{(16)} = 16 \end{cases}$$
$$\Rightarrow RS = [1, 2, 3, 5, 9, 4, 6, 7, 10, 11, 13, 8, 12, 14, 15, 16]. \tag{13}$$

For example, in the latter case, the best channel is channel number 16 and the worst is channel number 1.

As we can see, the complexity of this algorithm is $\mathcal{O}(N)$, while in the original algorithm of Arikan [1], because it is a sorting problem, the complexity is $\mathcal{O}(N \log(N))$.

1|2|3|5|9|17|33|4|6|7|10|11|13|18|19|21|25|34|35|37|41|49|8|12|14|**20**|**15**|22|23|26|**36**|**27**|**38**|**39**|**29**|42|43|45|50|51|53| **16**|**57**|24|28| **40**|**30**|**44**|**31**|46|**52**|**47**|54|55|58|59|**32**|**61**|48|56|60|62|63|64
1|2|3|5|9|17|33|4|6|7|10|11|13|18|19|21|25|34|35|37|41|49|8|12|14|**15**|**20**|22|23|26|**27**|**29**|**36**|**38**|**39**|42|43|45|50|51|53| **57**|**16**|24|28| **30**|**31**|**40**|**44**|46|**47**|**52**|54|55|58|59|**61**|**32**|48|56|60|62|63|64

Fig. 6. Reliability sequences for $N = 64$ (top: proposed, bottom: Arikan's with $\epsilon = 0.05$). Disagreements are shown in red.

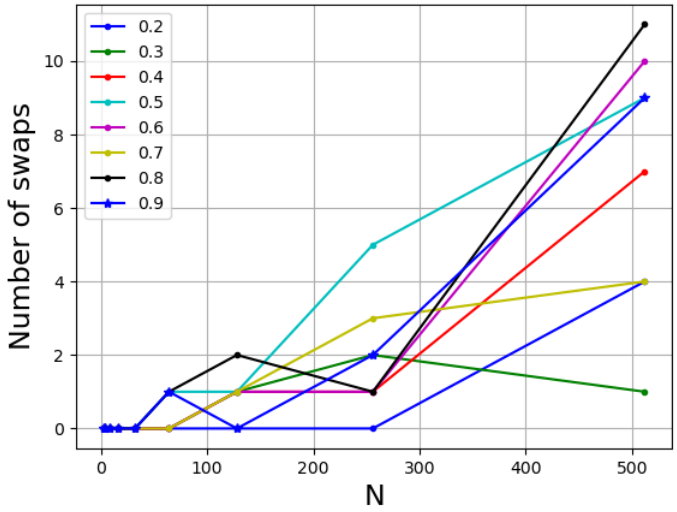

Fig. 7. Number of bit-channel swaps for erasure probability $\epsilon = 0.05$.

V. Swapped Bit-Channels

To evaluate the efficiency of our algorithm, we compare our results with Arikan's (i.e., the true) RS as follows. According to the code rate, we split all bit-channels into two sets: the frozen set and the information set. Then, we count the number of indices swaps between these sets and compare this number with the maximum possible number of swaps, namely, $\min(R, 1-R) \times N$.

In Figure 7, we can see the number of swaps between the frozen set and the data set for $N = 2, 4, 8, \ldots, 512$, $R = 0.2, 0.3, \ldots, 0.9$, and erasure probability $\epsilon = 0.05$.

TABLE I
Efficiency (Related to Fig. 7)

	$R=0.2$	$R=0.3$	$R=0.4$	$R=0.5$	$R=0.6$	$R=0.7$	$R=0.8$	$R=0.9$
$N=64$	100	100	100	96.88	100	100	92.19	84.38
$N=128$	100	97.4	98	98.44	98.05	97.4	92.19	100
$N=256$	100	97.4	99	96.09	99.02	96.1	98.05	92.19
$N=512$	96.1	99.35	96.6	96.48	95.11	97.4	89.26	82.2

In Table I, related to Figure 7, we can see the efficiency of the proposed algorithm. For example, if $N = 256$, $R = 0.3$ (green line in Figure 7), we can see that there are two swaps. So, the efficiency is

$$\frac{\overbrace{2}^{\text{our swaps}}}{\underbrace{(R \times N)}_{\text{max swaps}}} = \frac{100 - E}{100} \Rightarrow E = 100 - \frac{100 \times 2}{(0.3 \times 256)} = 97.4\%$$

Due to the bit swaps between the data set and the frozen set, a few information bits may be sent over frozen bit-channels. Therefore, we evaluate the impact of adopting the proposed RS on the performance of the polar code. The interesting issue related to the swapped bit-channels is that their capacities are very similar. In fact, we have some swaps between those bit-channels close to the border between the two sets, and this does not affect the performance of transmission and error correction significantly.

For example when $N = 256$, $R = 0.3$, and $\epsilon = 0.05$ (green line in Figure 7), our swaps are bit-channels number 139 and 141. In fact, we have to consider 139 and 141 as frozen bits (channels with capacity near zero), but in our algorithm instead of these two channels, we considered 105 and 113 as frozen bits. If we compare their capacities, it is clear that the difference in channel capacity is very small, as follows:

$$\begin{array}{rcrcrcr} 139 & \leftrightarrow & 105, & & 2.85 \times 10^{-31} & \leftrightarrow & 2.4 \times 10^{-30}, \\ 141 & \leftrightarrow & 113, & & 1.14 \times 10^{-30} & \leftrightarrow & 5.7 \times 10^{-28} \end{array}.$$

As another example, for $N = 256$, $R = 0.7$, and $\epsilon = 0.05$, with respect to Figure 7, we have three swaps:

$$\begin{array}{rcrcrcr} 144 & \leftrightarrow & 233, & & 1.3 \times 10^{-7} & \leftrightarrow & 4.36 \times 10^{-6} \\ 152 & \leftrightarrow & 241, & & 2.61 \times 10^{-7} & \leftrightarrow & 9.32 \times 10^{-5} \\ 156 & \leftrightarrow & 111, & & 5.23 \times 10^{-7} & \leftrightarrow & 6.30 \times 10^{-7} \end{array}$$

As an obvious result, the swapped channels having close capacities, is that the overall capacity loss due to swapping is negligible. In other words, our algorithm does not promote a swap between a very good channel and a rather poor channel.

Finally, it is also important to observe the discrepancy between the proposed and Arikan's sequences in terms of bit position switching within the information set. In Figure 6, we make this comparison with respect to the length $N = 64$. It is clear that all switchings occur within short blocks of symbols, meaning, again, that the reliabilities of the switched bit positions are very similar. As a result, very little impact on the decoder's performance is expected.

VI. Conclusion

In this work, we have described a new method to obtain the positions of frozen and data bits with the minimum possible complexity $\mathcal{O}(N)$ that we can apply for transmission in the binary erasure channel, regardless of the erasure probability. The so called RS produced by the algorithm is very close to the true one, which is usually obtained in the literature with complexity $\mathcal{O}(N \log(N))$.

It should be mentioned that all the results and conclusions in this paper are based on short examples. Further investigation would be necessary in order to prove more general results. For instance, it would be important to know whether the same good efficiency (in terms of bit swaps) and bit switching behaviour

are also achieved for large N. Of course, this is of paramount importance and should be considered as future work. Finally, bit error rate performance should be evaluated to support our claims in the paper.

REFERENCES

[1] Erdal Arikan, "Channel polarization: a method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Transactions on Information Theory*, 3051 (DOI: 10.1109/TIT.2009.2021379). 2009.

[2] I. Tal and A. Vardy, "How to construct polar codes," *IEEE Transactions on Information Theory,*vol. 59, no. 10, pp. 6562–6582, 2013.

[3] P. Trifonov, "Efficient design and decoding of polar codes," *IEEE Transactions on Communications,* vol. 60, no. 11, pp. 1–7, November 2012.

[4] Y. Zhang, A. Liu, K. Pan, C. Gong, and S. Yang, "A practical construction method for polar codes," *IEEE Communication Letters,* vol. 18, no. 11, pp. 1871–1874, November 2014.

[5] D. Wu, Y. Li, and Y. Sun,"Construction and block error rate analysis of polar codes over awgn channel based on gaussian approximation," *IEEE Communication Letters,*,vol. 18, no. 7, pp. 1099–1102, July 2014.

[6] G. Bonik, S. Goreinov, and N. Zamarashkin, "Construction and analysis of polar and concatenated polar codes: practical approach," arXiv:1207.4343 [cs.IT], pp. 1–17, 2012.

[7] Haotian Zheng , Seyyed Ali Hashemi , Bin Chen , Zizheng Cao, and A. M. J. Koonen, "Inter-Frame Polar Coding With Dynamic Frozen Bits," *IEEE Communication Letters,*, Vol. 23, NO. 9, september 2019.

[8] H. Vangala, E. Viterbo, and Y. Hong, "A comparative study of polar code constructions for the AWGN channel," *in arXiv preprint arXiv:1501.02473*, Jan. 2015.

[9] V. Bioglio, C. Condo and I. Land, "Design of Polar Codes in 5G New Radio," *IEEE Communications Surveys Tutorials,* vol. 23, no. 1, pp. 29-40, First quarter 2021, doi: 10.1109/COMST.2020.2967127.

[10] Kai Niu, Kai Chen, Jiaru Lin, and Q. T. Zhang "Polar Codes: Primary Concepts and Practical Decoding Algorithms," *IEEE Communications Magazine*, Jul. 2014.