On Finite-Length Performance of Polar Codes: Stopping Sets, Error Floor, and Concatenated Design

A. Eslami, *Student Member, IEEE*, and H. Pishro-Nik, *Member, IEEE*

***Abstract*—This paper investigates properties of polar codes that can be potentially useful in real-world applications. We start with analyzing the performance of finite-length polar codes over the binary erasure channel (BEC), while assuming belief propagation as the decoding method. We provide a stopping set analysis for the factor graph of polar codes, where we find the size of the minimum stopping set. We also find the girth of the graph for polar codes. Our analysis along with bit error rate (BER) simulations demonstrate that finite-length polar codes show superior error floor performance compared to the conventional capacity-approaching coding techniques. In order to take advantage from this property while avoiding the shortcomings of polar codes, we consider the idea of combining polar codes with other coding schemes. We propose a polar code-based concatenated scheme to be used in *Optical Transport Networks* (OTNs) as a potential real-world application. Comparing against conventional concatenation techniques for OTNs, we show that the proposed scheme outperforms the existing methods by closing the gap to the capacity while avoiding error floor, and maintaining a low complexity at the same time.**

***Index Terms*—Polar codes, concatenated codes, belief propagation, stopping sets, error floor.**

I. Introduction

SINCE their introduction, polar codes have attracted a lot of attention among researchers due to their capability to solve some problems (sometimes open problems) that could not be handled using other schemes. However, theoretical approaches have been mostly taken toward polar codes in the literature. Our goal is to study polar codes from a practical point of view to find out about properties that can be useful in real-world applications. Hence, we are mainly concerned with the performance of polar codes in the finite regime (i.e. with finite lengths) as opposed to the asymptotic case. Some of the previous work related to finite-length polar codes include [1]–[9]. Particularly, [2] proposes a successive cancellation list decoder that bridges the gap between successive cancellation and maximum-likelihood decoding of polar codes. Inspired by [2], [10]–[12] propose using CRC along with list decoding to improve the performance of polar codes. [3] presents a method to improve the finite-length performance of successive cancellation decoding by means of simple and short inner block codes. A linear program (LP) decoding for polar codes is considered in [5]. In [7], a method for efficient construction of polar codes is presented and analyzed. In addition, scaling laws are provided in [13]–[17] for the behavior of polar codes that, in some cases, have finite-length implications.

Since an analysis in the finite regime can be very difficult in general, we start with studying the performance of polar codes over the binary erasure channel (BEC). While being fairly manageable, such an analysis leads to a better understanding of the behavior of polar codes. We provide an analysis of the stopping sets in the *factor graph realization* of polar codes. Such a realization for polar codes was first employed by [18] and [19] to run Belief Propagation (BP) as the decoding algorithm. Stopping sets are important as they contribute to the decoding failure and error floor, when BP is used for decoding [20]. Particularly, in the case of BEC, stopping sets are the sole reason of the decoding failure. We find the structure of the minimum stopping set and its size, called *stopping distance*. We will show that the stopping distance grows polynomially for polar codes. This is a clear advantage over capacity-approaching LDPC codes. We also find the girth of the factor graph of polar codes, showing that polar codes hold a relatively large girth. The effect of such a large girth and stopping distance on the error floor behavior of polar codes is depicted in our simulation results for the binary erasure and AWGN (Additive White Gaussian Noise) channels.

It is well-known that finite-length polar codes show poor error probability performance when compared to some of the existing coding schemes such as LDPC and Turbo codes. Nevertheless, showing a set of good characteristics such as being capacity-achieving, low encoding and decoding complexity, and good error floor performance suggests that a combination of polar coding with another coding scheme could eliminate shortcomings of both, hence providing a powerful coding paradigm. In this paper, we consider the design of polar code-based concatenated coding schemes that can contribute to closing the gap to the capacity. Concatenated coding has been studied extensively for different combinations of coding schemes. Furthermore, there have been many applications, such as deep space communications, magnetic recording channels, and optical transport systems that use

Manuscript received October 15, 2011; revised July 2, and September 20, 2012. The associate editor coordinating the review of this paper and approving it for publication was S.-Y. Chung.

The authors are with the Electrical and Computer Engineering Department, University of Massachusetts, Amherst, MA, USA (e-mail: {eslami, pishro}@ecs.umass.edu).

The material in this paper was presented in part at the International Symposium of Information Theory (ISIT), 2011, and the 48th Annual Allerton Conference on Communication, Control, and Computing, 2010.

This work was supported by the National Science Foundation under grants CCF-0830614 and ECCS-0636569.

Digital Object Identifier 10.1109/TCOMM.2013.012313.110692

0090-6778/13$31.00 © 2013 IEEE

a concatenated coding scheme [21]–[24]. A coding scheme employed in these applications needs to show strong error correction capability. Here, we investigate the potentials of using polar codes in a concatenated scheme to achieve very low error rates while avoiding error floor. While the idea of concatenated polar codes was first introduced in [25], the problem of designing practical concatenated schemes using polar codes is yet to be studied. In [25], the authors study the classical idea of code concatenation using short polar codes as inner codes and a high-rate Reed-Solomon (RS) code as the outer code. It is shown that such a concatenation scheme with a careful choice of parameters boosts the rate of decay of error probability to almost exponential in the block-length with essentially no loss in computational complexity. While [25] mainly considers the asymptotic case, we are interested in improving the performance in practical finite lengths.

In this paper, we study the combination of polar codes and LDPC codes, suggesting a polar code as the outer code and a LDPC code as the inner code. LDPC codes can be decoded in linear time using BP, while they can get very close to the capacity. However, LDPC codes with good waterfall characteristics are known to mostly suffer from the error floor problem. Here, polar codes come to play their role making the combination to show a good error floor performance. In order to investigate the performance of this scheme in a real-world application, we compare our proposed scheme against some of the conventional schemes used for OTNs. These schemes include a capacity-approaching LDPC code, the ITU-T Recommendation G.709 for OTNs, and some of the "super codes" of ITU-T G.975.1 for DWDM (Dense Wavelength Division Multiplexing) submarine cable systems. we will show that polar-LDPC combination actually outperforms these schemes as it closes the gap to capacity without showing error floor. Our results suggest that polar codes have a great potential to be used in combination with other codes in real-world communication systems.

The rest of the paper is organized as follows. We first explain the notations and provide a short background on the belief propagation. Section III gives an analysis on the minimum stopping set of polar codes. We provide a girth analysis of polar codes in Section IV where we also present simulation results for error floor performance. We propose concatenated polar codes to be used in a real-world application in Section V. Finally, Section VI concludes the paper.

II. Preliminaries

In this section, we explain the notations and some preliminary concepts we will use in our analysis. Let $F = \left[\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right]$ be the kernel used for construction of polar codes. Apply the transform $F^{\otimes n}$ (where $\otimes n$ denotes the nth Kronecker power) to a block of $N = 2^n$ bits and transmit the output through independent copies of a symmetric *binary discrete memoryless channel (B-DMC)*, call it W. As n grows large, the channels seen by individual bits (suitably defined in [26]) start polarizing to either a noiseless channel or a pure-noise channel, where the fraction of channels becoming noiseless is close to the capacity $I(W)$. Polar codes use the noiseless channels for transmitting information while fixing the symbols transmitted through the noisy ones to a value known both to the sender as well as the receiver. Accordingly, part of the block that carries information includes "information bits" while the rest of the block includes "frozen bits". Since we only deal with symmetric channels in this paper, we assume without loss of generality that the fixed positions are set to 0. The code is defined through its generator matrix as follows. Compute the Kronecker product $F^{\otimes n}$. This gives a $2^n \times 2^n$ matrix. The generator matrix of polar codes is a sub-matrix of $F^{\otimes n}$ in which only a subset of rows of $F^{\otimes n}$ are present. These rows are in fact the rows of $F^{\otimes n}$ corresponding to information bits. In the following, let $\bar{x} = (x_1, ..., x_N)$ and $\bar{y} = (y_1, ..., y_N)$ denote, respectively, the vectors of code-bits and channel output bits.

A Successive Cancelation (SC) decoding scheme is employed in [26] to prove the capacity-achieving property of polar codes. However, [18] and [19] later proposed using belief propagation decoding to obtain better BER performance while keeping the decoding complexity at $O(N \log N)$. Belief propagation can be run on the factor graph representation of the code [18]. Such a representation is easily obtained by adding check nodes to the encoding graph of polar codes, as it is shown in Fig. 1 for a code of length 8. We refer to this graph as the code's *factor graph*. Note that the factor graph is formed of columns of variable nodes and check nodes. There are, respectively, $n+1$ and n columns of variable and check nodes in the graph. We denote the variable nodes in jth column by $v(1,j), v(2,j), ..., v(N,j)$ for $j = 1, ..., n+1$. This is also shown in Fig. 1. Similarly, check nodes are labeled as $c(1,j), c(2,j), ..., c(N,j)$ for $j = 1, ..., n$. The rightmost column in the graph includes code-bits, while the leftmost column includes frozen and information bits. As it will become clear, our analysis does not depend on any specific choice of the frozen and information bits. Therefore, we treat all the nodes in the left-most column as variable nodes. Among $v(i,1), i = 1, ..., N$, some are associated to the information bits. We denote the index set of information bits by $\mathcal{A}$ where $\mathcal{A} \subseteq \{1, 2, ..., N\}$. Also, the row in $F^{\otimes n}$ associated with an information bit $i \in \mathcal{A}$ will be denoted by $\mathbf{r}_i = [r_{i,1}\ r_{i,2}\ ...\ r_{i,N}]$. Note that this is the ith row of $F^{\otimes n}$. We denote by $wt(\mathbf{r}_i)$ the Hamming weight of $\mathbf{r}_i$.

BP runs on the factor graph in a column-by-column fashion. That is, BP runs on each column of the adjacent variable and check nodes. The parameters are then passed to the next column. Each column, as it can be seen in Fig. 1, is formed of some Z-shaped subgraphs. In our proofs, we sometimes simply call a Z-shaped part a "Z". The schedule with which BP runs is very important for channels other than BEC. Here, we use the same scheduling used in [19], i.e. we update the LLRs for Z parts from bottom to top for each column, starting from the rightmost one. After arriving at the leftmost column, we reverse the course and update the Zs from top to bottom for each column, moving toward the rightmost one. This makes one round of iteration, and will repeat at each round. While we tried other schedules as well, this one led to a better overall performance.

We denote the factor graph of a code of length $N = 2^n$ by T_n. A key observation is the symmetric structure of this graph due to the recursive way of finding the generator matrix: T_{n+1} includes two factor graphs T_n as its upper and lower

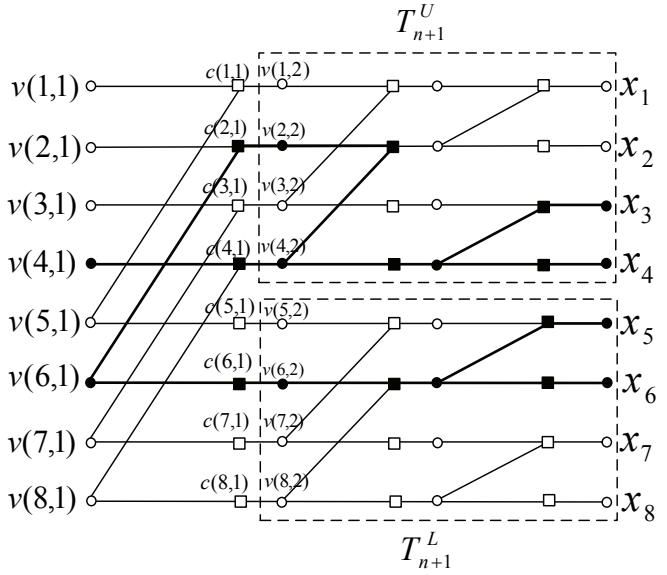


Fig. 1. Normal realization of the encoding graph for $N = 8$. An example of a GSS is shown with black variable and check nodes.

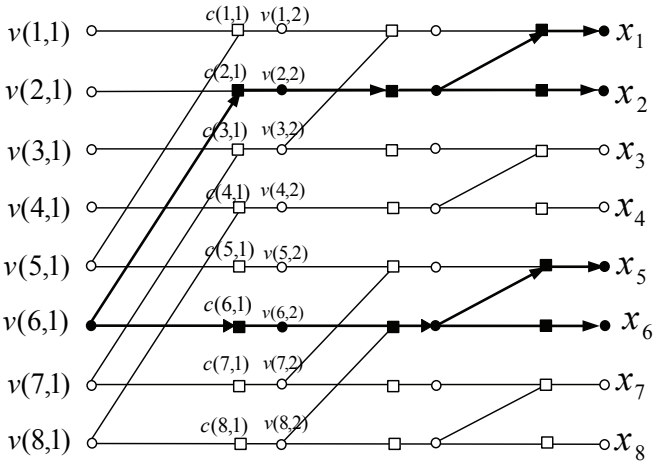


Fig. 2. The stopping tree for $v(6, 1)$ is shown with black variable and check nodes.

halves, connected together via $v(1,1), v(2,1), ..., v(N,1)$ and $c(1,1), c(2,1), ..., c(N,1)$. We denote these two subgraphs by T^U_{n+1} and T^L_{n+1}, as it is shown in Fig. 1. This observation will be later used in our analysis.

In this paper, we are particularly interested in the analysis of *stopping sets* in the factor graph of polar codes. A stopping set is a non-empty set of variable nodes such that every neighboring check node of the set is connected to at least two variable nodes in the set. Fig. 1 shows an example of the stopping set in the polar codes' graph, where we have also included the corresponding set of check nodes. A stopping set with minimum number of variable nodes is called a *minimum stopping set*.

A. Stopping Trees

An important category of stopping sets in the factor graph of polar codes are *stopping trees*. A stopping tree is a stopping set that contains one and only one information bit. It can be easily seen that this sub-graph is indeed a tree, therefore justifying its name. We say that the stopping tree is rooted at its (single) information bit (on the left side of the graph), with leaves at code-bits (on the right side of the graph). An example of such a stopping set is shown in Fig. 2 with black variable nodes. We also included the corresponding set of check nodes in order to visualize the structure of the tree. A stopping tree like the one shown in Fig. 2 can be immediately realized for any information bit. As we will later see (in Fact 2 below), this would in fact be the unique stopping tree for each information bit. We denote the stopping tree rooted at $v(i,1)$ by $ST(i)$. Among all the stopping trees, the one with minimum number of variable nodes is called a *minimum stopping tree*. We refer to the set of leaf nodes of a stopping tree as the *leaf set* of the tree. The size of the leaf set for $ST(i)$ is denoted by $f(i)$. We refer to a stopping tree with minimum leaf set as a *Minimum-Leaf Stopping Tree (MLST)*. Note that a minimum stopping tree does not necessarily have the minimum $f(i)$ among all the stopping trees.

B. Graph Stopping Sets vs. Variable-Node Stopping Sets

By looking at the factor graph of polar codes, one can observe that the middle variable nodes, i.e. $v(i,j)$ for $j = 2, ..., n$ and $i = 1, ..., N$, are always treated as erasures by the BP decoder. This is also true about information bits. Frozen bits, on the other hand, are known to the decoder. As a result, the only real "variable" nodes are the code-bits, i.e. $v(1, n+1), ..., v(N, n+1)$. These are in effect the variable nodes that if erased may cause a decoding failure. Here, we refer to a stopping set on the graph as a *Graph Stopping Set (GSS)*, while we refer to the set of code-bits on such a GSS as a *Variable-Node Stopping Set (VSS)*. In Fig. 1, the set $\{x_3, x_4, x_5, x_6\}$ is the VSS for the depicted GSS. As we will see later, every GSS must include some information bits and some code-bits. Thus, VSS is nonempty for each GSS. Accordingly, we define a *minimum VSS* (*MVSS*) as a VSS with minimum number of code-bits among all the VSSs. That is, a minimum VSS is the set of code-bits on a GSS with minimum number of code-bits among all GSSs. Note that a minimum VSS is not necessarily on a minimum GSS. We refer to the size of a minimum VSS as *stopping distance* of the code.

Now, for any given index set $J \subseteq \mathcal{A}$, there always exists an information bit $j \in J$ whose corresponding stopping tree has the smallest leaf set among all the elements in J. We call such an information bit a *minimum information bit* for J, denoted by $MIB(J)$. Note that there may exist more than one MIB in J. In general, any given index set $J \subseteq \mathcal{A}$ can be associated to several GSSs in the factor graph. We denote by $GSS(J)$ the set of all the GSSs that include J and only J as information bits. Each member of $GSS(J)$ includes a set of code-bits. The set of code-bits in each of these GSSs is a VSS for J. We refer to the set of these VSSs as *variable-node stopping sets* (VSSs) of J, denoted by $VSS(J)$. Among the sets in $VSS(J)$, we refer to the one with minimum cardinality as a *minimum VSS for* J, denoted by $MVSS(J)$. Let us also mention that all the proofs for the facts, lemmas, and theorems have been moved to the Appendix at the end of the paper.

III. Stopping Set Analysis of Polar Codes

In this section, we provide a stopping set analysis for polar codes. For the BEC, it is proved [20] that the set of erasures which remain when the decoder stops is equal to the unique maximal stopping set within the erased bits. In general, an analysis of the structure and size of the stopping sets can reveal important information about the error correction capability of the code. A minimum stopping set is more likely to be erased than larger stopping sets. Thus, minimum stopping sets play an important role in the decoding failure. In code design, codes with large minimum stopping sets are generally desired. We consider the problem of finding the minimum stopping set for a given polar code of length N. The results of this analysis may also help finding the optimal rule of choosing information bits to achieve the best error correction performance under belief propagation decoding.

A. Minimum VSS in the Graph

It is important to realize that what prevents the BP decoder from recovering a subset J of information bits is the erasure of the code-bits in one of the sets in $VSS(J)$. Therefore, what will eventually show up in any error probability analysis is the set of VSSs and their size. Particularly, $MVSS(J)$ represents the smallest set of code-bits whose erasure causes a decoding failure of J. We will find the size of $MVSS(J)$ for any given J. Furthermore, we will find the size of minimum VSS for a given polar code.

We start our analysis by stating some of the facts about the structure of stopping sets in the factor graph of polar codes. The factor graph of polar codes has a simple recursive structure which points to some useful observations. Here we mention some of these observations.

Fact 1: Any GSS in the factor graph of a polar code includes variable nodes from all columns of the graph. In particular, any GSS includes at least one information bit and one code-bit. ■

This implies that any given GSS includes a nonempty VSS.

Fact 2: Each information bit has a unique stopping tree. ■

Fact 3: Any GSS in T_{n+1} is formed of a GSS in T^U_{n+1} and/or a GSS in T^L_{n+1}, and a number of variable nodes $v(i,1),\ i=1,...,N$. ■

This implies that any GSS in T_{n+1} induces a GSS in T^U_{n+1} and/or T^L_{n+1}. This can be also seen in Fig. 1. The stopping set shown in the figure induces a stopping set in each of T^U_{n+1} and T^L_{n+1}. Now, consider size of the leaf set for different stopping trees. Note that we have $f(1)=1$, $f(2)=2$, $f(3)=2$, $f(4)=4$, so on. In general, we can state the following facts about $f(\cdot)$.

Fact 4: For a polar code of length $N=2^n$, the function $f(\cdot)$ can be formulated as follows:

$$\begin{aligned} &f(2^l)=2^l \quad \text{for} \quad l=0,1,...,n,\\ &f(2^l+m)=2f(m) \quad \text{for} \quad 1\le m\le 2^l-1, \quad 1\le l\le n-1. \end{aligned} \tag{1}$$

Thus $f(\cdot)$ is not necessarily an increasing function. ■

Fact 5: For a given polar code of length N formed by the kernel F, and for any $i\in\mathcal{A}$, we have $f(i)=wt(\mathbf{r}_i)$. In other word, the size of the leaf set for any stopping tree is in fact equal to the weight of the corresponding row in the generator matrix. Particularly, the leaf set of the stopping tree for any input bit represents the locations of 1's in the corresponding row of the matrix $F^{\otimes n}$. ■

Now, let us consider variable-node stopping sets for $J\subseteq\mathcal{A}$. The following theorem is proved for $MVSS(J)$ in the Appendix. The proof uses facts 1, 3, and 4.

Theorem 1: Given any set $J\subseteq\mathcal{A}$ of information bits in a polar code of length $N=2^n$, we have $|MVSS(J)|\ge \min_{j\in J} f(j)$. ■

Theorem 1 sets a lower bound on the size of the $MVSS$ for a subset J of information bits. It also implies that the size of the minimum VSS for a polar code is at least equal to $\min_{i\in\mathcal{A}} f(i)$. However, we already know that the leaf set of the stopping tree for any node $i\in\mathcal{A}$ is a VSS of size $f(i)$. This leads us to the following corollary.

Corollary 1: For a polar code with information bit index $\mathcal{A}$, the size of a minimum variable-node stopping set is equal to $\min_{i\in\mathcal{A}} f(i)$, i.e. the size of the leaf set for the minimum-leaf stopping tree. ■

Corollary 1 implies that in order to find the size of the minimum VSS, we need to find the information bit with minimum leaf stopping tree among all the information bits.

B. Size Distribution of Stopping Trees and their Leaf Sets

We provide a method for finding the size distribution of stopping trees and their leaf sets. First, note that the recursive construction of the factor graph dictates a relationship between the size of stopping trees in T_{n+1} and T_n.

Fact 6: Let $\mathbf{A}_n$ and $\mathbf{B}_n$ be two vectors of length 2^n showing, respectively, the size of stopping trees and their leaf sets for all input bits in T_n. That is, $\mathbf{A}_n = [|ST(1)|\ \ |ST(2)|\ \ ...\ \ |ST(2^n)|]$ and $\mathbf{B}_n = [f(1)\ f(2)\ ...\ f(2^n)]$. We then have

$$\begin{aligned} \mathbf{A}_{n+1} &= [\mathbf{A}_n\ \ 2\mathbf{A}_n] + \mathbf{1}_{n+1}\\ \mathbf{B}_{n+1} &= [\mathbf{B}_n\ \ 2\mathbf{B}_n], \end{aligned} \tag{2}$$

where $\mathbf{1}_{n+1}$ is the all-ones vector of length 2^{n+1}. ■

These two recursive equations can be solved with complexity $O(N)$ to find the desired size distributions for a code of length N. Note that Fact 4 can be also concluded from Fact 6. Furthermore, Fact 5 can be used to find the size of leaf set for a specific stopping tree within time $O(N)$.

C. Stopping Distance for Polar Codes

Fact 6 gives the stopping distance for a finite-length polar code, when the set of information bits is known. However, it is not always easy to choose the optimal information set, particularly with large code-lengths. In order to approach this problem, we first show that a slight modification in the set of information bits may actually result in a larger stopping distance without a significant impact on the BER performance.

Theorem 2: In the factor graph of a polar code of length N, the number of input bits $v(i,1)$ for which $f(i)<N^{\epsilon},\ 0<\epsilon<\frac{1}{2}$ is less than $N^{H(\epsilon)}$. ■

The above theorem implies that, for any $0<\epsilon<1/2$, we can always replace $N^{H(\epsilon)}$ information bits by some frozen

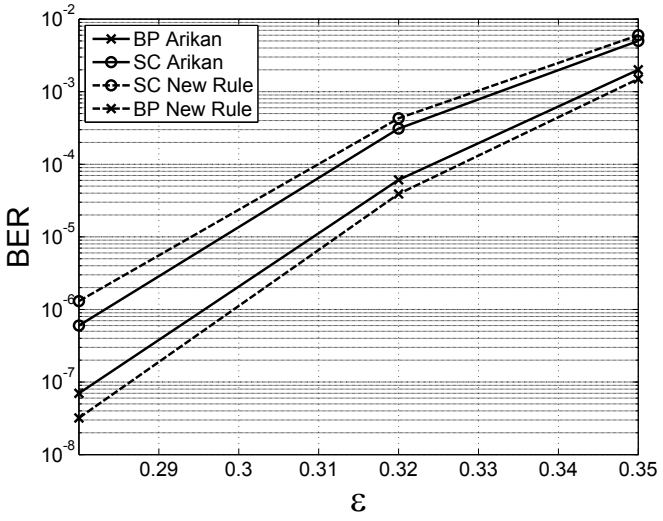


Fig. 3. BER comparison for different methods of choosing information bits under BP and SC decoding. Code-rate and code-length are 1/2 and 2^{13}, respectively.

bits for which the stopping tree has a leaf set larger than N^{ϵ}. It is easy to show that such a replacement does not effectively change the overall BER under BP, asymptotically. When $N \to \infty$ and $\epsilon < 1/2$, $N^{H(\epsilon)}$ will be vanishing with N. In a sparse factor graph, such as the one in polar codes, erroneous decoding of a small set of information bits affects only a few number (vanishing with N as $N \to \infty$) of other information bits. Therefore, given a finite number of iterations, BER will not change asymptotically. Accordingly, We can expect such a modification to have little impact on the BER performance in the finite regime, while resulting in a better error floor performance. Fig. 3 is used to demonstrate this case. The BER is depicted for Arikan's rule and its modified version introduced above (we call it *new rule*) applied to a code of length 2^{13} and rate 1/2. We replaced information bits with leaf sets smaller than 2^8, by frozen bits with minimum Bhattacharyya parameter who also had a leaf set larger than 2^8. As it can be seen, when SC decoding is used, the new rule performs slightly worse than the Arikan's rule. However, under BP decoding, it does slightly better than Arikan's rule. While the figure only shows the BER performance in the waterfall region, We conjecture that this rule results in a superior error floor performance of the new rule due to its larger stopping distance. It is also noteworthy that if we use the new rule to pick all the information bits, i.e. if we only pick input bits with largest leaf sets as information bits, then the resulting code will be a Reed-Muller code for which BP performance is worse than polar codes [18]. Therefore, we only considered a limited use of the new rule. This apparently helps to preserve some of the good characteristics of polar codes while increasing the stopping distance. We also like to mention two points regarding the stopping distance.

1) Asymptotic Case: Theorem 2 asserts that given any capacity-achieving polar code and any $\sigma > 0$, we can always construct another capacity-achieving code with a stopping distance $N^{1/2-\sigma}$, by replacing some information bits by some frozen bits with larger $f(.)$. The following theorem gives the stopping distance for polar codes in the asymptotic case. Note that this only holds asymptotically and the analysis is different for finite-length codes, as we explained above.

Theorem 3: The stopping distance for a polar code of length N grows as $\Omega(N^{1/2})$. ■

2) Minimum Distance vs. Stopping Distance: The following theorem states the relation between the stopping distance and minimum distance of polar codes.

Theorem 4: The stopping distance of a polar code defined on a normal realization graph such as the one in Fig. 1, is equal to the minimum distance of the code, d_{min}. ■

According to Theorem 4, the number of code-bits in the minimum VSS grows as fast as the minimum distance. It is noteworthy that for linear block codes, d_{min} (i.e. the minimum Hamming weight among all codewords) puts an upper bound on the stopping distance [27]–[29]. This is because if all the ones in the received vector are erased, then it is impossible for the decoder to find out if an all-zero codeword has been sent or another codeword. For a code, it is a desirable property to have a stopping distance equal to its minimum distance. Therefore, Theorem 4 can be interpreted as a positive result, particularly compared to the capacity-approaching LDPC codes for which both the stopping and minimum distances are fairly small in comparison to the block length [27]–[29].

IV. Error Floor Performance of Polar Codes

A large stopping distance is desirable in order to improve the error floor performance of a code over the BEC. After exploring the stopping sets of polar codes in the pervious section, here we focus on "girth" of polar codes as another important factor in error floor performance. Afterward, we examine the error floor performance of polar codes over the BEC and binary Gaussian channel via simulations.

A. Girth of Polar Codes

The *girth* of a graph is the length of shortest cycle contained in the graph. cycles in the Tanner graph prevent the sum-product (BP) algorithm from converging [30]. Furthermore, cycles, especially short ones, degrade the performance of the decoder, because they affect the independence of the extrinsic information exchanged in the iterative decoding. When decoded by belief propagation, the external information at every variable node remains uncorrelated until the iteration number reaches half the girth. Hence, we are often interested in constructing large girth codes that can achieve high performance under BP decoding [31]–[33]. As it can be seen in the factor graph shown in Fig. 4, there exist two types of cycles: first, the cycles including nodes only from one of the top or bottom part of the graph (shown by thick solid lines), and second, the cycles including nodes from both top and bottom parts of our symmetric graph (shown by thick dashed lines). The first type of cycles have the same shape in both upper and lower halves of the graph. The interesting fact about the cycles is that because the graph for a code of length 2^m is contained in the graph of a code of length 2^{m+1}, all the cycles of the shorter code are also present in the graph of the longer code. The shortest cycle appears in the graph of a length-4 polar code, as it is shown in Fig. 4. It is a cycle of size 12, including 6 variable nodes and 6 check nodes. The shortest

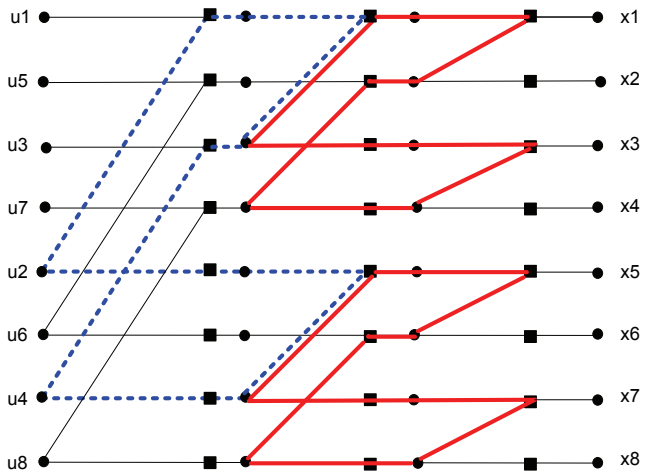


Fig. 4. Different types of cycles in the factor graph of polar codes for $N = 8$. Thick solid and dashed lines show the first and second types of cycles, respectively.

cycle of the second type appears first in the graph of a length-8 polar code, and have a size of 12 (dotted lines in Fig. 4). Thus, based on the above, the girth of a polar code is 12.

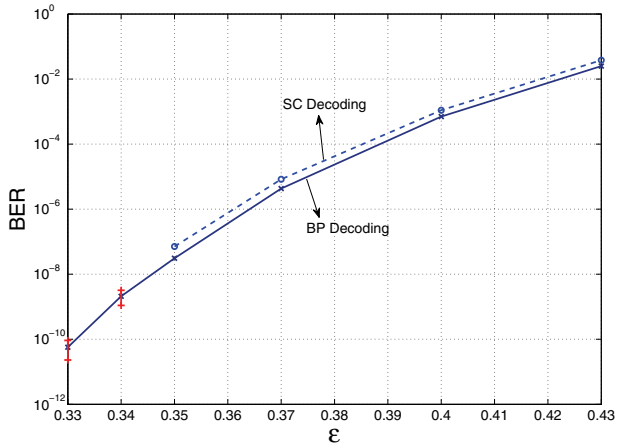
B. Simulation Results for Error Floor

We performed simulations to examine the effect of the relatively large stopping distance and girth of the polar codes' factor graph on the error correction performance of these codes. Fig. 5(a) shows the simulation results for a code of length 2^{15} and rate 1/2 over the BEC. As it can be seen, no sign of error floor is apparent. This is consistent with the relatively large stopping distance of polar codes. We indicated the 99% confidence interval for low BERs on the curve to show the precision of the simulation. Fig. 5(b) also shows the simulation results for a rate $\frac{1}{2}$ polar code of length 2^{13} over a binary-input Gaussian channel subjected to additive white Gaussian noise with zero mean and variance σ^2. The figure shows no sign of error floor down to the BERs of 10^{-9}.

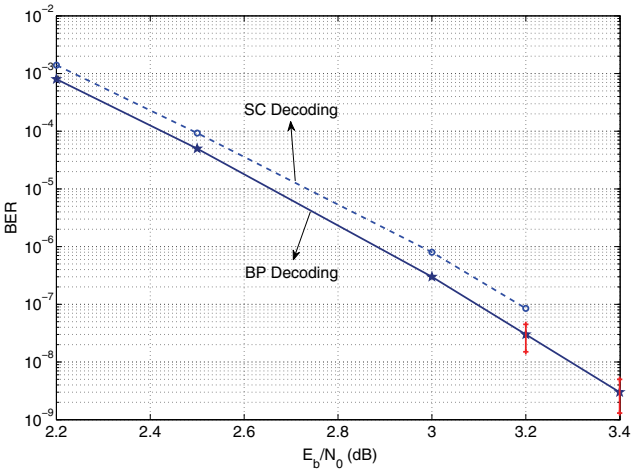
Regarding the error floor, we should mention here a prior work by Mori and Tanaka [34], which gives theoretical upper and lower bounds on *block error probability*, for SC decoding of polar codes over the BEC. According to these bounds, no error floor is expected for block error probability. Note also that for the BEC, BP decoding is strictly better than SC decoding [19]. Thus, if SC decoding shows no error floor problems, so does BP decoding. For large block lengths, however, a stopping distance of $\Omega(\sqrt{N})$ (as it was shown in Theorem 3) implies a good error floor performance for polar codes over the BEC.

V. A Potential Application for Polar Codes

Polar codes show a set of good characteristics that are needed in many real-world communication systems. Among these properties are good error floor performance, being capacity-achieving, and a low encoding and decoding complexity. In this section, we take advantage of these properties to design a polar code-based scheme as a solution to a practical problem. An *Optical Transport Network (OTN)* is a set of



(a) BER for BP and SC decoding over BEC. The code-length and code-rate are 2^{15} and $1/2$, respectively. The 99%



(b) BER for BP and SC decoding over Gaussian channel. The code-length and code-rate are 2^{13} and $1/2$, respectively.

Fig. 5. BER performance of polar codes over the binary erasure and Gaussian channels. The 99% confidence interval is shown for the two lowest BER's.

optical network elements connected by optical fiber links, able to transport client signals at data rates as high as 100 Gbit/s and beyond. These networks are standardized under ITU-T Recommendation G.709, and stand for an important part of the high data-rate transmission systems such as Gigabit Ethernet and the intercontinental communication network. A minimum BER of at least 10^{-13} is generally required in such systems [23], [24]. Because of very high-rate data transmission, OTNs need to employ a low complexity coding scheme to keep the delay in a low level. Furthermore, these systems generally use a long frame for data transmission, which allows using large code-lengths.

We propose concatenated polar-LDPC codes to be used in OTNs. Our proposed scheme is formed of a Polar code as the outer code, and a LDPC code as the inner code. Fig. 6 shows the block diagram of this scheme. We consider long powerful LDPC codes as the inner code with rates close to the channel capacity. LDPC codes with good waterfall characteristics are known to mostly suffer from the error floor problem. However, the polar code plays a dominant role in the error floor region of the LDPC code. Based on the analysis

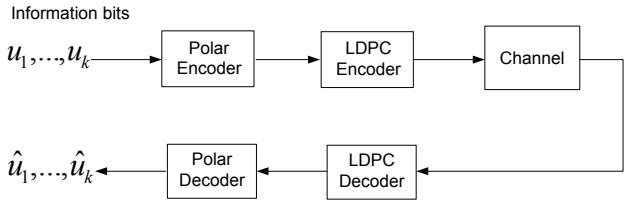


Fig. 6. Block diagram of the proposed concatenated system of polar and LDPC codes.

provided in previous sections, the combination of polar and LDPC codes is expected to form a powerful concatenated scheme with a BER performance close to the capacity for a broad range of the channel parameter. We consider a binary polar code concatenated with a binary LDPC code. This is different from the traditional concatenated schemes [35] in which a non-binary code is usually used as the outer code.

OTU4 is the standard designed to transport a 100 Gigabit Ethernet signal. The FEC (Forward Error Correction) in the standard OTU4 employs a block interleaving of 16 words of the (255, 239, 17) Reed-Solomon codes, resulting in an overall overhead of 7%. This scheme guarantees an error floor-free performance using a bounded distance decoder, and provides a coding gain of 5.8 dB at a BER of 10^{-13}. Since the approval of this standard (February 2001), several concatenated coding schemes have been proposed in the literature and some as patents, targeting to improve the performance of this standard. In most cases, these schemes propose a concatenation of two of Reed-Solomon, LDPC, and BCH codes [22]–[24], [36]. Here, for the first time, we consider polar-LDPC concatenation for the OTU4 setting.

A. Encoder

In order to satisfy the overhead of 7%, we adopt an effective code rate of 0.93. That is, if we denote the code-rates for the polar and LDPC codes by R_p and R_l respectively, then $R_{eff} = R_p \times R_l$ needs to be 0.93. The first problem is to find the optimal code-rate combination for the two codes to achieve the best BER performance. While this is an interesting analytical problem, it might be a difficult problem to solve. Therefore, we find the best rate combination for our application empirically. First, note that both R_p and R_l are greater than 0.93. We are also aware of the relatively poor error rate performance of finite-length polar codes compared to LDPC codes. Therefore, in order to minimize the rate loss, we choose R_l close to the R_{eff}. As a result, R_p would be close to 1. The values of R_l and R_p can be found empirically. Fig. 7 shows the BER performance of three different rate couples, as a sample of all the rate couples we simulated. Code-length for the polar code is fixed to $2^{15} = 32768$ for all the rate couples. Showing a rate couple by (R_p, R_l), these three rate couples are (0.989, 0.94), (0.979, 0.95), (0.969, 0.96). We picked (0.979, 0.95) for the rest of our simulations in this paper as it shows a better performance in the low-error-rate region. Fixing the code-length $2^{15} = 32768$ for the polar code and fixing the rates to (0.979, 0.95), the LDPC code-length would be 34493. We used the following optimal degree distribution pair which

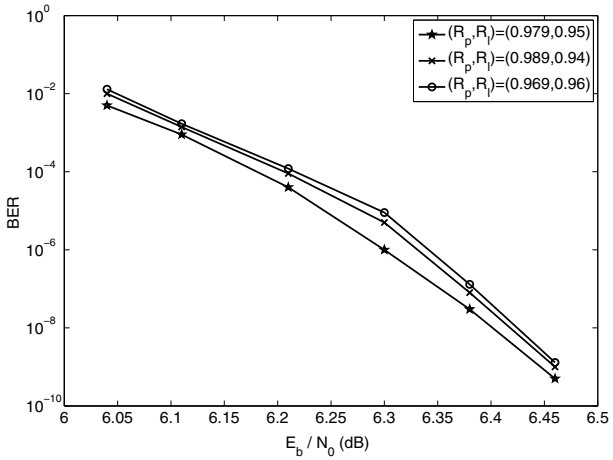


Fig. 7. BER performance comparison for different rate combinations in a polar-LDPC concatenated scheme.

has a threshold value of 0.47 for the binary AWGN channel under BP [37]:

$$\lambda(x) = 0.156935\ x + 0.138295\ x^2 + 0.325131\ x^3 + 0.168818\ x^{11} + 0.210821\ x^{12}, \quad (3)$$

$$\rho(x) = 0.039239\ x^{34} + 0.144375\ x^{35} + 0.302308\ x^{70} + 0.514078\ x^{71}. \quad (4)$$

An interesting question here is how to design the polar code in this concatenated scheme, while the channel seen by the polar code is not an AWGN channel anymore. It is well known, that when the iterative BP decoder fails, the residual erroneous bits after decoding are organized in graphical structures (e.g. stopping sets on BEC or trapping sets for other types of channels). In order to find the distribution of such patterns, one method is to prepare a histogram of these (post-decoding) error patterns. However, here we simply assume that the error patterns are distributed randomly (equally likely) at the output of the LDPC decoder, hence assuming the channel seen by the polar code as an AWGN channel with capacity 0.979. We then designed our polar code for this channel. The problem of designing optimal polar codes for this concatenated scheme remains as an interesting problem for further research.

B. Decoder

At the decoder side, we perform belief propagation decoding with soft-decision for both the polar and LDPC codes. Upon finishing its decoding, the LDPC decoder will pass its output vector of LLRs to the polar decoder. Polar decoder then treats this vector as the input for its belief propagation process.

C. Simulation Results

Fig. 8 depicts the BER performance for the concatenated scheme explained above, when using the LDPC code above. For the channel, we assumed a binary symmetric Gaussian channel as it is used by [22]–[24], [36]. Along with the concatenated scheme, we have shown the performance of the LDPC code when used alone with an effective rate of 0.93, which is equal to the effective rate of the concatenated scheme.

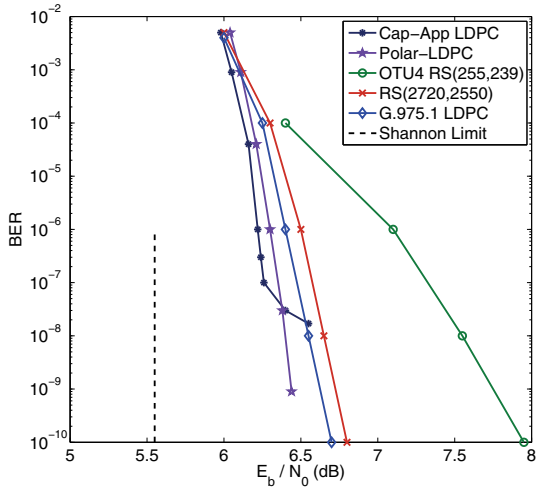


Fig. 8. BER performance for different concatenated schemes.

As it can be seen, the concatenated scheme follows the performance of LDPC code in the waterfall region closely. Since both polar and LDPC codes here are capacity-approaching (capacity-achieving in case of polar codes), this technique does not suffer from rate-loss theoretically. Therefore, by increasing the code-length we expect the curve for polar-LDPC scheme to close the gap to capacity. The curve also shows no sign of error floor down to BERs of 10^{-10}, as opposed to the curve for LDPC code which shows error floor at around 10^{-8}. What actually happens in a polar-LDPC concatenation is that the two codes are orchestrated to cover for each other's shortcomings: LDPC plays the dominant role in its waterfall region, while polar code is dominant in the error floor region of the LDPC code.

We should also mention that a soft BP decoder is used with a 9 bit quantization (512 values) of the LLRs. We are also limiting the LLR values to the range of (-20, 20). The maximum number of iterations used in our simulations is 60; however, we counted the average number of iterations (let us call it the ANI) for LDPC and polar-LDPC schemes in order to get some ideas about their decoding latency. At a BER of 10^{-6}, the ANI for the capacity-approaching LDPC code when used alone was 11.3. On the other hand, the ANI for the LDPC and polar codes used in the polar-LDPC scheme was 13.1 and 16.7, respectively. It should be noted that the BP-Polar iterations are heavier than the iterations for LDPC due to the $N \log N$ time of each iteration in BP-Polar in comparison to the linear time of each iteration in BP-LDPC. In our simulations for the lower points in the curves, we kept sending blocks until we encounter 100 erroneous blocks. For example, for polar-LDPC curve at 6.4 dB (the lowest BER), we ended up simulating over 300 million blocks. This particular point took us the longest amongst all the simulated points. The lowest point in the cap-app LDPC curve was obtained by simulating about 30 million blocks.

In order to see the significant potential of polar codes for concatenated schemes, we compared the BER performance of the polar-LDPC approach against some of the existing coding techniques for OTNs, including the G.709 standard explained earlier in the paper. We also included two "super FECs" proposed in ITU-T standard G.975.1 for high bit-rate DWDM (Dense Wavelength Division Multiplexing) submarine systems [38]. These schemes share some features, specifically the rate, block-length, and low decoding latency, with G.709, while achieving a much better performance. All the schemes use a code rate of 0.93. Furthermore, all of them are using codes of length around 2^{15}. We borrowed the BER curves of these schemes from [38].

As it is shown, an improvement of 1.3 dB at BER of 10^{-8} is achieved by polar-LDPC over the RS(255,239) of G.709 standard. Another scheme is an RS(2720,2550) with 12-bit symbols that has a block-length of 32640 bits. It has been shown to achieve a significant coding gain and to have superior burst correction capabilities [38]. As it is shown, polar-LDPC concatenation achieves an improvement of 0.25 dB over this scheme. Presented in the figure is also the performance of a systematic binary LDPC code of length 32640, with 30592 information-carrying bits [38]. This LDPC code is suitable for implementation in current chip technologies for 10G and 40G optical systems offering low latency and feasibility of low power consumption in case of 40G implementation showing a significantly higher coding gain than the standardized RS code in G.709. As it can be seen, polar-LDPC shows an edge of 0.15 dB over this LDPC scheme. The decoding complexity for LDPC and RS codes is $O(N)$ and $O(N^2)$, respectively, while the polar-LDPC scheme has a complexity of $O(N \log N)$ which is closer to the LDPC code.

VI. Conclusion

As a first step in a practical approach to polar codes, we studied the BER performance of finite-length polar codes under belief propagation decoding. We analyzed the structure of stopping sets in the factor graph of polar codes as one of the main contributors to the decoding failure and error floor over the BEC. The size of the minimum stopping set and the girth of the factor graph have been found for polar codes. We then investigated the error floor performance of polar codes through simulations where no sign of error floor was observed down to BERs of 10^{-10}. Motivated by good error floor performance, we proposed using polar codes in combination with other coding schemes. We particularly studied the polar-LDPC concatenation to be used in OTNs as a potential real-world application. Comparing the performance for our proposed scheme to some of the existing coding schemes for OTNs, we showed that polar-LDPC concatenation can achieve a significantly better performance.

Appendix

Proof of Fact 1: First, note that we only have degree 2 and 3 check nodes in the graph. In every Z-shaped part there are two check nodes, one at the top and one at the bottom. The top check node is always of degree 3 and the bottom one is always of degree 2. When a check node is a neighbor of a variable node or a set of variable nodes, we say that the check (variable) node is *adjacent to* that variable (check) node or the set of variable (check) nodes. We show that if a GSS is adjacent to either one of these check nodes in the ith column, then it must involve check nodes and variable nodes

from both $(i-1)$th and $(i+1)$th columns. Therefore, any GSS includes variable nodes from all columns of the graph, including information bits and code-bits.

We consider two cases. Since each neighboring check node of a GSS needs to be connected to at least two variable nodes in the set, if the bottom check node is adjacent to the GSS, then both of its neighboring variable nodes must be in the set. Since all the check nodes connected to a variable node in the GSS are also adjacent to the set, this means that some of the check nodes in the $(i-1)$th and $(i+1)$th columns are also adjacent to the set. In the second case, if the upper check node (of degree 3) is adjacent the GSS, then its neighbors in the GSS are either a variable node at its right and one at its left, or two variable nodes at its left, one at the top and one at the bottom of the Z. In the former case, the GSS clearly includes nodes from the $(i-1)$th and $(i+1)$th columns. In the latter case, the bottom variable node has the bottom check node as its neighbor in the GSS, leading to the same situation we discussed above. ■

Proof of Fact 2: Suppose an information bit i has two non-overlapping stopping trees, ST and ST'. Also, suppose ST has a form like the stopping tree shown in Fig. 2. That is only one variable node from each Z can participate in ST. Also, Note that a check (variable) node in the graph is adjacent to only one variable (check) node on the right (left). Thus, if a check node is adjacent to ST, it is adjacent to exactly one variable node on the left and one on the right.

Now assume that the difference between ST and ST' starts at the jth column. $j \neq 1$ Since, by definition, a stopping tree can include only one information bit; hence, $v(i,1)$ is the only variable node of column 1 participating in ST and ST'. Suppose there exists a variable node $v(k',j) \in ST', j \neq 1$, which is not part of ST. $v(k',j)$ is adjacent to $c(k',j-1)$ from left. However, $c(k',j-1)$ can not be adjacent to ST, otherwise we would have $v(k',j) \in ST$ because of what we mentioned above. But $c(k',j-1)$ must be adjacent to at least one variable node in ST' form the left since it needs to be adjacent to at least two variable nodes in ST' (definition of a stopping set). Therefore, $c(k',j-1)$ is adjacent to at least one variable node in ST' in the $(j-1)$th column, which is not part of ST. This is contradiction since we assumed ST and ST' start to differ at the jth column. ■

Proof of Fact 3: Fact 1 implies that any GSS in T_{n+1} includes at least one information bit. Consider such a GSS. According to Fact 1, this GSS includes a set of variable nodes in T^U_{n+1} and/or T^L_{n+1}. Let us denote these sets by S^U and S^L, respectively. Now, it is easy to see that the variable and check nodes in S^U and S^L, if non-empty, still satisfy the conditions of a GSS. This is because $v(1,1), v(2,1), ..., v(N,1)$ are connected to the rest of the graph only through $c(1,1), c(2,1), ..., c(N,1)$. Therefore, for any GSS in T_{n+1}, the induced non-empty subsets in T^U_{n+1} and T^L_{n+1} also form a GSS for these subgraphs. ■

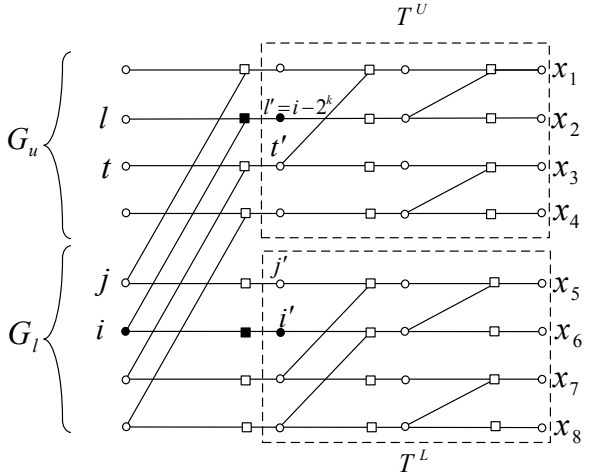
Proof of Fact 4: This fact can be concluded directly by looking at the recursive structure of the factor graph. ■

Proof of Fact 5: This is true because based on Arikan's paper, the encoding graph of polar codes is obtained from the matrix $F^{\otimes n}$. In fact, this graph is a representation of the recursive algebraic operations in this Kronecker product. ■

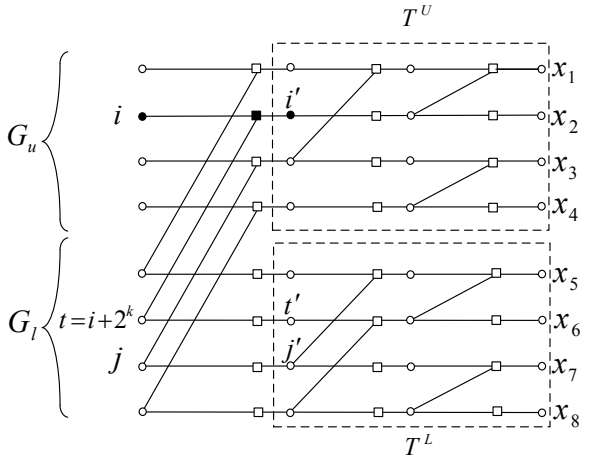
Proof of Theorem 1: We prove the theorem by induction on n where $N = 2^n$ is the code-length. For $n = 1$ $(N = 2)$, there are only two information bits, $v(1,1)$ and $v(2,1)$. It is trivial to check the correctness of the theorem in this case. Now suppose the hypothesis holds for a polar code of length 2^k. We prove that it also holds for a code of length 2^{k+1}. Consider a set J and let $MIB(J) = i$. In the case that there exist more than one MIB in J, without loss of generality, we pick the one with the largest index as the $MIB(J)$. That is, we pick the one which occupies the lowest place in the graph among the MIBs of J. Let VSS^* be a minimum VSS for J, and let GSS^* be the corresponding GSS for VSS^*. We also denote the upper and lower halves of the factor graph by G_U and G_L, as it is shown in Fig. 9(a). Note that G_U and G_L are identical in shape, and each of them includes half of the variable and check nodes in the factor graph. Without loss of generality, we assume that VSS^* includes variable nodes (code-bits in this case) from both G_U and G_L. We denote these two subsets of VSS^* by VSS^*_U and VSS^*_L, respectively. Also, GSS^* includes some variable nodes from the second column, i.e. from $v(1,2), ..., v(N,2)$. Let us denote the index set of these nodes by J'. For example, for the GSS shown in Fig. 1, J' is $\{2,4,6\}$. We also denote the subsets of J' in the upper and lower halves of the graph by J'_U and J'_L, respectively. Furthermore, We simply use T^U and T^L instead of T^U_{k+1} and T^L_{k+1}, since it is clear that we are dealing with the case $n = k+1$. Accordingly, we use $f_U(j')$ $(f_L(j'))$ to show the size of the leaf set for the stopping tree of $j' \in J'_U$ $(j' \in J'_L)$ in T^U (T^L).

For this setting, we need to show that for bit i to be erased, at least $f(i)$ code-bits must be erased, or equivalently, $|VSS^*| \geq f(i)$. We consider two cases: 1. $i \in G_L$, and 2. $i \in G_U$.

1) $i \in G_L$: This case is depicted in Fig. 9(a). First, note that $i - 2^k$ can not be in the VSS^*, because $f(i-2^k) = 1/2 f(i)$ and then i would not be a MIB. Now, for i to be erased, i' and $l' = i' - 2^k$ must be erased. Fact 3 asserts that J induces two stopping sets in T^U and T^L for J'_U and J'_L, respectively. We claim that i' and l' are MIB for J'_L and J'_U, respectively. If $i' \neq MIB(J'_L)$, then there exists a node j' such that $f_L(j') < f_L(i')$. Then, there exists $j \in \mathcal{A}$ such that $f(j) < f(i)$ which is in contradiction with the fact that i is a MIB.
If $l' \neq MIB(J'_U)$, then there exists t' such that $f_U(t') < f_U(l')$. This means that we have $t \in J$ and/or $t+2^k \in J$. However, we then have $f(t) < f(i)$ and $f(t+2^k) < f(i)$, which is again a contradiction with i being a MIB. Now, since $i' = MIB(J'_L)$ and $l' = MIB(J'_U)$, then the induction hypothesis implies that $|VSS^*_L| \geq f_L(i')$ and $|VSS^*_U| \geq f_U(l')$. Therefore, $|VSS^*| = |VSS^*_L| + |VSS^*_U| \geq f_L(i') + f_U(l') = f(i)$.
2) $i \in G_U$: This case is depicted in Fig. 9(b). If $J \cap G_L = \phi$, then we can prove that $i' = MIB(J'_U)$ along the same lines as the proof of case 1 above. Then the induction hypothesis implies that $VSS^* \geq f_U(i') = f(i)$,



(a) Case 1 in Theorem 1.



(b) Case 2 in Theorem 1.

Fig. 9. Figure is used to visualize different cases considered in the proof of Theorem 1.

and the proof would be complete for this case.
Now suppose that $J \cap G_L \neq \phi$. Consider any $j \in J \cap G_L$. We show that $f(j) > f(i+2^k)$. Let us denote $i+2^k$ by t. First note that $f(j) > f(i)$; otherwise if $f(j) = f(i)$, then according to our definition of MIB, we would pick j as the MIB since $j \in G_L$ and $i \in G_U$. Also note that $f(.)$ only takes value as powers of 2. Hence, we have $f(j) \geq 2f(i)$. Therefore, $f_L(j') = 1/2 f(j) \geq f(i) = f_L(t')$. As a result, $|VSS^*| \geq |VSS_L^*| \geq f_L(t') = f(i)$. ■

Proof of Fact 6: The fact becomes clear by looking at the recursive structure of the graph: T_{n+1} is formed of two copies of T_n, one at the top and one at the bottom, that are connected together. ■

Proof of Theorem 2: In the matrix $F^{\otimes n}$, there are $\binom{n}{i}$ rows with weight 2^i [19]. This means that in the factor graph of a polar code, there are $\binom{n}{i}$ stopping trees with a leaf set of size 2^i. Thus the corresponding tree of these input bits is at least of size 2^i. As a result, the number of input bits with less than $2^{\epsilon n} = N^{\epsilon}$ variable nodes in their tree is less than $\sum_{i=0}^{\epsilon n} \binom{n}{i}$, which is itself upper-bounded by $2^{H(\epsilon)n} = N^{H(\epsilon)}$ for $0 < \epsilon < \frac{1}{2}$. ■

Proof of Theorem 3: The block error probability for SC decoding over every B-DMC is proved to be $O(2^{-\sqrt{N}})$ [39]. Noting that the error correction performance of BP is at least as good as SC over the BEC [19], we conclude that block error probability for BP over the BEC decays as $O(2^{-\sqrt{N}})$ as well. Let us denote by $P_B(E)$ and $\Pr\{E_{MVSS}\}$, the block erasure probability and the probability of MVSS being erased. We then have

$$\begin{aligned}\Pr\{E_{MVSS}\} =& \epsilon^{|MVSS|} = (1/\epsilon)^{-|MVSS|} \leq P_B(E) \\ &= O(2^{-\sqrt{N}}) \Rightarrow |MVSS| = \Omega(\sqrt{N}), \quad (5)\end{aligned}$$

where ϵ is the channel erasure probability. ■

Proof of Theorem 4: First note that according to Fact 5, $f(i) = wt(\mathbf{r}_i)$ for any $i \in \mathcal{I}$. On the other hand, according to [17], [19], $d_{min} = \min_{i \in \mathcal{A}} wt(\mathbf{r}_i)$ for a polar code. Now using Corollary 1, $d_{min} = \min_{i \in \mathcal{A}} wt(\mathbf{r}_i) = \min_{i \in \mathcal{A}} f(i) = |MVSS|$. ■

REFERENCES

[1] R. Mori and T. Tanaka, "Non-binary polar codes using Reed-Solomon codes and algebraic geometry codes," in *Proc. 2010 IEEE Inf. Theory Workshop*.

[2] I. Tal and A. Vardy, "List decoding of polar codes," in *Proc. 2011 IEEE Int. Symp. Inf. Theory*.

[3] M. Seidl and J. B. Huber, "Improving successive cancellation decoding of polar codes by usage of inner block codes," in *Proc. 2010 IEEE Int. Symp. Turbo Codes Iterative Inf. Process.*

[4] A. Alamdar-Yazdi and F. R. Kschischang, "A simplified successive-cancellation decoder for polar codes," *IEEE Commun. Lett.*, vol. 15, pp. 1378–1380, Dec. 2011.

[5] N. Goela, S. Korada, and M. Gastpar, "On LP decoding of polar codes," in *Proc. 2010 IEEE Inf. Theory Workshop*.

[6] R. Pedarsani, H. Hassani, I. Tal, and E. Telatar, "On the construction of polar codes," in *Proc. 2011 IEEE Int. Symp. Inf. Theory*.

[7] I. Tal and A. Vardy, "How to construct polar codes," arXiv:1105.6164v1 [cs.IT], May 2011.

[8] A. Eslami and H. Pishro-Nik, "On bit error rate performance of polar codes in finite regime," in *Proc. 2010 Allerton Conf. Commun., Control, Computing*.

[9] A. Eslami and H. Pishro-Nik, "A practical approach to polar codes," in *Proc. 2011 IEEE Int. Symp. Inf. Theory*.

[10] G. Bonik, S. Goreinov, and N. Zamarashkin, "A variant of list plus CRC concatenated polar code," arXiv:1207.4661v1 [cs.IT], July 2012.

[11] B. Li, H. Shen, and D. Tse, "An adaptive successive cancellation list decoder for polar codes with cyclic redundancy check," arXiv:1208.3091 [cs.IT], Aug. 2012.

[12] K. Niu and K. Chen, "CRC-aided decoding of polar codes," *IEEE Commun. Lett.*, vol. 16, no. 10, pp. 1668–1671, Oct. 2012.

[13] S. Hassani and R. Urbanke, "On the scaling of polar codes—I: the behavior of polarized channels," in *Proc. 2010 IEEE Int. Symp. Inf. Theory*.

[14] S. Hassani, K. Alishahi, and R. Urbanke, "On the scaling of polar codes—I: the behavior of un-polarized channels," in *Proc. 2010 IEEE Int. Symp. Inf. Theory*.

[15] S. Korada, A. Montanari, E. Telatar, and R. Urbanke, "An emprical scaling law for polar codes," in *Proc. 2010 IEEE Int. Symp. Inf. Theory*.

[16] A. Goli, S. H. Hassani, and R. Urbanke, "Universal bounds on the scaling behavior of polar codes," in *Proc. 2012 IEEE Int. Symp. Inf. Theory*.

[17] S. H. Hassani, R. Mori, T. Tanaka, and R. Urbanke, "Rate-dependent analysis of the asymptotic behavior of channel polarization," Oct. 2011. Available: http://arxiv.org/abs/1110.0194v2

[18] E. Arikan, "A performance comparison of polar codes and Reed-Muller codes," *IEEE Commun. Lett.*, vol. 12, no. 6, pp. 447–449, 2008.

[19] N. Hussami, S. Korada, and R. Urbanke, "Performance of polar codes for channel and source coding," in *Proc. 2009 IEEE Int. Symp. Inf. Theory*.

[20] C. Di, D. Proietti, I. E. Telatar, T. Richardson, and R. Urbanke, "Finite-length analysis of low-density parity-check codes on the binary erasure channel," *IEEE Trans. Inf. Theory*, vol. 48, pp. 1570–1579, 2002.

[21] E. M. Kurtas, A. Kuznetsov, and I. Djurdjevic, "System perspectives for the application of structured LDPC codes to data storage devices," *IEEE Trans. Magn.*, vol. 42, no. 2, pp. 200–207, 2006.

[22] C. Wu and J. Cruz, "RS plus LDPC codes for perpendicular magnetic recording," *IEEE Trans. Magn.*, vol. 46, no. 16, pp. 1416–1419, 2010.

[23] X. Ningde, X. Wei, Z. Tong, E. F. Haratsch, and M. Jaekyun, "Concatenated low-density parity-check and BCH coding system for magnetic recording read channel with 4 kb sector format," *IEEE Trans. Magn.*, vol. 44, no. 12, pp. 4784–4789, 2008.

[24] T. Mizuochi, Y. Konishi, Y. Miyata, T. Inoue, K. Onohara, S. Kametani, T. Sugihara, K. Kubo, H. Yoshida, T. Kobayashi, and T. Ichikawa, "Experimental demonstration of concatenated LDPC and RS codes by FPGAs emulation," *IEEE Photonics Technol. Lett.*, vol. 21, no. 18, pp. 1302–1304, 2009.

[25] M. Bakshi, S. Jaggi, and M. Effros, "Concatenated polar codes," in *Proc. 2010 IEEE Int. Symp. Inf. Theory*.

[26] E. Arikan, "Channel polarization: a method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. Inf. Theory*, vol. 55, pp. 3051–3073, July 2009.

[27] C. Di, T. J. Richardson, and R. L. Urbanke, "Weight distribution of low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 52, no. 11, pp. 4839–4855, 2006.

[28] A. Orlitsky, K. Viswanathan, and J. Zhang, "Stopping set distribution of LDPC code ensembles," *IEEE Trans. Inf. Theory*, vol. 51, no. 3, pp. 929–953, 2005.

[29] A. Orlitsky, R. Urbanke, K. Viswanathan, and J. Zhang, "Stopping sets and the girth of Tanner graphs," in *Proc. 2002 IEEE Int. Symp. Inf. Theory*.

[30] F. R. Kschischang, B. J. Frey, and H. A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, pp. 498–519, Feb. 2001.

[31] M. Gholami and M. Esmaeili, "Maximum-girth cylinder-type block-circulant LDPC codes," *IEEE Trans. Commun.*, vol. 60, pp. 952–962, Apr. 2012.

[32] I. E. Bocharova, F. Hug, R. Johannesson, B. D. Kudryashov, and R. V. Satyukov, "Searching for voltage graph-based LDPC tailbiting codes with large girth," *IEEE Trans. Inf. Theory*, vol. 58, pp. 2265–2279, Apr. 2012.

[33] J. Huang, L. Liu, W. Zhou, and S. Zhou, "Large-girth nonbinary QC-LDPC codes of various lengths," *IEEE Trans. Commun.*, vol. 58, pp. 3436–3447, Dec. 2010.

[34] R. Mori and T. Tanaka, "Performance and construction of polar codes on symmetric binary-input memoryless channels," in *Proc. 2009 IEEE Int. Symp. Inf. Theory*, pp. 1496–1500.

[35] S. Lin and D. J. Costello, *Error Control Coding: Fundamentals and Applications*. Prentice-Hall, 1983.

[36] H. Griesser and J. P. Elbers, "Forward error correction coding." U.S. Patent, US 7,484,165 B2, Jan. 2009.

[37] Available: http://sigpromu.org/ldpc/

[38] ITU-T, "Forward error correction for high bit-rate dwdm submarine systems." International Telecommunication Union, Series G.975, Feb. 2004.

[39] E. Arikan and E. Telatar, "On the rate of channel polarization," in *Proc. 2009 IEEE Int. Symp. Inf. Theory*, 2009.

Ali Eslami received his B.S. and M.S. in electrical engineering from Sharif University of Technology, Tehran, Iran, in 2004 and 2006, respectively. He was a research assistant in the Information Systems and Security Lab (ISSL) at Sharif University from 2004 to 2007. He is currently pursuing a Ph.D. degree in electrical and computer engineering at the University of Massachusetts, Amherst. His research interests include analysis of wireless networks, network information theory, and error correcting codes.

Hossein Pishro-Nik is an Associate Professor of electrical and computer engineering at the University of Massachusetts, Amherst. He received a B.S. degree from Sharif University of Technology, and M.Sc. and Ph.D. degrees from the Georgia Institute of Technology, all in electrical and computer engineering. His research interests include mathematical analysis of communication systems, in particular, error control coding, wireless networks, and vehicular ad hoc networks. His awards include a NSF Faculty Early Career Development (CAREER) award, an Outstanding Junior Faculty Award from UMass, and an Outstanding Graduate Research Award from Georgia Tech.