

# On the Origin of Polar Coding

Erdal Arıkan

Department of Electrical and Electronics Engineering  
Bilkent University, Ankara, Turkey  
Email: arikan@ee.bilkent.edu.tr

**Abstract**—Polar coding was conceived originally as a technique for boosting the cutoff rate of sequential decoding, along the lines of earlier schemes of Pinsker and Massey. The key idea in boosting the cutoff rate is to take a vector channel (either given or artificially built), split it into multiple correlated subchannels, and employ a separate sequential decoder on each subchannel. Polar coding was originally designed to be a low-complexity recursive channel combining and splitting operation of this type, to be used as the inner code in a concatenated scheme with outer convolutional coding and sequential decoding. However, the polar inner code turned out to be so effective that no outer code was actually needed to achieve the original aim of boosting the cutoff rate to channel capacity. This paper explains the cutoff rate considerations that motivated the development of polar coding.

**Index Terms**—Channel polarization, polar codes, cutoff rate, sequential decoding.

## I. INTRODUCTION

The most fundamental parameter regarding a communication channel is unquestionably its capacity  $C$ , a concept introduced by Shannon [1] that marks the highest rate at which information can be transmitted reliably over the channel. Unfortunately, Shannon’s methods that established capacity as an achievable limit were non-constructive in nature, and the field of coding theory came into being with the agenda of turning Shannon’s promise into practical reality. Progress in coding theory was very rapid initially, with the first two decades producing some of the most innovative ideas in that field, but no truly practical capacity-achieving coding scheme emerged in this early period. A satisfactory solution of the coding problem had to await the invention of turbo codes [2] in 1990s. Today, there are several classes of capacity-achieving codes, among them a refined version of Gallager’s LDPC codes from 1960s [3]. (The fact that LDPC codes could approach capacity with feasible complexity was not realized until after their rediscovery in the mid-1990s.) A story of coding theory from its inception until the attainment of the major goal of the field can be found in the excellent survey article [4].

A recent addition to the class of capacity-achieving coding techniques is polar coding [5]. Polar coding was originally conceived as a method of boosting the channel cutoff rate  $R_0$ , a parameter that appears in two main roles in coding theory. First, in the context of random coding and maximum likelihood (ML) decoding,  $R_0$  governs the pairwise error probability  $2^{-NR_0}$ , which leads to the union bound

$$\bar{P}_e < 2^{-N(R_0-R)} \quad (1)$$

on the probability of ML decoding error  $\bar{P}_e$  for a randomly selected code with  $2^{NR}$  codewords. Second, in the context

of sequential decoding,  $R_0$  emerges as the “computational cutoff rate” beyond which sequential decoding—a decoding algorithm for convolutional codes—becomes computationally infeasible.

While  $R_0$  has a fundamental character in its role as an error exponent as in (1), its significance as the cutoff rate of sequential decoding is a fragile one. It has long been known that the cutoff rate of sequential decoding can be boosted by designing variants of sequential decoding that rely on various channel combining and splitting schemes to create correlated subchannels on which multiple sequential decoders are employed to achieve a sum cutoff rate that goes beyond the sum of the cutoff rates of the original memoryless channels used in the construction. An early scheme of this type is due to Pinsker [6], who used a concatenated coding scheme with an inner block code and an outer sequential decoder to get arbitrarily high reliability at constant complexity per bit at any rate  $R < C$ ; however, this scheme was not practical. Massey [7] subsequently described a scheme to boost the cutoff by splitting a nonbinary erasure channel into correlated binary erasure channels. We will discuss both of these schemes in detail, developing the insights that motivated the formulation of polar coding as a practical scheme for boosting the cutoff rate to its ultimate limit, the channel capacity  $C$ .

The account of polar coding given here is not intended to be the shortest or the most direct introduction to the subject. Rather, the goal is to give a historical account, highlighting the ideas that were essential in the course of developing polar codes, but have fallen aside as these codes took their final form. On a personal note, my acquaintance with sequential decoding began in 1980s during my doctoral work [8] which was about sequential decoding for multiple access channels. Early on in this period, I became aware of the “anomalous” behavior of the cutoff rate, as exemplified in the papers by Pinsker and Massey cited above, and the resolution of the paradox surrounding the boosting of the cutoff rate has been a central theme of my research over the years. Polar coding is the end result of such efforts.

The rest of this paper is organized as follows. We discuss the role of  $R_0$  in the context of ML decoding of block codes in Section II and its role in the context of sequential decoding of tree codes in Section III. In Section IV, we discuss the two methods by Pinsker and Massey mentioned above for boosting the cutoff rate of sequential decoding. In Section V, we examine the successive-cancellation architecture as an alternative method for boosting the cutoff rate, and in Section VI introduce polar coding as a special instance of that architecture. The paper concludes in Section VII with a

summary.

Throughout the paper, we use the notation  $W : \mathcal{X} \rightarrow \mathcal{Y}$  to denote a discrete memoryless channel  $W$  with input alphabet  $\mathcal{X}$ , output alphabet  $\mathcal{Y}$ , and channel transition probabilities  $W(y|x)$  (the conditional probability that  $y \in \mathcal{Y}$  is received given that  $x \in \mathcal{X}$  is transmitted). We use the notation  $a^N$  to denote a vector  $(a_1, \dots, a_N)$  and  $a_i^j$  to denote a subvector  $(a_i, \dots, a_j)$ . All logarithms in the paper are to the base two.

## II. THE PARAMETER $R_0$

The goal of this section is to discuss the significance of the parameter  $R_0$  in the context of block coding and ML decoding. Throughout this section, let  $W : \mathcal{X} \rightarrow \mathcal{Y}$  be a fixed but arbitrary memoryless channel. We will suppress in our notation the dependence of channel parameters on  $W$  with the exception of some definitions that are referenced later in the paper.

### A. Random-coding bound

Consider a communication system using block coding at the transmitter and ML decoding at the receiver. Specifically, suppose that the system employs a  $(N, R)$  block code, i.e., a code of length  $N$  and rate  $R$ , for transmitting one of  $M \triangleq \lfloor 2^{NR} \rfloor$  messages by  $N$  uses of  $W$ . We denote such a code as a list of codewords  $\mathcal{C} = \{x^N(1), \dots, x^N(M)\}$ , where each codeword is an element of  $\mathcal{X}^N$ . Each use of this system comprises selecting, at the transmitter, a message  $m \in \{1, \dots, M\}$  at random from the uniform distribution, encoding it into the codeword  $x^N(m)$ , and sending  $x^N(m)$  over  $W$ . At the receiver, a channel output  $y^N$  is observed with probability

$$W^N(y^N | x^N(m)) \triangleq \prod_{i=1}^N W(y_i | x_i(m))$$

and  $y^N$  is mapped to a decision  $\hat{m}$  by the ML rule

$$\hat{m}(y^N) \triangleq \arg\max_{m'} W^N(y^N | x^N(m')). \quad (2)$$

The performance of the system is measured by the probability of ML decoding error,

$$P_e(\mathcal{C}) \triangleq \sum_{m=1}^M \frac{1}{M} \sum_{y^N : \hat{m}(y^N) \neq m} W^N(y^N | x^N(m)). \quad (3)$$

Determining  $P_e(\mathcal{C})$  for a specific code  $\mathcal{C}$  is a well-known intractable problem. Shannon's *random-coding* method [1] circumvents this difficulty by considering an ensemble of codes, in which each individual code  $\mathcal{C} = \{x^N(1), \dots, x^N(M)\}$  is regarded as a sample with a probability assignment

$$\Pr(\mathcal{C}) = \prod_{m=1}^M \prod_{n=1}^N Q(x_n(m)), \quad (4)$$

where  $Q$  is a channel input distribution. We will refer to this ensemble as a " $(N, R, Q)$  ensemble". We will use the upper-case notation  $X^N(m)$  to denote the random codeword for message  $m$ , viewing  $x^N(m)$  as a realization of  $X^N(m)$ . The product-form nature of the probability assignment (4) signifies

that the array of  $MN$  symbols  $\{X_n(m); 1 \leq n \leq N, 1 \leq m \leq M\}$ , constituting the code, are sampled in i.i.d. manner.

The probability of error averaged over the  $(N, R, Q)$  ensemble is given by

$$\bar{P}_e(N, R, Q) \triangleq \sum_{\mathcal{C}} \Pr(\mathcal{C}) P_e(\mathcal{C}),$$

where the sum is over the set of all  $(N, R)$  codes. Classical results in information theory provide bounds of the form

$$\bar{P}_e(N, R, Q) \leq 2^{-NE_r(R, Q)}, \quad (5)$$

where the function  $E_r(R, Q)$  is a *random-coding exponent*, whose exact form depends on the specific bounding method used. For an early reference on such random-coding bounds, we refer to Fano [9, Chapter 9], who also gives a historical account of the subject. Here, we use a version of the random-coding bound due to Gallager [10], [11, Theorem 5.6.5]), in which the exponent is given by

$$E_r(R, Q) = \max_{0 \leq \rho \leq 1} [E_0(\rho, Q) - \rho R], \quad (6)$$

$$E_0(\rho, Q) \triangleq -\log \sum_{y \in \mathcal{Y}} \left[ \sum_{x \in \mathcal{X}} Q(x) W(y|x)^{1/(1+\rho)} \right]^{1+\rho}.$$

It is shown in [11, pp. 141-142] that, for any fixed  $Q$ ,  $E_r(R, Q) > 0$  for all  $R < C(Q)$ , where  $C(Q) = C(W, Q)$  is the channel capacity with input distribution  $Q$ , defined as

$$C(W, Q) \triangleq \sum_{x, y} Q(x) W(y|x) \log \left( \frac{W(y|x)}{\sum_{x'} Q(x') W(y|x')} \right).$$

This establishes that reliable communication is possible for all rates  $R < C(Q)$ , with the probability of ML decoding error approaching zero exponentially in  $N$ . Noting that the channel capacity is given by

$$C = C(W) \triangleq \max_Q C(W, Q), \quad (7)$$

the channel coding theorem follows as a corollary.

In a converse result [12], Gallager also shows that  $E_r(R, Q)$  is the best possible exponent of its type in the sense that

$$-\frac{1}{N} \log \bar{P}_e(N, R, Q) \rightarrow E_r(R, Q)$$

for any fixed  $0 \leq R \leq C(Q)$ . This converse shows that  $E_r(R, Q)$  is a channel parameter of fundamental significance in ML decoding of block codes.

For the best random-coding exponent of the type (6) for a given  $R$ , one may maximize  $E_r(R, Q)$  over  $Q$ , and obtain the optimal exponent as

$$E_r(R) \triangleq \max_Q E_r(R, Q). \quad (8)$$

We conclude this part with an example. Fig. 1 gives a sketch of  $E_r(R)$  for a binary symmetric channel (BSC), i.e., a channel  $W : \mathcal{X} \rightarrow \mathcal{Y}$  with  $\mathcal{X} = \mathcal{Y} = \{0, 1\}$  and  $W(1|0) = W(0|1) = p$  for some fixed  $0 \leq p \leq \frac{1}{2}$ . In this example, the  $Q$  that achieves the maximum in (8) happens to be the uniform distribution,  $Q(0) = Q(1) = \frac{1}{2}$  [11, p. 146], as one might expect due to symmetry.

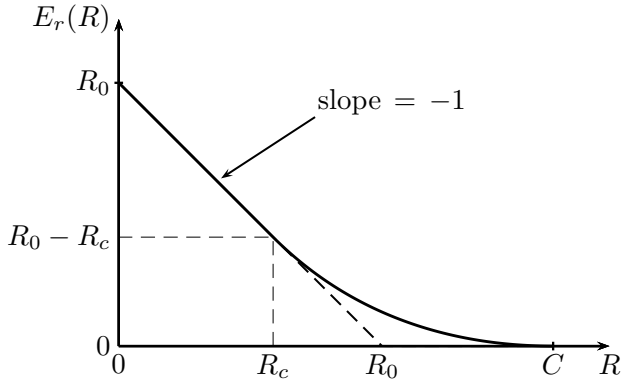


Fig. 1. Random-coding exponent  $E_r(R)$  as a function of  $R$  for a BSC.

The figure shows that  $E_r(R)$  is a convex function, starting at a maximum value  $R_0 \triangleq E_r(0)$  at  $R = 0$  and decreasing to 0 at  $R = C$ . The exponent has a straight-line portion for a range of rates  $0 \leq R \leq R_c$ , where the slope is  $-1$ . The parameters  $R_0$  and  $R_c$  are called, respectively, the *cutoff rate* and the *critical rate*. The union bound (1) coincides with the random-coding bound over the straight-line portion of the latter, becomes suboptimal in the range  $R_c < R \leq R_0$  (shown as a dashed line in the figure), and useless for  $R > R_0$ . These characteristics of  $E_r(R)$  and its relation to  $R_0$  are general properties that hold for all channels. In the rest of this section, we focus on  $R_0$  and discuss it from various other aspects to gain a better understanding of this ubiquitous parameter.

### B. The union bound

In general, the union bound is defined as

$$\bar{P}_e(N, R, Q) < 2^{-N[R_0(Q) - R]}, \quad (9)$$

where  $R_0(Q) = R_0(W, Q)$  is the channel cutoff rate with input distribution  $Q$ , defined as

$$R_0(W, Q) \triangleq -\log \sum_{y \in \mathcal{Y}} \left[ \sum_{x \in \mathcal{X}} Q(x) \sqrt{W(y|x)} \right]^2. \quad (10)$$

The union bound (9) may be obtained from the random-coding bound by setting  $\rho = 1$  in (6), instead of maximizing over  $0 \leq \rho \leq 1$  and noticing that  $R_0(Q)$  equals  $E_0(1, Q)$ . The union bound and the random-coding bound coincide over a range of rates  $0 \leq R \leq R_c(Q)$ , where  $R_c(Q)$  is the critical rate at input distribution  $Q$ . The tightest form of the union bound (9) is obtained by using an input distribution  $Q$  that maximizes  $R_0(Q)$ , in which case we obtain the usual form of the union bound as given by (1) with

$$R_0 = R_0(W) \triangleq \max_Q R_0(W, Q). \quad (11)$$

The role of  $R_0$  in connection with random-coding bounds can be understood by looking at the pairwise error probabilities under ML decoding. To discuss this, consider a specific code  $\mathcal{C} = \{x^N(1), \dots, x^N(M)\}$  and fix two distinct messages  $m \neq m'$ ,  $1 \leq m, m' \leq M$ . Let  $P_{m,m'}(\mathcal{C})$  denote the probability of pairwise error, namely, the probability that the erroneous

message  $m'$  appears to an ML decoder at least as likely as the correct message  $m$ ; more precisely,

$$P_{m,m'}(\mathcal{C}) \triangleq \sum_{y^N \in E_{m,m'}(\mathcal{C})} W^N(y^N | x^N(m)), \quad (12)$$

where  $E_{m,m'}(\mathcal{C})$  is a pairwise error event defined as

$$E_{m,m'}(\mathcal{C}) \triangleq \{y^N : W^N(y^N | x^N(m')) \geq W^N(y^N | x^N(m))\}.$$

Although  $P_{m,m'}(\mathcal{C})$  is difficult to compute for specific codes, its ensemble average,

$$\bar{P}_{m,m'}(N, Q) \triangleq \sum_{\mathcal{C}} \Pr(\mathcal{C}) P_{m,m'}(\mathcal{C}), \quad (13)$$

is bounded as

$$\bar{P}_{m,m'}(N, Q) \leq 2^{-NR_0(Q)}, \quad m \neq m'. \quad (14)$$

We provide a proof of inequality (14) in the Appendix to show that this well-known and basic result about the cutoff rate can be proved easily from first principles.

The union bound (9) now follows from (14) by noting that an ML error occurs only if some pairwise error occurs:

$$\begin{aligned} \bar{P}_e(N, R, Q) &\leq \sum_m \frac{1}{M} \sum_{m' \neq m} \bar{P}_{m,m'}(N, Q) \\ &\leq (2^{NR} - 1) 2^{-NR_0(Q)} < 2^{-N[R_0(Q) - R]}. \end{aligned}$$

This completes our discussion of the significance of  $R_0$  as an error exponent in random-coding. In summary,  $R_0$  governs the random-coding exponent at low rates and is fundamental in that sense.

For future reference, we note here that, when  $W$  is a binary-input channel with  $\mathcal{X} = \{0, 1\}$ , the cutoff rate expression (10) simplifies to

$$R_0(W, Q) = -\log(1 - q + qZ) \quad (15)$$

where  $q \triangleq 2Q(0)Q(1)$  and  $Z$  is the channel *Bhattacharyya parameter*, defined as

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

### C. Guesswork and $R_0$

Consider a coding system identical to the one described in the preceding subsection, except now suppose that the decoder is a *guessing decoder*. Given the channel output  $y^N$ , a guessing decoder is allowed to produce a sequence of guesses  $m_1(y^N), m_2(y^N), \dots$  at the correct message  $m$  until a helpful “genie” tells the decoder when to stop. More specifically, after the first guess  $m_1(y^N)$  is submitted, the genie tells the decoder if  $m_1(y^N) = m$ ; if so, decoding is completed; otherwise, the second guess  $m_2(y^N)$  is submitted; and, so on. The operation continues until the decoder produces the correct message. We assume that the decoder never repeats an earlier guess, so the task is completed after at most  $M$  guesses.

An appropriate “score” for such a guessing decoder is the *guesswork*, which we define as the number of *incorrect* guesses  $G_0(\mathcal{C})$  until completion of the decoding task. The guesswork

$G_0(\mathcal{C})$  is a random variable taking values in the range 0 to  $M-1$ . It should be clear that the optimal strategy for minimizing the average guesswork is to use the ML order: namely, to set the first guess  $m_1(y^N)$  equal to a most likely message given  $y^N$  (as in (2)), the second guess  $m_2(y^N)$  equal to a second most likely message given  $y^N$ , etc. We call a guessing decoder of this type an *ML-type* guessing decoder.

Let  $\mathbb{E}[G_0(\mathcal{C})]$  denote the average guesswork for an ML-type guessing decoder for a specific code  $\mathcal{C}$ . We observe that an incorrect message  $m' \neq m$  precedes the correct message  $m$  in the ML guessing order only if a channel output  $y^N$  is received such that  $W^N(y^N|x^N(m')) \geq W^N(y^N|x^N(m))$ ; thus,  $m'$  contributes to the guesswork only if a pairwise error event takes place between the correct message  $m$  and the incorrect message  $m'$ . Hence, we have

$$\mathbb{E}[G_0(\mathcal{C})] = \sum_m \frac{1}{M} \sum_{m' \neq m} P_{m,m'}(\mathcal{C}) \quad (16)$$

where  $P_{m,m'}(\mathcal{C})$  is the pairwise error probability under ML decoding as defined in (12). We observe that the right side of (16) is the same as the union bound on the probability of ML decoding error for code  $\mathcal{C}$ . As in the union bound, rather than trying to compute the guesswork for a specific code, we consider the ensemble average over all codes in a  $(N, R, Q)$  code ensemble, and obtain

$$\begin{aligned} \overline{G}_0(N, R, Q) &\triangleq \sum_{\mathcal{C}} \Pr(\mathcal{C}) \mathbb{E}[G_0(\mathcal{C})] \\ &= \sum_m \frac{1}{M} \sum_{m' \neq m} \overline{P}_{m,m'}(N, Q), \end{aligned}$$

which in turn simplifies by (14) to

$$\overline{G}_0(N, R, Q) \leq 2^{N[R-R_0(Q)]}.$$

The bound on the guesswork is minimized if we use an ensemble  $(N, R, Q^*)$  for which  $Q^*$  achieves the maximum of  $R_0(Q)$  over all  $Q$ ; in that case, the bound becomes

$$\overline{G}_0(N, R, Q^*) \leq 2^{N[R-R_0]}. \quad (17)$$

In [13], the following converse result was provided for any code  $\mathcal{C}$  of rate  $R$  and block length  $N$ :

$$\mathbb{E}[G_0(\mathcal{C})] \geq \max\{0, 2^{N(R-R_0-o(N))} - 1\}, \quad (18)$$

where  $o(N)$  is a quantity that goes to 0 as  $N$  becomes large. Viewed together, (17) and (18) state that  $R_0$  is a rate threshold that separates two very distinct regimes of operation for an ML-type guessing decoder on channel  $W$ : for  $R > R_0$ , the average guesswork is exponentially large in  $N$  regardless of how the code is chosen; for  $R < R_0$ , it is possible to keep the average guesswork close to 0 by an appropriate choice of the code. In this sense,  $R_0$  acts as a *computational cutoff rate*, beyond which guessing decoders become computationally infeasible.

Although a guessing decoder with a genie is an artificial construct, it provides a valid computational model for studying the computational complexity of the sequential decoding algorithm, as we will see in Sect. III. The interpretation of  $R_0$  as a computational cutoff rate in guessing will carry over directly to sequential decoding.

### III. SEQUENTIAL DECODING

The random-coding results show that a code picked at random is likely to be a very good code with an ML decoder error probability exponentially small in code block length. Unfortunately, randomly-chosen codes do not solve the coding problem, because such codes lack structure, which makes them hard to encode and decode. For a practically acceptable balance between performance and complexity, there are two broad classes of techniques. One is the algebraic-coding approach that eliminates random elements from code construction entirely; this approach has produced many codes with low-complexity encoding and decoding algorithms, but so far none that is capacity-achieving with a practical decoding algorithm. The second is the probabilistic-coding approach that retains a certain amount of randomness while imposing a significant degree of structure on the code so that low-complexity encoding and decoding are possible. A tree code with sequential decoding is an example of this second approach.

#### A. Tree codes

A tree code is a code in which the codewords conform to a tree structure. A convolutional code is a tree code in which the codewords are closed under vector addition. These codes were introduced by Elias [14] with the motivation to reduce the complexity of ML decoding by imposing a tree structure on block codes. In the discussion below, we will be considering tree codes with infinite length and infinite memory in order to avoid distracting details; although, in practice, one would use a finite-length finite-memory convolutional code.

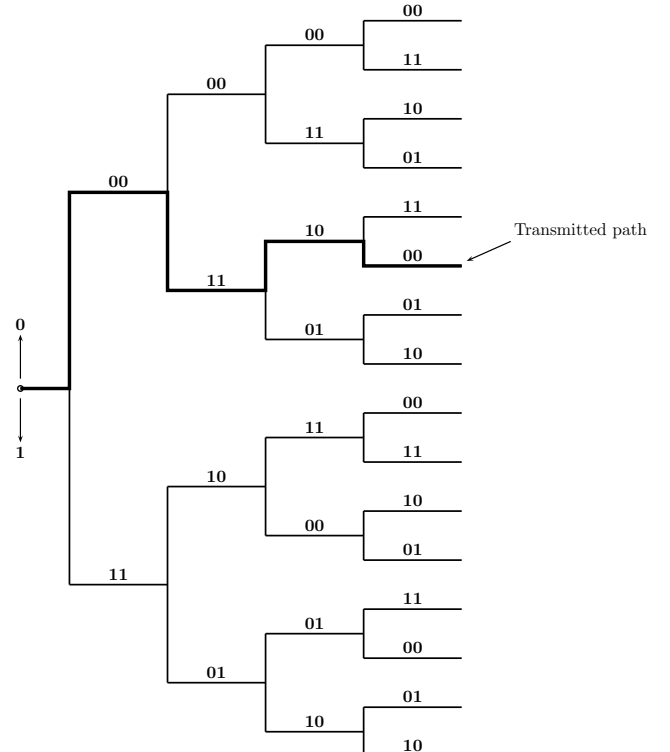


Fig. 2. A tree code.

The encoding operation for tree codes can be described with the aid of Fig. 2, which shows the first four levels of a tree code with rate  $R = 1/2$ . Initially, the encoder is situated at the root of the tree and the codeword string is empty. During each unit of time, one new data bit enters the encoder and causes it to move one level deeper into the tree, taking the upper branch if the input bit is 0, the lower one otherwise. As the encoder moves from one level to the next, it puts out the two-bit label on the traversed branch as the current segment of the codeword. An example of such encoding is shown in the figure, where in response to the input string 0101 the encoder produces 00111000.

The decoding task for a tree code may be regarded as a search for the correct (transmitted) path through the code tree, given a noisy observation of that path. Elias [14] gave a random-coding argument showing that tree codes are capacity-achieving. (In fact, he proved this result also for time-varying convolutional codes.) Thus, having a tree structure in a code comes with no penalty in terms of capacity, but makes it possible to implement ML decoding at reduced complexity thanks to various search heuristics that exploit this structure.

### B. Sequential decoding

Consider implementing ML decoding in an infinite tree code. Clearly, one cannot wait until the end of transmissions, decoding has to start with a partial (and noisy) observation of the transmitted path. Accordingly, it is reasonable to look for a decoder that has, at each time instant, a working hypothesis with respect to the transmitted path but is permitted to go back and change that hypothesis as new observations arrive from the channel. There is no final decision in this framework; all hypotheses are tentative.

An algorithm of this type, called *sequential decoding*, was introduced by Wozencraft [15], [16], and remained an important research area for more than a decade. A version of sequential decoding due to Fano [17] was used in the Pioneer 9 deep-space mission in the late 1960s [18]. Following this brief period of popularity, sequential decoding was eclipsed by other methods and never recovered. (For a perspective on the rise and fall of sequential decoding, we refer to [19].)

The main drawback of sequential decoding, which partly explains its decline, is the variability of computation. Sequential decoding is an ML algorithm, capable of producing error-free output given enough time, but this performance comes at the expense of using a backtracking search. The time lost in backtracking increases with the severity of noise in the channel and the rate of the code. From the very beginning, it was recognized [15], [20] that the computational complexity of sequential decoding is characterized by the existence of a computational cutoff rate, denoted  $R_{\text{cutoff}}$  (or  $R_{\text{comp}}$ ), that separates two radically different regimes of operation in terms of complexity: at rates  $R < R_{\text{cutoff}}$  the average number of decoding operations per bit remains bounded by a constant, while for  $R > R_{\text{cutoff}}$ , the decoding latency grows arbitrarily large. Later work on sequential decoding established that  $R_{\text{cutoff}}$  coincides with the channel parameter  $R_0$ . For a proof of the achievability part,  $R_{\text{cutoff}} \geq R_0$ , and bibliographic

references, we refer to [11, Theorem 6.9.2]; for the converse,  $R_{\text{cutoff}} \leq R_0$ , we refer to [21], [22].

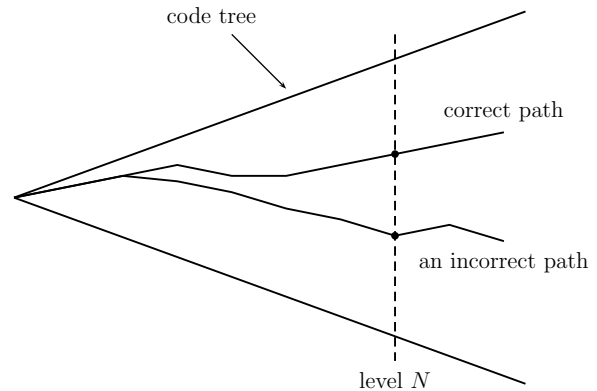


Fig. 3. Searching for the correct node at level  $N$ .

An argument that explains why “ $R_{\text{cutoff}} = R_0$ ” can be given by a simplified complexity model introduced by Jacobs and Berlekamp [21] that abstracts out the essential features of sequential decoding while leaving out irrelevant details. In this simplified model one fixes an arbitrary level  $N$  in the code tree, as in Fig. 3, and watches decoder actions only at this level. The decoder visits level  $N$  a number of times over the span of decoding, paying various numbers of visits to various nodes. A sequential decoder restricted to its operations at level  $N$  may be seen as a type of guessing decoder in the sense of Sect. II-C operating on the block code of length  $N$  obtained by truncating the tree code at level  $N$ . Unlike the guessing decoder for a block code, a sequential decoder does not need a genie to find out whether its current guess is correct; an incorrect turn by the sequential decoder is sooner or later detected with probability one with the aid of a *metric*, i.e., a likelihood measure that tends to decrease as soon as the decoder deviates from the correct path. To follow the guessing decoder analogy further, let  $G_{0,N}$  be the number of distinct nodes visited at level  $N$  by the sequential decoder before its first visit to the correct node at that level. In light of the results given earlier for the guessing decoder, it should not be surprising that  $\mathbb{E}[G_{0,N}]$  shows two types of behavior depending on the rate: for  $R > R_0$ ,  $\mathbb{E}[G_{0,N}]$  grows exponentially with  $N$ ; for  $R < R_0$ ,  $\mathbb{E}[G_{0,N}]$  remains bounded by a constant independent of  $N$ . Thus, it is natural that  $R_{\text{cutoff}} = R_0$ .

To summarize, this section has explained why  $R_0$  appears as a cutoff rate in sequential decoding by linking sequential decoding to guessing. While  $R_0$  has a firm meaning in its role as part of the random-coding exponent, it is a fragile parameter as the cutoff rate in sequential decoding. By devising variants of sequential decoding, it is possible to break the  $R_0$  barrier, as examples in the next section will demonstrate.

## IV. BOOSTING THE CUTOFF RATE IN SEQUENTIAL DECODING

In this section, we discuss two methods for boosting the cutoff rate in sequential decoding. The first method, due to Pinsker [6], was introduced in the context of a theoretical

analysis of the tradeoff between complexity and performance in coding. The second method, due to Massey [7], had more immediate practical goals and was introduced in the context of the design of a coding and modulation scheme for an optical channel. We present these schemes in reverse chronological order since Massey's scheme is simpler and contains the prototypical idea for boosting the cutoff rate.

#### A. Massey's scheme

A paper by Massey [7] revealed a truly interesting aspect of the cutoff rate by showing that it could be boosted by simply "splitting" a given channel. The simplest channel where Massey's idea can be employed is a quaternary erasure channel (QEC) with erasure probability  $\epsilon$ , as shown in Fig. 4(a). The capacity and cutoff rate of this channel are given by  $C_{\text{QEC}}(\epsilon) = 2(1 - \epsilon)$  and  $R_{0,\text{QEC}}(\epsilon) = \log \frac{4}{1+3\epsilon}$ .

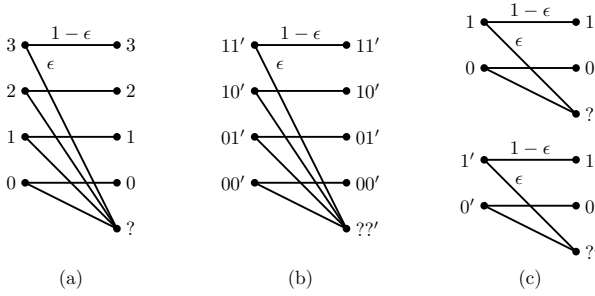


Fig. 4. Splitting a QEC into two fully-correlated BECs by input-relabeling.

Consider relabeling the inputs of the QEC with a pair of bits as in Fig. 4(b). This turns the QEC into a vector channel with input  $(b, b')$  and output  $(s, s')$  and transition probabilities

$$(s, s') = \begin{cases} (b, b'), & \text{with probability } 1 - \epsilon, \\ (?, ?'), & \text{with probability } \epsilon. \end{cases}$$

Following such relabeling, we can split the QEC into two binary erasure channels (BECs), as shown in Fig. 4(c). The resulting BECs are fully correlated in the sense that an erasure occurs in one if and only if an erasure occurs in the other.

One way to employ coding on the original QEC is to split it as above into two BECs and employ coding on each BEC independently, ignoring the correlation between them. In that case, the achievable sum capacity is given by  $2C_{\text{BEC}}(\epsilon) = 2(1 - \epsilon)$ , which is the same as the capacity of the original QEC. Even more surprisingly, the achievable sum cutoff rate after splitting is  $2R_{0,\text{BEC}}(\epsilon) = 2\log \frac{2}{1+\epsilon}$ , which is strictly larger than  $R_{0,\text{QEC}}(\epsilon)$  for any  $0 < \epsilon < 1$ . The capacity and cutoff rates for the two coding alternatives are sketched in Fig. 5, showing that substantial gains in the cutoff rate are obtained by splitting.

The above example demonstrates in very simple terms that just by splitting a composite channel into its constituent subchannels one may be able obtain a net gain in the cutoff rate without sacrificing capacity. Unfortunately, it is not clear how to generalize Massey's idea to other channels. For example, if the channel has a binary input alphabet, it cannot be split. Even if the original channel is amenable to splitting, ignoring

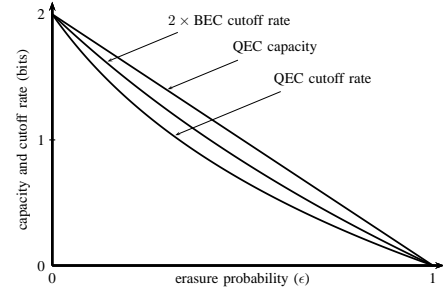


Fig. 5. Capacity and cutoff rates with and without splitting a QEC.

the correlations among the subchannels created by splitting may be costly in terms of capacity. So, Massey's scheme remains an interesting isolated instance of cutoff-rate boosting by channel splitting. Its main value lies in its simplicity and the suggestion that building correlated subchannels may be the key to achieving cutoff rate gains. In closing, we refer to [23] for an alternative discussion of Massey's scheme from the viewpoint of multi-access channels.

#### B. Pinsker's method

Pinsker was perhaps the first to draw attention to the flaky nature of the cutoff rate and suggest a general method to turn that into an advantage in terms of complexity of decoding. Pinsker's scheme, shown in Fig. 6, combines sequential decoding with Elias' product-coding method [24]. The main idea in Pinsker's scheme is to have an inner block code clean up the channels seen by a bank of outer sequential decoders, boosting the cutoff rate seen by each sequential decoder to near 1 bit. In turn, the sequential decoders boost the reliability to arbitrarily high levels at low complexity. Stated roughly, Pinsker showed that his scheme can operate arbitrarily close to capacity while providing arbitrarily low probability of error at constant average complexity per decoded bit. The details are as follows.

Following Pinsker's exposition, we will assume that the channel  $W$  in the system is a BSC with crossover probability  $0 \leq p \leq 1$ , in which case the capacity is given by

$$C(p) \triangleq 1 + p \log p + (1 - p) \log(1 - p).$$

The user data consists of  $K_2$  independent bit-streams, denoted  $d_1, d_2, \dots, d_{K_2}$ . Each stream is encoded by a separate convolutional encoder (CE), with all CEs operating at a common rate  $R_1$ . Each block of  $K_2$  bits coming out of the CEs is encoded by an inner block code, which operates at rate  $R_2 = K_2/N_2$  and is assumed to be a linear code. Thus, the overall transmission rate is  $R = R_1 R_2$ .

The codewords of the inner block code are sent over  $W$  by  $N_2$  uses of that channel as shown in the figure. The received sequence is first passed through an ML decoder for the inner block code, then each bit obtained at the output of the ML decoder is fed into a separate sequential decoder (SD), with the  $i$ th SD generating an estimate  $\hat{d}_i$  of  $d_i$ ,  $1 \leq i \leq K_2$ . The SDs operate in parallel and independently (without exchanging any information). An error is said to occur if  $\hat{d}_i \neq d_i$  for some  $1 \leq i \leq K_2$ .

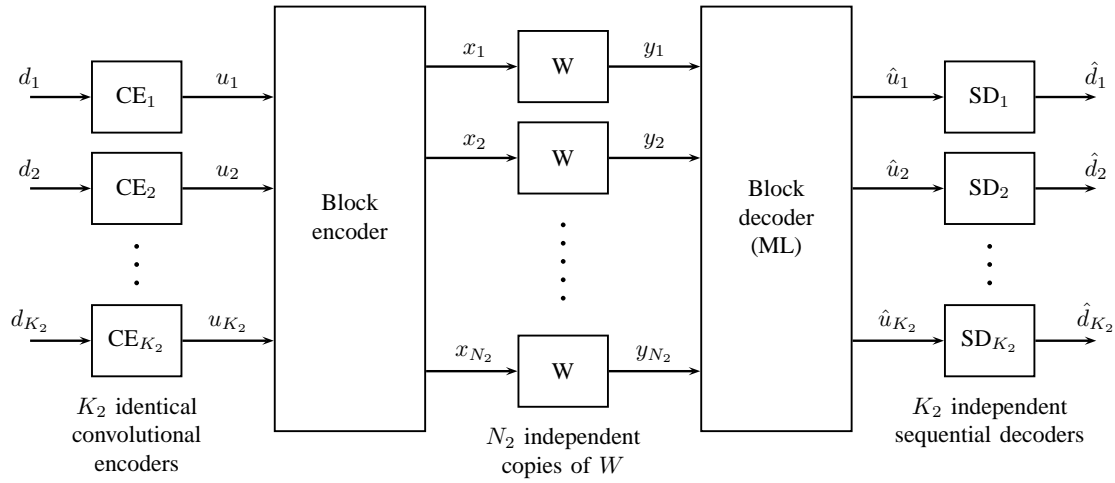


Fig. 6. Pinsker's scheme for boosting the cutoff rate.

The probability of ML decoding error for the inner code,  $p_e \triangleq P(\hat{u}^N \neq u^N)$ , is independent of the transmitted codeword since the code is *linear* and the channel is a BSC<sup>1</sup>. Each frame error in the inner code causes a burst of bit errors that spread across the  $K_2$  parallel bit-channels, but do not affect more than one bit in each channel thanks to interleaving of bits by the product code. Thus, each bit-channel is a memoryless BSC with a certain crossover probability,  $p_i$ , that equals the bit-error rate  $P(\hat{u}_i \neq u_i)$  on the  $i$ th coordinate of the inner block code. So, the the cutoff rate “seen” by the  $i$ th CE-SD pair is

$$R_0(p_i) \triangleq 1 - \log(1 + 2\sqrt{p_i(1 - p_i)}),$$

which is obtained from (15) with  $Q(0) = Q(1) = \frac{1}{2}$ . These cutoff rates are uniformly good in the sense that

$$R_0(p_i) \geq R_0(p_e), \quad 1 \leq i \leq K_2,$$

since  $0 \leq p_i \leq p_e$ , as in any block code.

It follows that the aggregate cutoff rate of the outer bit-channels is  $\geq K_2 R_0(p_e)$ , which corresponds to a normalized cutoff rate of better than  $R_2 R_0(p_e)$  bits per channel use. Now, consider fixing  $R_2$  just below capacity  $C(p)$  and selecting  $N_2$  large enough to ensure that  $p_e \approx 0$ . Then, the normalized cutoff rate satisfies  $R_2 R_0(p_e) \approx C(p)$ . This is the sense in which Pinsker's scheme boosts the cutoff rate to near capacity.

Although Pinsker's scheme shows that arbitrarily reliable communication at any rate below capacity is possible within constant complexity per bit, the “constant” entails the ML decoding complexity of an inner block code operating near capacity and providing near error-free communications. So, Pinsker's idea does not solve the coding problem in any practical sense. However, it points in the right direction, suggesting that channel combining and splitting are the key to boosting the cutoff rate.

<sup>1</sup>The important point here is that the channel be *symmetric* in the sense defined later in Sect.V. Pinsker's arguments hold for any binary-input channel that is symmetric.

### C. Discussion

In this part, we discuss the two examples above in a more abstract way in order to identify the essential features that are behind the boosting of the cutoff rate.

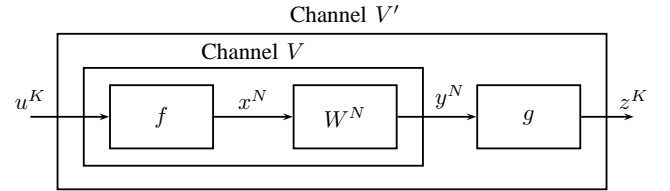


Fig. 7. Channels derived from  $W$  by pre- and post-processing operations.

Consider the system shown in Fig. 7 that presents a framework general enough to accommodate both Pinsker's method and Massey's method as special instances. The system consists of a *mapper*  $f$  and a *demapper*  $g$  that implement, respectively, the combining and splitting operations for a given memoryless channel  $W : \mathcal{X} \rightarrow \mathcal{Y}$ . The mapper and the demapper can be any functions of the form  $f : \mathcal{A}^K \rightarrow \mathcal{X}^N$  and  $g : \mathcal{Y}^N \rightarrow \mathcal{B}^K$ , where the alphabets  $\mathcal{A}$  and  $\mathcal{B}$ , as well as the dimensions  $K$  and  $N$  are design parameters.

The mapper acts as a pre-processor to create a derived channel  $V : \mathcal{A}^K \rightarrow \mathcal{Y}^N$  from vectors of length  $K$  over  $\mathcal{A}$  to vectors of length  $N$  over  $\mathcal{Y}$ . The demapper acts as a post-processor to create from  $V$  a second derived channel  $V' : \mathcal{A}^K \rightarrow \mathcal{B}^K$ . The well-known data-processing theorem of information theory [11, p. 50] states that

$$C(V') \leq C(V) \leq NC(W),$$

where  $C(V')$ ,  $C(V)$ , and  $C(W)$  denote the capacities of  $V'$ ,  $V$ , and  $W$ , respectively. There is also a data-processing theorem that applies to the cutoff rate, stating that

$$R_0(V') \leq R_0(V) \leq NR_0(W). \quad (19)$$

The data-processing result for the cutoff rate follows from a more general result given in [11, pp. 149-150] in the context

of “parallel channels.” In words, inequality (19) states that it is impossible to boost the cutoff rate of a channel  $W$  if one employs a *single* sequential decoder on the channels  $V$  or  $V'$  derived from  $W$  by any kind of pre-processing and post-processing operations.

On the other hand, there are cases where it is possible to split the derived channel  $V'$  into  $K$  memoryless channels  $V'_i : \mathcal{A} \rightarrow \mathcal{B}$ ,  $1 \leq i \leq K$ , so that the normalized cutoff rate after splitting shows a cutoff rate gain in the sense that

$$\frac{1}{N} \sum_{i=1}^K R_0(V'_i) > R_0(W). \quad (20)$$

Both Pinsker’s scheme and Massey’s scheme are examples where (20) is satisfied. In Pinsker’s scheme, the alphabets are  $\mathcal{A} = \mathcal{B} = \{0, 1\}$ ,  $f$  is an encoder for a binary block code of rate  $K_2/N_2$ ,  $g$  is an ML decoder for the block code, and the bit channel  $V'_i$  is the channel between  $u_i$  and  $z_i$ ,  $1 \leq i \leq K_2$ . In Massey’s scheme, with the QEC labeled as in Fig. 4-(a), the length parameters are  $K = 2$  and  $N = 1$ ,  $f$  is the identity map on  $\mathcal{A} = \{0, 1\}^2$ , and  $g$  is the identity map on  $\mathcal{B} = \{0, 1, ?\}^2$ .

As we conclude this section, a word of caution is necessary about the application of the above framework for cutoff rate gains. The coordinate channels  $\{V'_i\}$  created by the above scheme are in general not memoryless; they interfere with each other in complex ways depending on the specific  $f$  and  $g$  employed. For a channel  $V'_i$  with memory, the parameter  $R_0(V'_i)$  loses its operational meaning as the cutoff rate of sequential decoding. Pinsker avoids such technical difficulties in his construction by using a *linear* code and restricting the discussion to a *symmetric* channel. In designing systems that target cutoff rate gains as promised by (20), these points should not be overlooked.

## V. SUCCESSIVE-CANCELLATION ARCHITECTURE

In this section, we examine the successive-cancellation (SC) architecture, shown in Fig. 8, as a general framework for boosting the cutoff rate. The SC architecture is more flexible than Pinsker’s architecture in Fig. 7, and may be regarded as a generalization of it. This greater flexibility provides significant advantages in terms of building practical coding schemes, as we will see in the rest of the paper. As usual, we will assume that the channel in the system is a binary-input channel  $W : \mathcal{X} = \{0, 1\} \rightarrow \mathcal{Y}$ .

### A. Channel combining and splitting

As seen in Fig. 8, the transmitter in the SC architecture uses a 1-1 mapper  $f_N$  that combines  $N$  independent copies of  $W$  to synthesize a channel

$$W_N : u^N \in \{0, 1\}^N \rightarrow y^N \in \mathcal{Y}^N$$

with transition probabilities

$$W_N(y^N | u^N) = \prod_{i=1}^N W(y_i | x_i), \quad x^N = f_N(u^N).$$

The SC architecture has room for  $N$  CEs but these encoders do not have to operate at the same rate, which is one

difference between the SC architecture and Pinsker’s scheme. The intended mode of operation in the SC architecture is to set the rate of the  $i$ th encoder  $CE_i$  to a value commensurate with the capability of that channel.

The receiver side in the SC architecture consists of a soft-decision generator (SDG) and a chain of SDs that carry out SC decoding. To discuss the details of the receiver operation, let us index the blocks in the system by  $t$ . At time  $t$ , the  $t$ th code block  $x^N$ , denoted  $x^N(t)$ , is transmitted (over  $N$  copies of  $W$ ) and  $y^N(t)$  is delivered to the receiver. Assume that each round of transmission lasts for  $T$  time units, with  $\{x^N(1), \dots, x^N(T)\}$  being sent and  $\{y^N(1), \dots, y^N(T)\}$  received. Let us write  $\{x^N(t)\}$  to denote  $\{x^N(1), \dots, x^N(T)\}$  briefly. Let us use similar time-indexing for all other signals in the system, for example, let  $d_i(t)$  denote the data at the input of the  $i$ th encoder  $CE_i$  at time  $t$ .

Decoding in the SC architecture is done layer-by-layer, in  $N$  layers: first, the data sequence  $\{d_1(t) : 1 \leq t \leq T\}$  is decoded, then  $\{d_2(t)\}$  is decoded, and so on. To decode the first layer of data  $\{d_1(t)\}$ , the SDG computes the soft-decision variables  $\{\ell_1(t)\}$  as a function of  $\{y^N(t)\}$  and feeds them into the first sequential decoder  $SD_1$ . Given  $\{\ell_1(t)\}$ ,  $SD_1$  calculates two sequences: the estimates  $\{\hat{d}_1(t)\}$  of  $\{d_1(t)\}$ , which it sends out as its final decisions about  $\{d_1(t)\}$ ; and, the estimates  $\{\hat{u}_1(t)\}$  of  $\{u_1(t)\}$ , which it feeds back to the SDG. Having received  $\{\hat{u}_1(t)\}$ , the SDG proceeds to compute the soft-decision sequence  $\{\ell_2(t)\}$  and feeds them into  $SD_2$ , which, in turn, computes the estimates  $\{\hat{d}_2(t)\}$  and  $\{\hat{u}_2(t)\}$ , sends out  $\{\hat{d}_2(t)\}$ , and feeds  $\{\hat{u}_2(t)\}$  back into the SDG. In general, at the  $i$ th layer of SC decoding, the SDG computes the sequence  $\{\ell_i(t)\}$  and feeds it to  $SD_i$ , which in turn computes a data decision sequence  $\{\hat{d}_i(t)\}$ , which it sends out, and a second decision sequence  $\{\hat{u}_i(t)\}$ , which it feeds back to SDG. The operation is completed when the  $N$ th decoder  $SD_N$  computes and sends out the data decisions  $\{\hat{d}_N(t)\}$ .

### B. Capacity and cutoff rate analysis

For capacity and cutoff rate analysis of the SC architecture, we need to first specify a probabilistic model that covers all parts of the system. As usual, we will use upper-case notation to denote the random variables and vectors in the system. In particular, we will write  $X^N$  to denote the random vector at the output of the 1-1 mapper  $f_N$ ; likewise, we will write  $Y^N$  to denote the random vector at the input of the SDG  $g_N$ . We will assume that  $X^N$  is uniformly distributed,

$$p_{X^N}(x^N) = 1/2^N, \quad \text{for all } x^N \in \{0, 1\}^N.$$

Since  $X^N$  and  $Y^N$  are connected by  $N$  independent copies of  $W$ , we will have

$$p_{Y^N|X^N}(y^N | x^N) = \prod_{i=1}^N W(y_i | x_i).$$

The ensemble  $(X^N, Y^N)$ , thus specified, will serve as the core of the probabilistic analysis. Next, we expand the probabilistic model to cover other signals of interest in the system. We define  $U^N$  as the random vector that appears at the output of the CEs in Fig. 8. Since  $U^N$  is in 1-1 correspondence

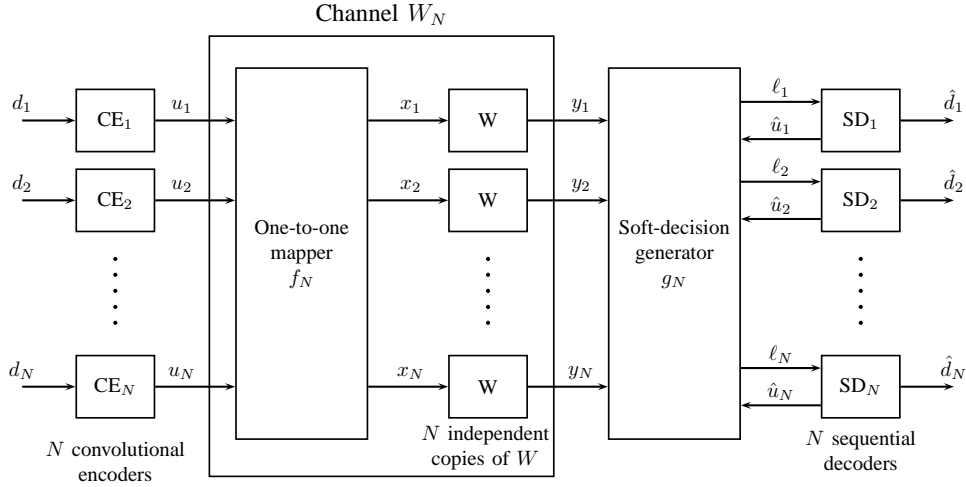


Fig. 8. Successive-cancellation architecture for boosting the cutoff rate.

with  $X^N$ , it is uniformly distributed. We define  $\hat{U}^N$  as the random vector that the SDs feed back to the SDG as the estimate of  $U^N$ . Ordinarily, any practical system has some non-zero probability that  $\hat{U}^N \neq U^N$ . However, modeling such decision errors and dealing with the consequent error propagation effects in the SC chain is a difficult problem. To avoid such difficulties, we will assume that the outer code in Fig. 8 is perfect, so that

$$\Pr(\hat{U}^N = U^N) = 1. \quad (21)$$

This assumption eliminates the complications arising from error-propagation; however, the capacity and cutoff rates calculated under this assumption will be optimistic estimates of what can be achieved by any real system. Still, the analysis will serve as a roadmap and provide benchmarks for practical system design. (In the case of polar codes, we will see that the estimates obtained under the above ideal system model are in fact achievable.) Finally, we define the soft-decision random vector  $L^N$  at the output of the SDG so that its  $i$ th coordinate is given by

$$L_i \triangleq (Y^N, U^{i-1}), \quad 1 \leq i \leq N.$$

(If it were not for the modeling assumption (21), it would be appropriate to use  $\hat{U}^{i-1}$  in the definition of  $L_i$  instead of  $U^{i-1}$ .) This completes the specification of the probabilistic model for all parts of the system.

We first focus on the capacity of the channel  $W_N$  created by combining  $N$  copies of  $W$ . Since we have specified a uniform distribution for channel inputs, the applicable notion of capacity in this analysis is the *symmetric* capacity, defined as

$$C_{\text{sym}}(W) \triangleq C(W, Q_{\text{sym}}),$$

where  $Q_{\text{sym}}$  is the uniform distribution,  $Q_{\text{sym}}(0) = Q_{\text{sym}}(1) = 1/2$ . Likewise, the applicable cutoff rate is now the symmetric one, defined as

$$R_{0,\text{sym}}(W) \triangleq R_0(W, Q_{\text{sym}}).$$

In general, the symmetric capacity may be strictly less than the true capacity, so there is a penalty for using a uniform distribution at channel inputs. However, since we will be dealing with linear codes, the uniform distribution is the only appropriate distribution here. Fortunately, for many channels of practical interest, the uniform distribution is actually the optimal one for achieving the channel capacity and the cutoff rate. These are the class of symmetric channels. A binary-input channel is called *symmetric* if for each output letter  $y$  there exists a “paired” output letter  $y'$  (not necessarily distinct from  $y$ ) such that  $W(y|0) = W(y'|1)$ . Examples of symmetric channels include the BSC, the BEC, and the additive Gaussian noise channel with binary inputs. As shown in [11, p. 94], for a symmetric channel, the symmetric versions of channel capacity and cutoff rate coincide with the true ones.

We now turn to the analysis of the capacities of the bit-channels created by the SC architecture. The SC architecture splits the vector channel  $W_N$  into  $N$  bit-channels, which we will denote by  $W_N^{(i)}$ ,  $1 \leq i \leq N$ . The  $i$ th bit-channel connects the output  $U_i$  of  $\text{CE}_i$  to the input  $L_i$  of  $\text{SD}_i$ ,

$$W_N^{(i)} : U_i \rightarrow L_i = (Y^N, U^{i-1}),$$

and has symmetric capacity

$$C_{\text{sym}}(W_N^{(i)}) = I(U_i; L_i) = I(U_i; Y^N, U^{i-1}).$$

Here,  $I(U_i; L_i)$  denotes the mutual information between  $U_i$  and  $L_i$ . In the following analysis, we will be using the mutual information function and some of its basic properties, such as the chain rule. We refer to [27, Ch. 2] for definitions and a discussion of such basic material.

The aggregate symmetric capacity of the bit-channels is

calculated as

$$\begin{aligned} \sum_{i=1}^N C_{\text{sym}}(W_N^{(i)}) &= \sum_{i=1}^N I(U_i; Y^N U^{i-1}) \\ &\stackrel{(1)}{=} \sum_{i=1}^N I(U_i; Y^N | U^{i-1}) \stackrel{(2)}{=} I(U^N; Y^N) \\ &\stackrel{(3)}{=} I(X^N; Y^N) \stackrel{(4)}{=} \sum_{i=1}^N I(X_i; Y_i) = N C_{\text{sym}}(W) \end{aligned} \quad (22)$$

where the equality (1) is due to the fact that  $U_i$  and  $U^{i-1}$  are independent, (2) is by the chain rule, (3) by the 1-1 property of  $f_N$ , (4) by the memoryless property of the channel  $W$ . Thus, the aggregate symmetric capacity of the underlying  $N$  copies of  $W$  is preserved by the combining and splitting operations.

Our main interest in using the SC architecture is to obtain a gain in the aggregate cutoff rate. We define the normalized symmetric cutoff rate under SC decoding as

$$\bar{R}_{0,\text{sym}}(W_N) \triangleq \frac{1}{N} \sum_{i=1}^N R_{0,\text{sym}}(W_N^{(i)}). \quad (23)$$

The objective in applying the SC architecture may be stated as devising schemes for which

$$\bar{R}_{0,\text{sym}}(W_N) > R_{0,\text{sym}}(W) \quad (24)$$

holds by a significant margin. The ultimate goal would be to have  $\bar{R}_{0,\text{sym}}(W_N)$  approach  $C_{\text{sym}}(W)$  as  $N$  increases.

For specific examples of schemes that follow the SC architecture and achieve cutoff rate gains in the sense of (24), we refer to [26] and the references therein. We must mention in this connection that the multilevel coding scheme of Imai and Hirakawa [25] is perhaps the first example of the SC architecture in the literature, but the focus there was not to boost the cutoff rate. In the next section, we discuss polar coding as another scheme that conforms to the SC architecture and provides the type of cutoff rate gains envisaged by (24).

## VI. POLAR CODING

Polar coding is an example of a coding scheme that fits into the framework of the preceding section and has the property that

$$\bar{R}_{0,\text{sym}}(W_N) \rightarrow C_{\text{sym}}(W) \quad \text{as } N \rightarrow \infty. \quad (25)$$

The name ‘‘polar’’ refers to a phenomenon called ‘‘polarization’’ that will be described later in this section. We begin by describing the channel combining and splitting operations in polar coding.

### A. Channel combining and splitting for polar codes

The channel combining and splitting operations in polar coding follow the general principles already described in detail in Sect. V-A. We only need to describe the particular 1-1 transformation  $f_N$  that is used for constructing a polar code of size  $N$ . We will begin this description starting with  $N = 2$ .

The basic module of the channel combining operation in polar coding is shown in Fig. 9, in which two independent

copies of  $W : \{0, 1\} \rightarrow \mathcal{Y}$  are combined into a channel  $W_2 : \{0, 1\}^2 \rightarrow \mathcal{Y}^2$  using a 1-1 mapping  $f_2$  defined by

$$f_2(u_1, u_2) \triangleq (u_1 \oplus u_2, u_2) \quad (26)$$

where  $\oplus$  denotes modulo-2 addition in the binary field  $\mathbb{F}_2 = \{0, 1\}$ . We call the basic transform  $f_2$  the *kernel* of the construction. (We defer the discussion of how to find a suitable kernel until the end of this subsection.)

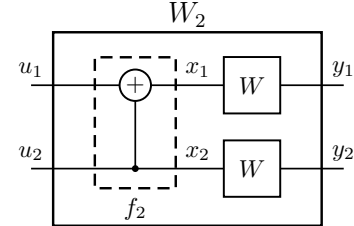


Fig. 9. Basic polar code construction.

Polar coding extends the above basic combining operation recursively to constructions of size  $N = 2^n$ , for any  $n \geq 1$ . For  $N = 4$ , the polar code construction is shown in Fig. 10, where 4 independent copies of  $W$  are combined by a 1-1 mapping  $f_4$  into a channel  $W_4$ .

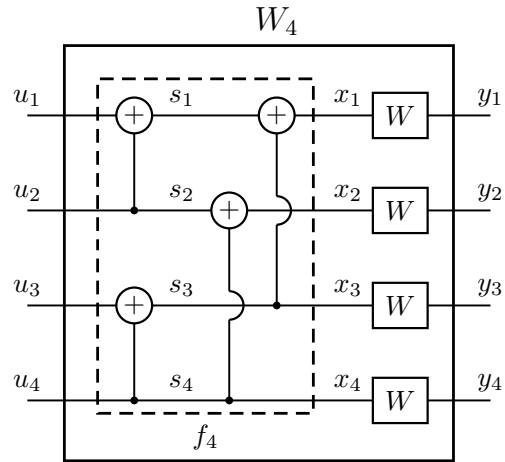


Fig. 10. Size-4 polar code construction.

The general form of the recursion in polar code construction is illustrated in Fig. 11 and can be expressed algebraically as

$$f_{2N}(u^{2N}) \triangleq (f_N(u^N) \oplus f_N(u_{N+1}^{2N}), f_N(u_{N+1}^{2N})), \quad (27)$$

where  $\oplus$  denotes the componentwise mod-2 addition of two vectors of the same length over  $\mathbb{F}_2$ .

The transform  $x^N = f_N(u^N)$  is linear over the vector space  $(\mathbb{F}_2)^N$ , and can be expressed as

$$x^N = u^N F_N,$$

where  $u^N$  and  $x^N$  are row vectors, and  $F_N$  is a matrix defined recursively as

$$F_{2N} = \begin{bmatrix} F_N & 0_N \\ F_N & F_N \end{bmatrix}, \quad \text{with } F_2 \triangleq \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix},$$

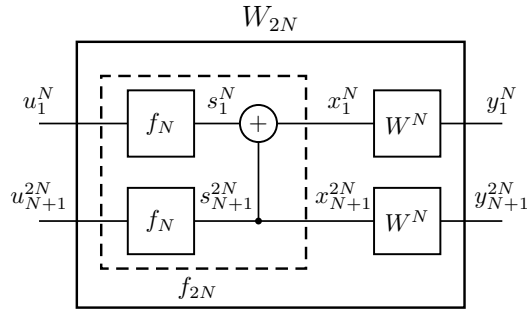


Fig. 11. Recursive extension of the polar code construction.

or, simply as

$$F_N = (F_2)^{\otimes n}, \quad n = \log N,$$

where the “ $\otimes$ ” in the exponent denotes Kronecker power of a matrix [5].

The recursive nature of the mapping  $f_N$  makes it possible to compute  $f_N(u^N)$  in time complexity  $O(N \log N)$ . The polar transform  $f_N$  is a “fast” transform over the field  $\mathbb{F}_2$ , akin to the “fast Fourier transform” of signal processing.

We wish to comment briefly on how to select a suitable kernel (basic module such as  $f_2$  above) to get the polar code construction started. In general, not only the kernel, but its size is also a design choice in polar coding; however, all else being equal, it is advantageous to use a small kernel to keep the complexity low. The specific kernel  $f_2$  above has been found by exhaustively studying all  $4!$  alternatives for a kernel of size  $N = 2$ , corresponding to all permutations (1-1 mappings) of binary vectors  $(u_1, u_2) \in \{0, 1\}^2$ . Six of the permutations are listed in Table I.

TABLE I  
BASIC PERMUTATIONS OF  $(u_1, u_2)$

No.	00	01	10	11	Matrix
1	00	01	10	11	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
2	00	10	01	11	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
3	00	11	10	01	$\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$
4	00	01	11	10	$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$
5	00	11	01	10	$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$
6	00	10	11	01	$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$

The title row displays the regular order of elements in  $\{0, 1\}^2$ , each subsequent row displays a particular permutation of the same elements. Each permutation listed in the table happens to be a linear transformation of  $(u_1, u_2)$ , with a transformation matrix as shown as the final entry of the related row. The remaining 18 permutations that are not listed in the table can be obtained as affine transformations of the six that are listed. For example, by adding (mod-2) a non-zero constant offset vector, such as 10, to each entry in the table, we obtain six additional permutations.

The first and the second permutations in the table are trivial permutations that provide no channel combining. The third and the fifth permutations are equivalent from a coding point of view; their matrices are column permutations of each other,

which corresponds to permuting the elements of the codeword during transmission—an operation that has no effect on the capacity or cutoff rate. The fourth and the sixth permutations are also equivalent to each other in the same sense that the third and the fifth permutations are. The fourth permutation is not suitable for our purposes since it does not provide any channel combining (entanglement) under the decoding order  $u_1$  first,  $u_2$  second. For the same reason, the sixth permutation is not suitable, either. The third and the fifth permutations (and their affine versions) remain as the only viable alternatives; and they are all equivalent from a capacity/cutoff rate viewpoint. Here, we use the third permutation since it is the simplest one among the eight viable candidates.

### B. Capacity and cutoff rate analysis for polar codes

For the analysis in this section, we will use the general setting and notation of Sect. V-B. We begin our analysis with the case  $N = 2$ . The basic transform (26) creates a channel  $W_2 : (U_1, U_2) \rightarrow (Y_1, Y_2)$  with transition probabilities

$$W_2(y_1, y_2 | u_1, u_2) = W(y_1 | u_1 \oplus u_2) W(y_2 | u_2).$$

This channel is split by the SC scheme into two bit-channels  $W_2^{(1)} : U_1 \rightarrow (Y_1, Y_2)$  and  $W_2^{(2)} : U_2 \rightarrow (Y_1, Y_2, U_1)$  with transition probabilities

$$W_2^{(1)}(y_1 y_2 | u_1) = \sum_{u_2 \in \{0, 1\}} Q_{\text{sym}}(u_2) W(y_1 | u_1 \oplus u_2) W(y_2 | u_2),$$

$$W_2^{(2)}(y_1 y_2 u_1 | u_2) = Q_{\text{sym}}(u_1) W(y_1 | u_1 \oplus u_2) W(y_2 | u_2).$$

Here, we introduce the alternative notation  $W^-$  and  $W^+$  to denote  $W_2^{(1)}$  and  $W_2^{(2)}$ , respectively. This notation will be particularly useful in the following discussion. We observe that the channel  $W^-$  treats  $U_2$  as pure noise; while,  $W^+$  treats  $U_1$  as an observed (known) entity. In other words, the transmission of  $U_1$  is hampered by interference from  $U_2$ ; while  $U_2$  “sees” a channel of diversity order two, after “canceling”  $U_1$ . Based on this interpretation, we may say that the polar transform creates a “bad” channel  $W^-$  and a “good” channel  $W^+$ . This statement can be justified by looking at the capacities of the two channels.

The symmetric capacities of  $W^-$  and  $W^+$  are given by

$$C_{\text{sym}}(W^-) = I(U_1; Y_1 Y_2), \quad C_{\text{sym}}(W^+) = I(U_2; Y_1 Y_2 U_1).$$

We observe that

$$C_{\text{sym}}(W^-) + C_{\text{sym}}(W^+) = 2C_{\text{sym}}(W), \quad (28)$$

which is a special instance of the general conservation law (22). The symmetric capacity is conserved, but redistributed unevenly. It follows from basic properties of mutual information function that

$$C_{\text{sym}}(W^-) \leq C_{\text{sym}}(W) \leq C_{\text{sym}}(W^+), \quad (29)$$

where the inequalities are strict unless  $C_{\text{sym}}(W)$  equals 0 or 1. For a proof, we refer to [5].

We will call a channel  $W$  *extreme* if  $C_{\text{sym}}(W)$  equals 0 or 1. Extreme channels are those for which there is no need for coding: if  $C_{\text{sym}}(W) = 1$ , one can send data uncoded; if

$C_{\text{sym}}(W) = 0$ , no code will help. Inequality (29) states that, unless the channel  $W$  is extreme, the size-2 polar transform creates a channel  $W^+$  that is strictly better than  $W$ , and a second channel  $W^-$  that is strictly worse than  $W$ . By doing so, the size-2 transform starts the polarization process.

As regards the cutoff rates, we have

$$R_{0,\text{sym}}(W^-) + R_{0,\text{sym}}(W^+) \geq 2R_{0,\text{sym}}(W), \quad (30)$$

where the inequality is strict unless  $W$  is extreme. This result, proved in [5], states that the basic transform *always* creates a cutoff rate gain, except when  $W$  is extreme.

An equivalent form of (30), which is the one that was actually proved in [5], is the following inequality about the Bhattacharyya parameters,

$$Z(W^-) + Z(W^+) \leq 2Z(W), \quad (31)$$

where strict inequality holds unless  $W$  is extreme. The equivalence of (30) and (31) is easy to see from the relation

$$R_{0,\text{sym}}(W) = 1 - \log[1 + Z(W)],$$

which is a special form of (15) with  $Q = Q_{\text{sym}}$ .

Given that the size-2 transform improves the cutoff rate of any given channel  $W$  (unless  $W$  is already extreme in which case there is no need to do anything), it is natural to seek methods of applying the same method recursively so as to gain further improvements. This is the main intuitive idea behind polar coding. As the cutoff-rate gains are accumulated over each step of recursion, the synthetic bit-channels that are created in the process keep moving towards extremes.

To see how recursion helps improve the cutoff rate as the size of the polar transform is doubled, let us consider the next step of the construction,  $N = 4$ . The key recursive relationships that tie the size-4 construction to size-2 construction are the following:

$$W_4^{(1)} \equiv (W_2^{(1)})^-, \quad W_4^{(2)} \equiv (W_2^{(1)})^+, \quad (32)$$

$$W_4^{(3)} \equiv (W_2^{(2)})^-, \quad W_4^{(4)} \equiv (W_2^{(2)})^+. \quad (33)$$

The first claim  $W_4^{(1)} \equiv (W_2^{(1)})^-$  means that  $W_4^{(1)}$  is equivalent to the bad channel obtained by applying a size-2 transform on two independent copies of  $W_2^{(1)}$ . The other three claims can be interpreted similarly.

To prove the validity of (32) and (33), let us refer to Fig. 10 again. Let  $(U^4, S^4, X^4, Y^4)$  denote the ensemble of random vectors that correspond to the signals  $(u^4, s^4, x^4, y^4)$  in the polar transform circuit. In accordance with the modeling assumptions of Sect. V-B, the random vector  $X^4$  is uniformly distributed over  $\{0,1\}^4$ . Since both  $S_1^4$  and  $U^4$  are in 1-1 correspondence with  $X_1^4$ , they, too, are uniformly distributed over  $\{0,1\}^4$ . Furthermore, the elements of  $X^4$  are i.i.d. uniform over  $\{0,1\}$ , and similarly for the elements of  $S^4$ , and of  $U^4$ .

Let us now focus on Fig. 12 which depicts the relevant part of Fig. 10 for the present discussion. Consider the two channels

$$W' : S_1 \rightarrow (Y_1, Y_3), \quad W'' : S_2 \rightarrow (Y_2, Y_4),$$

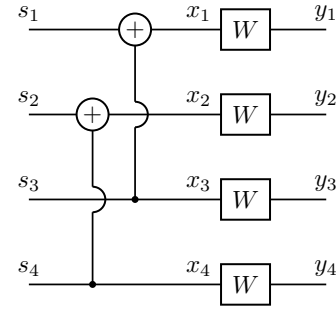


Fig. 12. Intermediate stage of splitting for size-4 polar code construction.

embedded in the diagram. It is clear that

$$W' \equiv W'' \equiv W_2^{(1)} \equiv W^-.$$

Furthermore, the two channels  $W'$  and  $W''$  are independent. This is seen by noticing that  $W'$  is governed by the set of random variables  $(S_1, S_3, Y_1, Y_3)$ , which is disjoint from the set of variables  $(S_2, S_4, Y_2, Y_4)$  that govern  $W''$ .

Returning to the size-4 construction of Fig. 10, we now see that the effective channel seen by the pair of inputs  $(U_1, U_2)$  is the combination of  $W'$  and  $W''$ , or equivalently, of two independent copies of  $W_2^{(1)} \equiv W^-$ , as shown in Fig. 13. The first pair of claims (32) follows immediately from this figure.

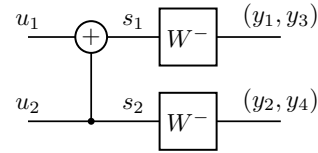


Fig. 13. A size-2 polar code construction embedded in a size-4 construction.

The second pair of claims (33) follows by observing that, after decoding  $(U_1, U_2)$ , the effective channel seen by the pair of inputs  $(U_3, U_4)$  is the combination of two independent copies of  $W_2^{(2)} \equiv W^+$  as shown in Fig. 14.

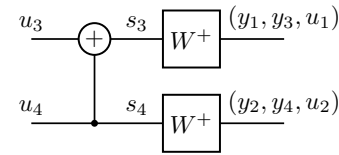


Fig. 14. A second size-2 polar code construction inside a size-4 construction.

The following conservation rules are immediate from (28).

$$C_{\text{sym}}(W^{--}) + C_{\text{sym}}(W^{+-}) = 2C_{\text{sym}}(W^-),$$

$$C_{\text{sym}}(W^{+-}) + C_{\text{sym}}(W^{++}) = 2C_{\text{sym}}(W^+).$$

Likewise, we have, from (30),

$$R_{0,\text{sym}}(W^{--}) + R_{0,\text{sym}}(W^{+-}) \geq 2R_{0,\text{sym}}(W^-),$$

$$R_{0,\text{sym}}(W^{+-}) + R_{0,\text{sym}}(W^{++}) \geq 2R_{0,\text{sym}}(W^+).$$

Here, we extended the notation and used  $W^{--}$  to denote  $(W^-)^-$ , and similarly for  $W^{++}$ , etc.

If we normalize the aggregate cutoff rates for  $N = 4$  and compare with the normalized cutoff rate for  $N = 2$ , we obtain

$$\bar{R}_{0,\text{sym}}(W_4) \geq \bar{R}_{0,\text{sym}}(W_2) \geq R_{0,\text{sym}}(W).$$

These inequalities are strict unless  $W$  is extreme.

The recursive argument given above can be applied to the situation in Fig. 11 to show that for any  $N = 2^n$ ,  $n \geq 1$ , and  $1 \leq i \leq N$ , the following relations hold

$$W_{2N}^{(2i-1)} \equiv (W_N^{(i)})^-, \quad W_{2N}^{(2i)} \equiv (W_N^{(i)})^+,$$

from which it follows that

$$\bar{R}_{0,\text{sym}}(W_{2N}) \geq \bar{R}_{0,\text{sym}}(W_N).$$

These results establish that the sequence of normalized cutoff rates  $\{\bar{R}_{0,\text{sym}}(W_N)\}$  is monotone non-decreasing in  $N$ . Since  $\bar{R}_{0,\text{sym}}(W_N) \leq C_{\text{sym}}(W)$  for all  $N$ , the sequence must converge to a limit. It turns out, as might be expected, that this limit is the symmetric capacity  $C_{\text{sym}}(W)$ . We examine the asymptotic behavior of the polar code construction process in the next subsection.

### C. Polarization and elimination of the outer code

As the construction size in the polar transform is increased, gradually a “polarization” phenomenon takes holds. All channels  $\{W_N^{(i)} : 1 \leq i \leq N\}$  created by the polar transform, except for a vanishing fraction, approach extreme limits (becoming near perfect or useless) with increasing  $N$ . One form of expressing polarization more precisely is the following. For any fixed  $\delta > 0$ , the channels created by the polar transform satisfy

$$\frac{1}{N} \left| \{i : R_{0,\text{sym}}(W_N^{(i)}) > 1 - \delta\} \right| \rightarrow C_{\text{sym}}(W) \quad (34)$$

and

$$\frac{1}{N} \left| \{i : R_{0,\text{sym}}(W_N^{(i)}) < \delta\} \right| \rightarrow 1 - C_{\text{sym}}(W) \quad (35)$$

as  $N$  increases. ( $|\mathcal{A}|$  denotes the number of elements in set  $\mathcal{A}$ .) A proof of this result using martingale theory can be found in [5]; for a recent simpler proof that avoids martingales, we refer to [28].

As an immediate corollary to (34), we obtain (25), establishing the main goal of this analysis. While this result is very reassuring, there are many remaining technical details that have to be taken care of before we can claim to have a practical coding scheme. First of all, we should not forget that the validity of (25) rests on the assumption (21) that there are no errors in the SC decoding chain. We may argue that we can satisfy assumption (21) to any desired degree of accuracy by using convolutional codes of sufficiently long constraint lengths. Luckily, it turns out using such convolutional codes is unnecessary to have a practically viable scheme. The polarization phenomenon creates sufficient number of sufficiently good channels fast enough that the validity of (25) can be maintained without any help from an outer convolutional code

and sequential decoder. The details of this last step of polar code construction are as follows.

Let us reconsider the scheme in Fig. 8. At the outset, the plan was to operate the  $i$ th convolutional encoder  $\text{CE}_i$  at a rate just below the symmetric cutoff rate  $R_{0,\text{sym}}(W_N^{(i)})$ . However, in light of the polarization phenomenon, we know that almost all the cutoff rates  $R_{0,\text{sym}}(W_N^{(i)})$  are clustered around 0 or 1 for  $N$  large. This suggests rounding off the rates of all convolutional encoders to 0 or 1, effectively eliminating the outer code. Such a revised scheme is highly attractive due to its simplicity, but dispensing with the outer code exposes the system to unmitigated error propagation in the SC chain.

To analyze the performance of the scheme that has no protection by an outer code, let  $\mathcal{A}$  denote the set of indices  $i \in \{1, \dots, N\}$  of input variables  $U_i$  that will carry data at rate 1. We call  $\mathcal{A}$  the set of “active” variables. Let  $\mathcal{A}^c$  denote the complement of  $\mathcal{A}$ , and call this set the set of “frozen” variables. We will denote the active variables collectively by  $U_{\mathcal{A}} \triangleq (U_i : i \in \mathcal{A})$  and the frozen ones by  $U_{\mathcal{A}^c} \triangleq (U_i : i \in \mathcal{A}^c)$ , each vector regarded as a subvector of  $U^N$ . Let  $K$  denote the size of  $\mathcal{A}$ .

Encoding is done by setting  $U_{\mathcal{A}} = D^K$  and  $U_{\mathcal{A}^c} = b^{N-K}$  where  $D^K$  is user data equally likely to take any value in  $\{0, 1\}^K$  and  $b^{N-K} \in \{0, 1\}^{N-K}$  is a fixed pattern. The user data may change from block to the next, but the frozen pattern remains the same and is known to the decoder. This system carries  $K$  bits of data in each block of  $N$  channel uses, for a transmission rate of  $R = K/N$ .

At the receiver, we suppose that there is an SC decoder that computes its decision  $\hat{U}^N$  by calculating the likelihood ratio

$$L_i \triangleq \Pr(U_i = 0 | Y^N, \hat{U}^{i-1}) / \Pr(U_i = 1 | Y^N, \hat{U}^{i-1}),$$

and setting

$$\hat{U}_i = \begin{cases} U_i, & \text{if } i \in \mathcal{A}^c, \\ 0, & \text{if } i \in \mathcal{A} \text{ and } L_i > 1; \\ 1, & \text{if } i \in \mathcal{A} \text{ and } L_i \leq 1, \end{cases}$$

successively, starting with  $i = 1$ . Since the variables  $U_{\mathcal{A}^c}$  are fixed to  $b^{N-K}$ , this decoding rule can be implemented at the decoder. The probability of frame error for this system is given by

$$P_e(\mathcal{A}, b^{N-K}) \triangleq P(\hat{U}_{\mathcal{A}} \neq U_{\mathcal{A}} | U_{\mathcal{A}^c} = b^{N-K}).$$

For a symmetric channel, the error probability  $P_e(\mathcal{A}, b^{N-K})$  does not depend on  $b^{N-K}$  [5]. A convenient choice in that case may be to set  $b^{N-K}$  to the zero vector. For a general channel, we consider the average of the error probability over all possible choices for  $b^{N-K}$ , namely,

$$P_e(\mathcal{A}) \triangleq \frac{1}{2^{(N-K)}} \sum_{b^{N-K} \in \{0,1\}^{N-K}} P(\hat{U}_{\mathcal{A}} \neq U_{\mathcal{A}} | U_{\mathcal{A}^c} = b^{N-K}).$$

It is shown in [5] that

$$P_e(\mathcal{A}) \leq \sum_{i \in \mathcal{A}} Z(W_N^{(i)}), \quad (36)$$

where  $Z(W_N^{(i)})$  is the Bhattacharyya parameter of  $W_N^{(i)}$ .

The bound (36) suggests that  $\mathcal{A}$  should be chosen so as to minimize the upper bound on  $P_e(\mathcal{A})$ . The performance attainable by such a design rule can be calculated directly from the following polarization result from [29].

For any fixed  $\beta < \frac{1}{2}$ , the Bhattacharyya parameters created by the polar transform satisfy

$$\frac{1}{N} \left| \{i : Z(W_N^{(i)}) < 2^{-N^\beta}\} \right| \rightarrow C_{\text{sym}}(W). \quad (37)$$

In particular, if we fix  $\beta = 0.49$ , the fraction of channels  $W_N^{(i)}$  in the population  $\{W_N^{(i)} : 1 \leq i \leq N\}$  satisfying

$$Z(W_N^{(i)}) < 2^{-N^{0.49}} \quad (38)$$

approaches the symmetric capacity  $C_{\text{sym}}(W)$  as  $N$  becomes large. So, if we fix the rate  $R < C_{\text{sym}}(W)$ , then for all  $N$  sufficiently large, we will be able to select an active set  $\mathcal{A}_N$  of size  $K = NR$  such that (38) holds for each  $i \in \mathcal{A}_N$ . With  $\mathcal{A}_N$  selected in this way, the probability of error is bounded as

$$P_e(\mathcal{A}_N) < \sum_{i \in \mathcal{A}_N} Z(W_N^{(i)}) \leq N 2^{-N^{0.49}} \rightarrow 0.$$

This establishes the feasibility of constructing polar codes that operate at any rate  $R < C_{\text{sym}}(W)$  with a probability of error going to 0 exponentially as  $\approx 2^{-\sqrt{N}}$ .

This brings us to the end of our discussion of polar codes. We will close by mentioning two important facts that relate to complexity. The SC decoding algorithm for polar codes can be implemented in complexity  $O(N \log N)$  [5]. The construction complexity of polar codes, namely, the selection of an optimal (subject to numerical precision) set  $\mathcal{A}$  of active channels (either by computing the Bhattacharyya parameters  $\{Z_N^{(i)}\}$  or some related set of quality parameters) can be done in complexity  $O(N)$ , as shown in the sequence of papers [30], [31], and [32].

## VII. SUMMARY

In this paper we gave an account of polar coding from a historical perspective, tracing the original line of thinking that led to its development.

The key motivation for polar coding was to boost the cutoff rate of sequential decoding. The schemes of Pinsker and Massey suggested a two-step mechanism: first build a vector channel from independent copies of a given channel; next, split the vector channel into correlated subchannels. With proper combining and splitting, it is possible to obtain an improvement in the aggregate cutoff rate. Polar coding is a recursive implementation of this basic idea. The recursiveness renders polar codes both analytically tractable, which leads to an explicit code construction algorithm, and also makes it possible to encode and decode these codes at low-complexity.

Although polar coding was originally intended to be the inner code in a concatenated scheme, it turned out (to our pleasant surprise) that the inner code was so reliable that there was no need for the outer convolutional code or the sequential decoder. However, to further improve polar coding, one could still consider adding an outer coding scheme, as originally planned.

## APPENDIX

### DERIVATION OF THE PAIRWISE ERROR BOUND

This appendix provides a proof of the pairwise error bound (14). The proof below is standard textbook material. It is reproduced here for completeness and to demonstrate the simplicity of the basic idea underlying the cutoff rate.

Let  $\mathcal{C} = \{x^N(1), \dots, x^N(M)\}$  be a specific code, let  $R = (1/N) \log M$  be the rate of the code. Fix two distinct messages  $m \neq m'$ ,  $1 \leq m, m' \leq M$ , and define  $P_{m,m'}(\mathcal{C})$  as in (12). Then,

$$\begin{aligned} P_{m,m'}(\mathcal{C}) &\triangleq \sum_{y^N \in E_{m,m'}(\mathcal{C})} W^N(y^N | x^N(m)) \\ &\stackrel{(1)}{\leq} \sum_{y^N} W^N(y^N | x^N(m)) \sqrt{\frac{W^N(y^N | x^N(m'))}{W^N(y^N | x^N(m))}} \\ &= \sum_{y^N} \sqrt{W^N(y^N | x^N(m)) W^N(y^N | x^N(m'))} \\ &\stackrel{(2)}{=} \prod_{n=1}^N \left[ \sum_{y_n} \sqrt{W(y_n | x_n(m)) W(y_n | x_n(m'))} \right] \\ &\stackrel{(3)}{=} \prod_{n=1}^N Z_{m,m'}(n), \end{aligned}$$

where the inequality (1) follows by the simple observation that

$$\sqrt{\frac{W^N(y^N | x^N(m'))}{W^N(y^N | x^N(m))}} \geq \begin{cases} 1, & y^N \in E_{m,m'}(\mathcal{C}) \\ 0, & y^N \notin E_{m,m'}(\mathcal{C}). \end{cases}$$

equality (2) follows by the memoryless channel assumption, and (3) by the definition

$$Z_{m,m'}(n) \triangleq \sum_y \sqrt{W(y | x_n(m)) W(y | x_n(m'))}. \quad (39)$$

At this point the analysis becomes dependent on the specific code structure. To continue, we consider the ensemble average of the pairwise error probability,  $\bar{P}_{m,m'}(N, Q)$ , defined by (13).

$$\begin{aligned} \bar{P}_{m,m'}(N, Q) &\leq \overline{\prod_{n=1}^N Z_{m,m'}(n)} \\ &\stackrel{(1)}{=} \prod_{n=1}^N \bar{Z}_{m,m'}(n) \stackrel{(2)}{=} [\bar{Z}_{m,m'}(1)]^N \end{aligned}$$

where the overbar denotes averaging with respect to the code ensemble, the equality (1) is due to the independence of the random variables  $\{Z_{m,m'}(n) : 1 \leq n \leq N\}$ , and (2) is by the fact that the same set of random variables are identically-distributed. Finally, we note that,

$$\begin{aligned} \bar{Z}_{m,m'}(1) &= \overline{\sum_y \sqrt{W(y | X_1(m)) W(y | X_1(m'))}} \\ &= \sum_y \left[ \sqrt{W(y | X_1(m))} \right] \left[ \sqrt{W(y | X_1(m'))} \right] \\ &= \sum_y \left[ \sum_x Q(x) \sqrt{W(y | x)} \right]^2. \end{aligned}$$

In view of the definition (10), this proves that  $\bar{P}_{m,m'}(N, Q) \leq 2^{-N R_0(Q)}$ , for any  $m \neq m'$ .

# ACKNOWLEDGEMENT

The author would like to thank Senior Editor M. Pursley and anonymous reviewers for extremely helpful suggestions to improve the paper. This work was supported by the FP7 Network of Excellence NEWCOM# under grant agreement 318306.

# REFERENCES

- [1] C. E. Shannon, "A mathematical theory of communication," *Bell System Tech. J.*, vol. 27, pp. 379–423, 623–656, July–Oct. 1948.
- [2] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: Turbo-codes," *IEEE Trans. Comm.*, vol. 44, pp. 1261–1271, Oct. 1996.
- [3] R. G. Gallager, "Low-density parity-check codes," *IRE Trans. Inform. Theory*, vol. 8, pp. 21–28, Jan. 1962.
- [4] D. J. Costello and G. D. Forney Jr., "Channel coding: The road to channel capacity," *Proc. IEEE*, vol. 95, pp. 1150–1177, June 2007.
- [5] E. Arkan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. Inform. Theory*, vol. 55, pp. 3051–3073, July 2009.
- [6] M. S. Pinsker, "On the complexity of decoding," *Problemy Peredachi Informatsii*, vol. 1, no. 1, pp. 84–86, 1965.
- [7] J. L. Massey, "Capacity, cutoff rate, and coding for a direct-detection optical channel," *IEEE Trans. Comm.*, vol. 29, pp. 1615–1621, Nov. 1981.
- [8] E. Arkan, "Sequential decoding for multiple access channels," Tech. Rep. LIDS-TH-1517, Lab. Inf. Dec. Syst., M.I.T., 1985.
- [9] R. M. Fano, *Transmission of Information*, 4th ed. The M.I.T. Press, 1961.
- [10] R. G. Gallager, "A simple derivation of the coding theorem and some applications," *IEEE Trans. Inform. Theory*, vol. 11, pp. 3–18, Jan. 1965.
- [11] R. G. Gallager, *Information Theory and Reliable Communication*. Wiley: New York, 1968.
- [12] R. G. Gallager, "The random coding bound is tight for the average code," *IEEE Trans. Inform. Theory*, vol. 19, pp. 244–246, Mar. 1973.
- [13] E. Arkan, "An inequality on guessing and its application to sequential decoding," *IEEE Trans. Inform. Theory*, vol. 42, pp. 99–105, Jan. 1996.
- [14] P. Elias, "Coding for noisy channels," in *IRE Convention Record*, pt. 4, pp. 37–46, March 1955. (Reprinted in: D. Slepian (Ed.), "Key Papers in The Development of Information Theory", IEEE Press, 1974, pp. 102–111).
- [15] J. M. Wozencraft, "Sequential decoding for reliable communication," Tech. Report 325, Res. Lab. Elect., M.I.T., Aug. 1957.
- [16] J. M. Wozencraft and B. Reiffen, *Sequential Decoding*. MIT Press: Cambridge, Mass., 1961.
- [17] R. M. Fano, "A heuristic discussion of probabilistic decoding," *IEEE Trans. Inform. Theory*, vol. 9, pp. 64–74, Apr. 1963.
- [18] J. W. Layland and W. A. Lushbaugh, "A flexible high-speed sequential decoder for deep space channels," *IEEE Trans. Comm. Tech.*, vol. 19, pp. 813–820, Oct. 1971.
- [19] G. D. Forney Jr., "1995 Shannon lecture—Performance and complexity," *IEEE Inform. Theory Soc. Newslett.*, vol. 46, pp. 3–4, Mar. 1996.
- [20] B. Reiffen, "Sequential encoding and decoding for the discrete memoryless channel," Tech. Report 374, Res. Lab. Elect., M.I.T., 1960.
- [21] I. M. Jacobs and E. R. Berlekamp, "A lower bound to the distribution of computation for sequential decoding," *IEEE Trans. Inform. Theory*, vol. 13, pp. 167–174, Apr. 1967.
- [22] E. Arkan, "An upper bound on the cutoff rate of sequential decoding," *IEEE Trans. Inform. Theory*, vol. 34, pp. 55–63, Jan. 1988.
- [23] R. G. Gallager, "A perspective on multiaccess channels," *IEEE Trans. Inform. Theory*, vol. 31, pp. 124–142, Mar. 1985.
- [24] P. Elias, "Error-free coding," *IRE Trans. Inform. Theory*, vol. 4, pp. 29–37, Sep. 1954.
- [25] H. Imai and S. Hirakawa, "A new multilevel coding method using error-correcting codes," *IEEE Trans. Inform. Theory*, vol. 23, pp. 371–377, May 1977.
- [26] E. Arkan, "Channel combining and splitting for cutoff rate improvement," *IEEE Trans. Inform. Theory*, vol. 52, pp. 628–639, Feb. 2006.
- [27] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. John Wiley & Sons, 2006.

- [28] M. Alsan and E. Telatar, "A simple proof of polarization and polarization for non-stationary channels," in *Proc. IEEE Int. Symp. Inform. Theory*, pp. 301–305, 2014.
- [29] E. Arkan and E. Telatar, "On the rate of channel polarization," in *Proc. 2009 IEEE Int. Symp. Inform. Theory*, (Seoul, S. Korea), pp. 1493–1495, 28 June - July 3 2009.
- [30] R. Mori and T. Tanaka, "Performance and construction of polar codes on symmetric binary-input memoryless channels," in *Proc. IEEE Int. Symp. Inform. Theory*, pp. 1496–1500, 2009.
- [31] I. Tal and A. Vardy, "How to construct polar codes," *IEEE Trans. Inform. Theory*, vol. 59, pp. 6562–6582, Oct. 2013.
- [32] R. Pedarsani, S. H. Hassani, I. Tal, and E. Telatar, "On the construction of polar codes," in *Proc. IEEE Int. Symp. Inform. Theory*, pp. 11–15, 2011.



**Erdal Arkan** (S84), (M79), (SM94), (F'11) was born in Ankara, Turkey, in 1958. He received the B.S. degree from the California Institute of Technology, Pasadena, CA, in 1981, and the S.M. and Ph.D. degrees from the Massachusetts Institute of Technology, Cambridge, MA, in 1982 and 1985, respectively, all in Electrical Engineering. Since 1987 he has been with the Electrical-Electronics Engineering Department of Bilkent University, Ankara, Turkey, where he works as a professor. He is the recipient of 2010 IEEE Information Theory Society Paper Award and the 2013 IEEE W.R.G. Baker Award, both for his work on polar coding.