How to Construct Polar Codes

Ido Tal, *Member, IEEE*, and Alexander Vardy, *Fellow, IEEE*

***Abstract*—A method for efficiently constructing polar codes is presented and analyzed. Although polar codes are explicitly defined, straightforward construction is intractable since the resulting polar bit-channels have an output alphabet that grows exponentially with the code length. Thus, the core problem that needs to be solved is that of faithfully approximating a bit-channel with an intractably large alphabet by another channel having a manageable alphabet size. We devise two approximation methods which "sandwich" the original bit-channel between a degraded and an upgraded version thereof. Both approximations can be efficiently computed and turn out to be extremely close in practice. We also provide theoretical analysis of our construction algorithms, proving that for any fixed $\varepsilon > 0$ and all sufficiently large code lengths n, polar codes whose rate is within ε of channel capacity can be constructed in time and space that are both linear in n.**

***Index Terms*—Channel degrading and upgrading, channel polarization, construction algorithms, polar codes.**

I. Introduction

POLAR codes, invented by Arıkan [3], achieve the capacity of arbitrary binary-input symmetric discrete memoryless channels (DMCs). Moreover, they have low encoding and decoding complexity and an explicit construction. Following Arıkan's seminal paper [3], his results have been extended in a variety of important ways. In [22], polar codes have been generalized to symmetric DMCs with *nonbinary* input alphabet. In [14], the polarization phenomenon has been studied for arbitrary kernel matrices, rather than Arıkan's original 2×2 polarization kernel, and error exponents were derived for each such kernel. It was shown in [24] that, under *list-decoding*, polar codes can achieve remarkably good performance at short code lengths. In terms of applications, polar coding has been used with great success in the context of multiple-access channels [2], [23], wiretap channels [16], data compression [1], [4], write-once channels [6], and channels with memory [21]. In this paper, however, we will restrict our attention to the original setting introduced by Arıkan in [3]. Namely, we focus on binary-input, discrete, memoryless, symmetric channels, with the standard 2×2 polarization kernel under standard successive cancellation decoding.

Manuscript received October 30, 2011; revised April 10, 2013; accepted June 05, 2013. Date of publication July 10, 2013; date of current version September 11, 2013. This work was supported in part by the National Science Foundation. This paper was presented in part at the 2010 IEEE Workshop on Information Theory, Dublin, Ireland.

I. Tal is with the Department of Electrical Engineering, Technion—Israel Institute of Technology, Haifa 32000, Israel (e-mail: idotal@ieee.org).

A. Vardy is with the Department of Electrical and Computer Engineering and the Department of Computer Science and Engineering, University of California San Diego, La Jolla, CA 92093-0407 USA (e-mail: avardy@ucsd.edu).

Communicated by E. Arıkan, Associate Editor for Coding Theory.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIT.2013.2272694

Although the construction of polar codes is *explicit*, there is only one known instance—namely, the binary erasure channel (BEC)—where the construction is also *efficient*. A first attempt at an efficient construction of polar codes in the general case was made by Mori and Tanaka [17], [18]. Specifically, it is shown in [17] that a key step in the construction of polar bit-channels can be viewed as an instance of density evolution [20]. Based on this observation, Mori and Tanaka [18] proposed a construction algorithm utilizing convolutions and proved that the number of convolutions needed scales linearly with the code length. However, as indeed noted in [17], it is not clear how one would implement such convolutions to be sufficiently precise on one hand while being tractable on the other hand.

In this paper, we further extend the ideas of [17] and [18]. An exact implementation of the convolutions discussed in [17] and [18] implies an algorithm with memory requirements that grow exponentially with the code length. It is thus impractical. Alternatively, one could use quantization (binning) to try and reduce the memory requirements. However, for such quantization scheme to be of interest, it must satisfy two conditions. First, it must be fast enough, which usually translates into a rather small number of quantization levels (bins). Second, after the calculations have been carried out, we must be able to interpret them in a precise manner. That is, the quantization operation introduces inherent inaccuracy into the computation, which we should be able to account for so as to ultimately make a precise statement.

Our aim in this paper is to provide a method by which polar codes can be efficiently constructed. Our main contribution consists of two approximation methods. In both methods, the memory limitations are specified, and not exceeded. One method is used to get a lower bound on the probability of error of each polar bit-channel, while the other is used to obtain an upper bound. The quantization used to derive a lower bound on the probability of error is called a *degrading quantization*, while the other is called an *upgrading quantization*. Both quantizations transform the "current channel" into a new one with a smaller output alphabet. The degrading quantization results in a channel degraded with respect to the original one, while the upgrading quantization results in a channel such that the original channel is degraded with respect to it.

The fidelity of both degrading and upgrading approximations is a function of a parameter μ, which can be freely set to an arbitrary integer value. Generally speaking, the larger μ is the better the approximation. The running time needed in order to approximate all n polar bit-channels is $O(n \cdot \mu^2 \log \mu)$.

Our results relate to both theory and practice of polar codes. In practice, it turns out that the degrading and upgrading approximations are typically very close, even for relatively small values of the fidelity parameter μ. This is illustrated in what follows with the help of two examples.

Example 1: Consider a polar code of length $n = 2^{20}$ for the binary symmetric channel (BSC) with crossover probability

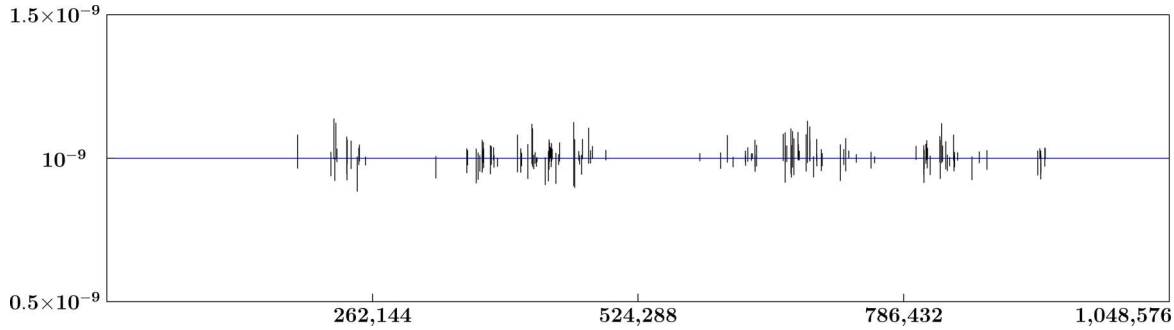


Fig. 1. Upper and lower bounds on the bit-channel probabilities of error for a polar code of length $n = 1\,048\,576$ on BSC(0.11), computed using degrading and upgrading algorithms with $\mu = 256$. Only those 132 bit-channels for which the gap between the upper and lower bounds crosses the 10^{-9} threshold are shown.

0.11. Let $\mathcal{W}_0, \mathcal{W}_1, \ldots, \mathcal{W}_{n-1}$ be the corresponding bit-channels (see the next section for a rigorous definition of a bit-channel). The basic task in the construction of polar codes is that of classifying bit-channels into those that are "good" and those that are "bad." Let $P_e(\mathcal{W}_i)$ denote the probability of error on the ith bit-channel (see (13) for a precise definition of this quantity) for $i = 0, 1, \ldots, n-1$. We arbitrarily choose a threshold of 10^{-9} and say that the ith bit channel is good if $P_e(\mathcal{W}_i) \leqslant 10^{-9}$ and bad otherwise. How well do our algorithms perform in determining for each of the n bit-channels whether it is good or bad?

Let us set $\mu = 256$ and compute upper and lower bounds on $P_e(\mathcal{W}_i)$ for all i, using the degrading and upgrading quantizations, respectively. The results of this computation are illustrated in Fig. 1. In 1,048,444 out of the 1,048,576 cases, we can provably classify the bit-channels into good and bad. Fig. 1 depicts the remaining 132 bit-channels for which the upper bound is above the threshold whereas the lower bound is below the threshold. The horizontal axis in Fig. 1 is the bit-channel index while the vertical axis is the gap between the two bounds. We see that the gap between the upper and lower bounds, and thus, the remaining uncertainty as to the true value of $P_e(\mathcal{W}_i)$ is very small in all cases. □

Example 2: Now suppose we wish to construct a polar code of a given length n having the best possible rate while guaranteeing a certain block-error probability P_{block} under successive cancellation decoding. Arıkan [3, Proposition 2] provides[1] a union bound on the block-error rate of polar codes:

$$P_{\text{block}} \leqslant \sum_{i \in \mathcal{A}} P_e(\mathcal{W}_i) \tag{1}$$

where $\mathcal{A}$ is the *information set* for the code (the set of unfrozen bit-channels). The construction problem for polar codes can be phrased (cf., [3, Sec. IX]) as the problem of choosing an information set $\mathcal{A}$ of a given size $|\mathcal{A}| = k$ so as to minimize the right-hand side of (1). Assuming the underlying channel W and the code length n are fixed, let

$$P_{W,n}(k) \stackrel{\text{def}}{=} \min_{|\mathcal{A}|=k} \sum_{i \in \mathcal{A}} P_e(\mathcal{W}_i). \tag{2}$$

Using our degrading and upgrading algorithms, we can efficiently compute upper and lower bounds on $P_{W,n}(k)$. These are plotted in Fig. 2 for two underlying channels: BSC with crossover probability 0.001 and the binary-input additive white Gaussian noise (AWGN) channel with a symbol SNR of 5.00 dB (noise variance $\sigma^2 = 0.1581$). In all[2] our calculations, the value of μ did not exceed 512.

As can be seen from Fig. 2, the bounds effectively coincide. As an example, consider polar codes of length 2^{20} and suppose we wish to guarantee $P_{W,n}(k) \leqslant 10^{-6}$. What is the best possible rate of such a code? According to Fig. 2(a), we can efficiently construct (specify the rows of a generator matrix) a polar code of rate $R = 0.9732$. On the other hand, we can also prove that there is no choice of an information set $\mathcal{A}$ in (2) that would possibly produce a polar code of rate $R \geqslant 0.9737$. According to Fig. 2(b), the corresponding numbers for the binary-input AWGN channel are 0.9580 and 0.9587. In practice, such minute differences in the code rate are negligible. □

From a theoretical standpoint, one of our main contributions is the following theorem. In essence, the theorem asserts that capacity-achieving polar codes can be constructed in time that is polynomial (in fact, linear) in their length n.

Theorem 1: Let W be a binary-input, symmetric, DMC of capacity $I(W)$. Fix arbitrary real constants $\varepsilon > 0$ and $\beta < 1/2$. Then, there exists an even integer

$$\mu_0 = \mu_0(W, \varepsilon, \beta), \tag{3}$$

which does *not* depend on the code length n, such that the following holds. For all even integers $\mu \geqslant \mu_0$ and all sufficiently large code lengths $n = 2^m$, there is a construction algorithm with running time $O(n \cdot \mu^2 \log \mu)$ that produces a polar code for W of rate $R \geqslant I(W) - \varepsilon$ such that $P_{\text{block}} \leqslant 2^{-n^\beta}$, where P_{block} is the probability of codeword error under successive cancellation decoding.

We defer the proof of Theorem 1 to Section VIII. Here, let us briefly discuss two immediate consequences of this theorem. First, observe that for a given channel W and any fixed ε and β, the integer μ_0 in (3) is a constant. Setting our fidelity parameter in Theorem 1 to $\mu = \mu_0$ thus yields a construction algorithm with running time that is *linear in* n. Still, some might argue that the complexity of construction in Theorem 1 does depend on a fidelity parameter μ, and this is unsatisfactory. The following

[1] In [3], Arıkan uses the Bhattacharyya parameter $Z(\mathcal{W}_i)$ instead of the probability of error $P_e(\mathcal{W}_i)$. As we shall see shortly, this is of no real importance.

[2] The initial degrading (upgrading) transformation of the binary-input continuous-output AWGN channel to a binary-input channel with a finite output alphabet was done according to the method of Section VI. For that calculation, we used a finer value of $\mu = 2000$. Note that the initial degrading (upgrading) transformation is performed only once.

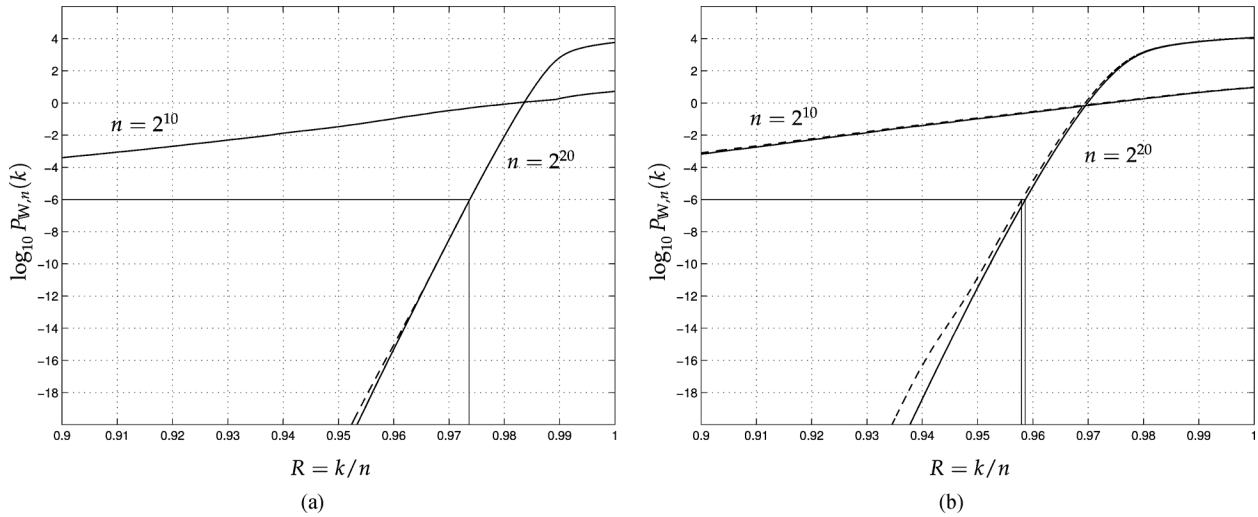


Fig. 2. Upper and lower bounds on $P_{W,n}(k)$ as a function of rate $R = k/n$, for two underlying channels and two code lengths $n = 2^{10}$ and $n = 2^{20}$. The upper bound is dashed while the lower bound is solid. For both channels, the difference between the bounds can only be discerned in the plot corresponding to $n = 2^{20}$. (a) Binary symmetric channel $\mathrm{BSC}(0.001)$. (b) binary-input AWGN channel with $E_s/N_0 = 5.00$ dB.

corollary eliminates this dependence altogether, at the expense of superlinear construction complexity.

Corollary 2: Let W be a binary-input, symmetric, DMC of capacity $I(W)$. Fix arbitrary real constants $\varepsilon > 0$ and $\beta < 1/2$. Then, there is a construction algorithm with running time $O(n \log^2 n \log\log n)$ that for all sufficiently large code lengths n, produces a polar code for W of rate $R \geqslant I(W) - \varepsilon$ such that $P_{\mathrm{block}} \leqslant 2^{-n^{\beta}}$.

Proof: Set $\mu = 2 \lfloor \log_2 n \rfloor$ in Theorem 1 (in fact, we could have used any function of n that grows without bound). ■

We would now like to draw the reader's attention to what Theorem 1 *does not* assert. Namely, given W, ε, and β, the theorem does not tell us how large n must be, only that some values of n are large enough. In fact, given W, ε, β, how large does n need to be in order to guarantee the *existence* of a polar code with $R \geqslant I(W) - \varepsilon$ and $P_{\mathrm{block}} \leqslant 2^{-n^{\beta}}$, let alone the complexity of its construction? This is one of the central questions in the theory of polar codes. Certain lower bounds on this value of n are given in [10]. In the other direction, the exciting recent result of Guruswami and Xia [11, Th. 1] shows that for any fixed W and $\beta \leqslant 0.49$, this value of n grows as a polynomial in $1/\varepsilon$. The work of [11] further shows that, for any fixed W and β, the parameter μ_0 in (3) can be also taken as a polynomial in $1/\varepsilon$.

The rest of this paper is organized as follows. In Section II, we briefly review polar codes and set up the necessary notation. Section III is devoted to channel degrading and upgrading relations, which will be important for us later on. In Section IV, we give a high-level description of our algorithms for approximating polar bit-channels. The missing details in Section IV are then fully specified in Section V. Namely, we show how to reduce the output alphabet of a channel so as to get either a degraded or an upgraded version thereof. In Section VI, we show how to either degrade or upgrade a channel with continuous output into a channel with a finite output alphabet of specified size. In Section VII, we discuss certain improvements to our general algorithms for a specialized case. The accuracy of the (improved) algorithms is then analyzed in Section VIII.

II. Polar Codes

In this section, we briefly review polar codes with the primary aim of setting up the relevant notation. We also indicate where the difficulty of constructing polar codes lies.

Let W be the underlying memoryless channel through which we are to transmit information. If the input alphabet of W is $\mathcal{X}$ and its output alphabet is $\mathcal{Y}$, we write $W : \mathcal{X} \to \mathcal{Y}$. The probability of observing $y \in \mathcal{Y}$ given that $x \in \mathcal{X}$ was transmitted is denoted by $W(y|x)$. We assume throughout that W has binary input and so $\mathcal{X} = \{0, 1\}$. We also assume that W is symmetric. As noted in [3], a binary-input channel W is symmetric if and only if there exists a permutation π of $\mathcal{Y}$ such that $\pi^{-1} = \pi$ (that is, π is an involution) and $W(y|1) = W(\pi(y)|0)$ for all $y \in \mathcal{Y}$ (see [9, p. 94] for an equivalent definition). When the permutation is understood from the context, we abbreviate $\pi(y)$ as $\bar{y}$, and say that $\bar{y}$ and y are *conjugates*. For now, we will further assume that the output alphabet $\mathcal{Y}$ of W is finite. This assumption will be justified in Section VI, where we show how to deal with channels that have continuous output.

Denote the length of the codewords we will be transmitting over W by $n = 2^m$. Given $\boldsymbol{y} = (y_0, y_1, \ldots, y_{n-1}) \in \mathcal{Y}^n$ and $\boldsymbol{u} = (u_0, u_1, \ldots, u_{n-1}) \in \mathcal{X}^n$, let

$$W^n(\boldsymbol{y}|\boldsymbol{u}) \stackrel{\mathrm{def}}{=} \prod_{i=0}^{n-1} W(y_i|u_i).$$

Thus, W^n corresponds to n independent uses of the channel W. A key paradigm introduced in [3] is that of transforming n identical copies (independent uses) of the channel W into n polar bit-channels, through a successive application of Arıkan channel transforms, introduced shortly. For $i = 0, 1, \ldots, n-1$, the ith *bit-channel* $\mathcal{W}_i$ has a binary input alphabet $\mathcal{X}$, an output alphabet $\mathcal{Y}^n \times \mathcal{X}^i$, and transition probabilities defined as follows. Let G be the polarization kernel matrix of [3], given by

$$G = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}.$$

Let $G^{\otimes m}$ be the m-fold Kronecker product of G and let B_n be the $n \times n$ bit-reversal premutation matrix defined in [3, Sec. VII-B]. Denote $\boldsymbol{u}_{i-1} = (u_0, u_1, \ldots, u_{i-1})$. Then

$$\mathcal{W}_i\big(\boldsymbol{y}, \boldsymbol{u}_{i-1}|u_i\big) \stackrel{\text{def}}{=} \frac{1}{2^{n-1}} \sum_{\boldsymbol{v}\in\{0,1\}^{n-1-i}} W^n\Big(\boldsymbol{y}\,|\,(\boldsymbol{u}_{i-1}, u_i, \boldsymbol{v})B_n G^{\otimes m}\Big). \quad (4)$$

Given the bit-channel output $\boldsymbol{y}$ and $\boldsymbol{u}_{i-1}$, the optimal (maximum-likelihood) decision rule for estimating u_i is

$$\widehat{u}_i \;=\; \operatorname{argmax}\big\{\mathcal{W}_i\big(\boldsymbol{y}, \boldsymbol{u}_{i-1}|0\big), \mathcal{W}_i\big(\boldsymbol{y}, \boldsymbol{u}_{i-1}|1\big)\big\}$$

with ties broken arbitrarily. This is the decision rule used in successive cancellation decoding [3]. As before, we let $P_e(\mathcal{W}_i)$ denote the probability that $\widehat{u}_i \neq u_i$ under this rule, assuming that the a priori distribution of u_i is $\text{Bernoulli}(1/2)$.

In essence, constructing a polar code of dimension k is equivalent to finding the k "best" bit-channels. In [3], one is instructed to choose the k bit-channels $\mathcal{W}_i$ with the lowest Bhattacharyya bound $Z(\mathcal{W}_i)$ on the probability of decision error $P_e(\mathcal{W}_i)$. We note that the choice of ranking according to these Bhattacharyya bounds stems from the relative technical ease of manipulating them. A more straightforward criterion would have been to rank directly according to the probability of error $P_e(\mathcal{W}_i)$, and this is the criterion we will follow here.

Since $\mathcal{W}_i$ is well defined through (4), this task is indeed explicit, and thus so is the construction of a polar code. However, note that the output alphabet size of each bit-channel is exponential in n. Thus, a straightforward evaluation of the ranking criterion is intractable for all but the shortest of codes. Our main objective will be to circumvent this difficulty.

As a first step toward achieving our goal, we recall that the bit-channels can be constructed recursively using the Arıkan channel transformations $\mathcal{W} \boxplus \mathcal{W}$ and $\mathcal{W} \circledast \mathcal{W}$, defined as follows. Let $\mathcal{W} : \mathcal{X} \to \mathcal{Y}$ be a binary-input, memoryless, symmetric (BMS) channel. Then, the output alphabet of $\mathcal{W} \boxplus \mathcal{W}$ is $\mathcal{Y}^2$, the output alphabet of $\mathcal{W} \circledast \mathcal{W}$ is $\mathcal{Y}^2 \times \mathcal{X}$, and their transition probabilities are given by

$$\big(\mathcal{W}\boxplus\mathcal{W}\big)(y_1, y_2|u_1) \stackrel{\text{def}}{=} \frac{1}{2} \sum_{u_2\in\mathcal{X}} \mathcal{W}(y_1|u_1 \oplus u_2)\mathcal{W}(y_2|u_2) \quad (5)$$

and

$$\big(\mathcal{W}\circledast\mathcal{W}\big)(y_1, y_2, u_1|u_2) \stackrel{\text{def}}{=} \frac{1}{2}\mathcal{W}(y_1|u_1 \oplus u_2)\mathcal{W}(y_2|u_2). \quad (6)$$

One consequence of this recursive construction is that the explosion in the output alphabet size happens gradually: each transform application roughly squares the alphabet size. We will take advantage of this fact in Section IV.

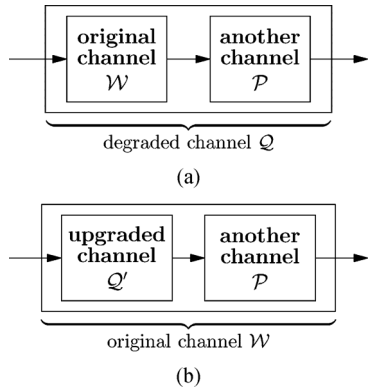


Fig. 3. Degrading and upgrading operations. (a) Degrading $\mathcal{W}$ to $\mathcal{Q}$. (b) Upgrading $\mathcal{W}$ to $\mathcal{Q}'$.

III. Channel Degradation and Upgradation

As outlined in the foregoing two sections, our solution to the exponential growth of the output alphabet of each bit-channel $\mathcal{W}_i$ is to replace $\mathcal{W}_i$ by an approximation thereof. In fact, we will have two approximations, one yielding "better" channels and the other yielding channels that are "worse" than the original one. In this section, we formalize these notions.

We say that a channel $\mathcal{Q} : \mathcal{X} \to \mathcal{Z}$ is (stochastically) *degraded* with respect to a channel $\mathcal{W} : \mathcal{X} \to \mathcal{Y}$ if there exists a channel $\mathcal{P} : \mathcal{Y} \to \mathcal{Z}$ such that

$$\mathcal{Q}(z|x) \;= \sum_{y\in\mathcal{Y}} \mathcal{W}(y|x)\mathcal{P}(z|y) \quad (7)$$

for all $z \in \mathcal{Z}$ and $x \in \mathcal{X}$. For a graphical depiction, see Fig. 3(a). We write $\mathcal{Q} \preccurlyeq \mathcal{W}$ to denote that $\mathcal{Q}$ is degraded with respect to $\mathcal{W}$.

In the interest of brevity and clarity later on, we also define the inverse relation: we will say that a channel $\mathcal{Q}' : \mathcal{X} \to \mathcal{Z}'$ is *upgraded* with respect to $\mathcal{W} : \mathcal{X} \to \mathcal{Y}$ if there exists a channel $\mathcal{P} : \mathcal{Z}' \to \mathcal{Y}$ such that

$$\mathcal{W}(y|x) \;= \sum_{z'\in\mathcal{Z}'} \mathcal{Q}'(z'|x)\mathcal{P}(y|z') \quad (8)$$

for all $z' \in \mathcal{Z}'$ and $x \in \mathcal{X}$ [see Fig. 3(b)]. In other words, the upgraded channel $\mathcal{Q}'$ can be degraded to $\mathcal{W}$. We use $\mathcal{Q}' \succcurlyeq \mathcal{W}$ to denote this relationship. By definition,

$$\mathcal{Q}' \succcurlyeq \mathcal{W} \quad \text{if and only if} \quad \mathcal{W} \preccurlyeq \mathcal{Q}'. \quad (9)$$

Since a channel is both degraded and upgraded with respect to itself (take the intermediate channel as the identity function), we have that both $\preccurlyeq$ and $\succcurlyeq$ relations are reflexive:

$$\mathcal{W} \preccurlyeq \mathcal{W} \quad \text{and} \quad \mathcal{W} \succcurlyeq \mathcal{W}. \quad (10)$$

Also, it can be easily shown that both $\preccurlyeq$ and $\succcurlyeq$ are transitive relations. Thus

$$\text{if} \quad \mathcal{W} \preccurlyeq \mathcal{W}' \quad \text{and} \quad \mathcal{W}' \preccurlyeq \mathcal{W}'', \text{ then} \quad \mathcal{W} \preccurlyeq \mathcal{W}''. \quad (11)$$

If a channel $\mathcal{W}'$ is both degraded and upgraded with respect to $\mathcal{W}$, then we say that $\mathcal{W}$ and $\mathcal{W}'$ are *equivalent*, and denote this by $\mathcal{W} \equiv \mathcal{W}'$. Since $\preccurlyeq$ and $\succcurlyeq$ are transitive relations, it follows

that $\equiv$ is transitive as well. Also, by (9), we have that $\equiv$ is a symmetric relation:

$$\mathcal{W} \equiv \mathcal{W}' \quad \text{if and only if} \quad \mathcal{W}' \equiv \mathcal{W}. \tag{12}$$

Finally, since a channel $\mathcal{W}$ is both upgraded and degraded with respect to itself, we have by (10) that $\equiv$ is a reflexive relation. Thus, channel equivalence is indeed an equivalence relation.

We now set the notation for three channel parameters of interest. Let $\mathcal{W}\colon \mathcal{X} \to \mathcal{Y}$ be a given BMS channel. We denote by $P_e(\mathcal{W})$ the probability of error under maximum-likelihood decision, where ties are broken arbitrarily and the a priori input distribution is $\mathrm{Bernoulli}(1/2)$. This probability is given by

$$P_e(\mathcal{W}) \;=\; \frac{1}{2}\sum_{y\in\mathcal{Y}} \min\big\{\mathcal{W}(y|0), \mathcal{W}(y|1)\big\}. \tag{13}$$

We denote by $Z(\mathcal{W})$ and $I(\mathcal{W})$, respectively, the *Bhattacharyya parameter* and the *capacity* of $\mathcal{W}$. These quantities are given by

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

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

The next lemma asserts that all three quantities behave as expected with respect to the degrading and upgrading relations. This is essentially known—see [20, p. 207], for example. Nevertheless, we include a brief proof, for completeness.

Lemma 3: Let $\mathcal{W}\colon \mathcal{X} \to \mathcal{Y}$ be a BMS channel and suppose that $\mathcal{Q}\colon \mathcal{X} \to \mathcal{Z}$ is degraded with respect to $\mathcal{W}$. Then

$$P_e(\mathcal{Q}) \;\geqslant\; P_e(\mathcal{W}), \tag{14}$$

$$Z(\mathcal{Q}) \;\geqslant\; Z(\mathcal{W}), \tag{15}$$

$$I(\mathcal{Q}) \;\leqslant\; I(\mathcal{W}). \tag{16}$$

Moreover, all of the above continues to hold if we replace "degraded" by "upgraded" and reverse the inequalities. Therefore, if $\mathcal{W} \equiv \mathcal{Q}$, then the inequalities are in fact equalities.

Proof: We consider only the first part, since the fact that all the inequalities will be reversed under upgradation follows immediately from (9). For an elementary proof of (14), combine (13) with the definition of degradation in (7), and note that

$$\begin{aligned}
P_e(\mathcal{Q}) &= \frac{1}{2}\sum_{z\in\mathcal{Z}} \min\Big\{\mathcal{Q}(z|0), \mathcal{Q}(z|1)\Big\} \\
&= \frac{1}{2}\sum_{z\in\mathcal{Z}} \min\left\{\sum_{y\in\mathcal{Y}}\mathcal{W}(y|0)\mathcal{P}(z|y), \sum_{y\in\mathcal{Y}}\mathcal{W}(y|1)\mathcal{P}(z|y)\right\} \\
&\geqslant \frac{1}{2}\sum_{z\in\mathcal{Z}}\sum_{y\in\mathcal{Y}} \min\Big\{\mathcal{W}(y|0)\mathcal{P}(z|y), \mathcal{W}(y|1)\mathcal{P}(z|y)\Big\} \\
&= \frac{1}{2}\sum_{y\in\mathcal{Y}} \min\Big\{\mathcal{W}(y|0)\mathcal{W}(y|1)\Big\}\sum_{z\in\mathcal{Z}}\mathcal{P}(z|y) \\
&= P_e(\mathcal{W}).
\end{aligned}$$

Further, the bound in (15) is concisely proved in [13, Lemma 1.8], while (16) is a simple consequence of the data-processing inequality [8, Th. 2.8.1]. ∎

Given a BMS channel $\mathcal{W}\colon \mathcal{X} \to \mathcal{Y}$, it may be the case that a symbol $y \in \mathcal{Y}$ is its own conjugate: $y = \bar{y}$, and thus, y is an erasure since $\mathcal{W}(y|0) = \mathcal{W}(y|1)$. It would make our proofs simpler if we did not have to deal with this special case. Indeed, we will henceforth assume that y and $\bar{y}$ are always distinct symbols. The next lemma shows that this can be assumed without loss of generality, up to channel equivalence.

Lemma 4: Let $\mathcal{W}\colon \mathcal{X} \to \mathcal{Y}$ be a BMS channel. Then, there exists a BMS channel $\mathcal{W}'\colon \mathcal{X} \to \mathcal{Z}$ such that $\mathcal{W}'$ is equivalent to $\mathcal{W}$, and for all $z \in \mathcal{Z}$, we have that z and $\bar{z}$ are distinct.

Proof: If $\mathcal{W}$ is such that y and $\bar{y}$ are distinct for all $y \in \mathcal{Y}$, then we are done: simply take $\mathcal{W}' = \mathcal{W}$. Otherwise, let $y_? \in \mathcal{Y}$ be a self-conjugate symbol with $\bar{y}_? = y_?$. Define the alphabet $\mathcal{Z}$ of $\mathcal{W}'$ as follows:

$$\mathcal{Z} \;\stackrel{\text{def}}{=}\; \Big(\mathcal{Y} \setminus \{y_?\}\Big) \cup \{z_1, z_2\},$$

where z_1 and z_2 are new symbols, not already in $\mathcal{Y}$. Next, define the transition probabilities of $\mathcal{W}'$ as follows:

$$\mathcal{W}'(z|x) \;=\; \begin{cases} \mathcal{W}(z|x), & \text{if } z \in \mathcal{Y} \\ \frac{1}{2}\mathcal{W}(y_?|x), & \text{if } z \in \{z_1, z_2\} \end{cases}$$

for all $z \in \mathcal{Z}$ and $x \in \mathcal{X}$. We claim that $\mathcal{W}' \succcurlyeq \mathcal{W}$. To see this, take the intermediate channel $\mathcal{P}\colon \mathcal{Z} \to \mathcal{Y}$ as the channel that maps (with probability 1) both z_1 and z_2 to $y_?$, while mapping the other symbols to themselves. Next, we show that $\mathcal{W}' \preccurlyeq \mathcal{W}$. To see this, define the channel $\mathcal{P} : \mathcal{Y} \to \mathcal{Z}$ as follows:

$$\mathcal{P}(z|y) = \begin{cases} 1, & \text{if } z = y \\ \frac{1}{2}, & \text{if } y = y_? \text{ and } z \in \{z_1, z_2\} \\ 0, & \text{otherwise.} \end{cases}$$

To sum up, we have constructed a channel $\mathcal{W}'$ which is equivalent to $\mathcal{W}$, and contains one less self-conjugate symbol. It is also easy to see that $\mathcal{W}'$ is BMS. We can now apply this construction repeatedly until the resulting channel has no self-conjugate symbols. ∎

With Lemma 4 at hand, we henceforth restrict our attention to BMS channels without self-conjugate symbols. Moreover, given a BMS channel $\mathcal{W}\colon \mathcal{X} \to \mathcal{Y}$, we will further assume that for all $y \in \mathcal{Y}$, at least one of the probabilities $\mathcal{W}(y|0) = \mathcal{W}(\bar{y}|1)$ or $\mathcal{W}(y|1) = \mathcal{W}(\bar{y}|0)$ is strictly greater than zero (otherwise, we can delete the output symbols $y, \bar{y}$ since they never occur).

Given a BMS channel $\mathcal{W}\colon \mathcal{X} \to \mathcal{Y}$, we now associate to each output symbol $y \in \mathcal{Y}$ a *likelihood ratio*, defined as follows:

$$\mathrm{LR}_{\mathcal{W}}(y) \;\stackrel{\text{def}}{=}\; \frac{\mathcal{W}(y|0)}{\mathcal{W}(y|1)} \;=\; \frac{\mathcal{W}(y|0)}{\mathcal{W}(\bar{y}|0)}.$$

Note that if $\mathcal{W}(y|1) = 0$, then $\mathcal{W}(y|0) > 0$ by assumption. In this case, we define $\mathrm{LR}_{\mathcal{W}}(y) = \infty$. If the channel $\mathcal{W}$ is understood from the context, we abbreviate $\mathrm{LR}_{\mathcal{W}}(y)$ to $\mathrm{LR}(y)$.

IV. High-Level Description of the Algorithms

In this section, we give a high-level description of our algorithms for approximating a bit channel. We then show how these algorithms can be used in order to construct polar codes.

In order to completely specify the approximating algorithms, one has to supply two *merging functions*: a degrading merge function and an upgrading merge function. In what follows, we define the properties required of the two merging functions, deferring the specification of the functions themselves to the next

section. The next section will also make clear why we have chosen to call these functions "merging."

For the degrading merge function `degrading_merge`, the following must hold. Given a BMS channel $\mathcal{W}$ and positive integer μ, the output of `degrading_merge`$(\mathcal{W}, \mu)$ is a BMS channel $\mathcal{Q}$ such that $\mathcal{Q} \preccurlyeq \mathcal{W}$ and the size of the output alphabet of $\mathcal{Q}$ is at most μ. We define the properties required of the `upgrading_merge` function similarly, with $\preccurlyeq$ replaced by $\succcurlyeq$.

Let $i < n$ be a nonnegative integer with binary representation $i = \langle b_1, b_2, \ldots, b_m \rangle_2$, where b_1 is the most significant bit. Algorithms A and B below contain our procedures for finding a degraded and upgraded approximations, respectively, of the ith bit channel $\mathcal{W}_i$.

Algorithm A: Bit-channel degrading procedure

input: An underlying BMS channel W, a bound $\mu = 2\nu$ on the output alphabet size, a code length $n = 2^m$, and an index i with binary representation $i = \langle b_1, b_2, \ldots, b_m \rangle_2$.

output: A BMS channel that is degraded with respect to the bit channel $\mathcal{W}_i$.

$\mathcal{Q} \leftarrow$ `degrading_merge`(W, μ)
for $j = 1, 2, \ldots, m$ **do**
  if $b_j = 0$ **then**
    $\mathcal{W} \leftarrow \mathcal{Q} \boxplus \mathcal{Q}$
  else
    $\mathcal{W} \leftarrow \mathcal{Q} \circledast \mathcal{Q}$
  $\mathcal{Q} \leftarrow$ `degrading_merge`$(\mathcal{W}, \mu)$
return $\mathcal{Q}$

Algorithm B: Bit-channel upgrading procedure

input: An underlying BMS channel W, a bound $\mu = 2\nu$ on the output alphabet size, a code length $n = 2^m$, and an index i with binary representation $i = \langle b_1, b_2, \ldots, b_m \rangle_2$.

output: A BMS channel that is upgraded with respect to the bit channel $\mathcal{W}_i$.

$\mathcal{Q}' \leftarrow$ `upgrading_merge`(W, μ)
for $j = 1, 2, \ldots, m$ **do**
  if $b_j = 0$ **then**
    $\mathcal{W} \leftarrow \mathcal{Q}' \boxplus \mathcal{Q}'$
  else
    $\mathcal{W} \leftarrow \mathcal{Q}' \circledast \mathcal{Q}'$
  $\mathcal{Q}' \leftarrow$ `upgrading_merge`$(\mathcal{W}, \mu)$
return $\mathcal{Q}'$

In words, we employ the recursive constructions (5) and (6), taking care to reduce the output alphabet size of each intermediate channel to at most μ. Observe that upon applying the channel transformations in either (5) or (6), the output alphabet size grows to either μ^2 or $2\mu^2$, respectively.

The key to proving the correctness of Algorithms A and B is the following lemma. It is essentially a restatement of [13, Lemma 4.7]. For completeness, we include the proof.

Lemma 5: Fix a binary input channel $\mathcal{W} : \mathcal{X} \to \mathcal{Y}$. Denote $\mathcal{W}_{\boxplus} = \mathcal{W} \boxplus \mathcal{W}$ and $\mathcal{W}_{\circledast} = \mathcal{W} \circledast \mathcal{W}$. Suppose that a channel $\mathcal{Q}$ is degraded with respect to $\mathcal{W}$, and denote $\mathcal{Q}_{\boxplus} = \mathcal{Q} \boxplus \mathcal{Q}$ and $\mathcal{Q}_{\circledast} = \mathcal{Q} \circledast \mathcal{Q}$. Then

$$\mathcal{Q}_{\boxplus} \preccurlyeq \mathcal{W}_{\boxplus} \quad \text{and} \quad \mathcal{Q}_{\circledast} \preccurlyeq \mathcal{W}_{\circledast}. \tag{17}$$

Namely, the channel degradation relation is preserved by the Arıkan channel transformations (5) and (6). Moreover, all of the above continues to hold if we replace "degraded" by "upgraded" and $\preccurlyeq$ by $\succcurlyeq$.

Proof: We will prove only the "degraded" part, since it immediately implies the "upgraded" part by interchanging the roles of $\mathcal{W}$ and $\mathcal{Q}$. Let $\mathcal{P} : \mathcal{Y} \to \mathcal{Z}$ be the channel that degrades $\mathcal{W}$ to $\mathcal{Q}$. That is, for all $z \in \mathcal{Z}$ and $x \in \mathcal{X}$, we have

$$\mathcal{Q}(z|x) = \sum_{y \in \mathcal{Y}} \mathcal{W}(y|x)\mathcal{P}(z|y). \tag{18}$$

We first prove that $\mathcal{Q}_{\boxplus} \preccurlyeq \mathcal{W}_{\boxplus}$. Applying transformation (5) to $\mathcal{Q}$, we find that for all $(z_1, z_2) \in \mathcal{Z}^2$ and $u_1 \in \mathcal{X}$, we have

$$\mathcal{Q}_{\boxplus}\big(z_1, z_2|u_1\big) = \frac{1}{2} \sum_{u_2 \in \mathcal{X}} \mathcal{Q}(z_1|u_1 \oplus u_2)\mathcal{Q}(z_2|u_2).$$

Next, we expand $\mathcal{Q}$ twice according to (18), to get

$$\mathcal{Q}_{\boxplus}\big(z_1, z_2|u_1\big) = \frac{1}{2} \sum_{u_2 \in \mathcal{X}} \sum_{(y_1, y_2) \in \mathcal{Y}^2} W(y_1|u_1 \oplus u_2)W(y_2|u_2)\mathcal{P}(z_1|y_1)\mathcal{P}(z_2|y_2).$$

In view of (5), this reduces to

$$\mathcal{Q}_{\boxplus}\big(z_1, z_2|u_1\big) = \sum_{(y_1, y_2) \in \mathcal{Y}^2} \mathcal{W}_{\boxplus}(y_1, y_2|u_1)\mathcal{P}(z_1|y_1)\mathcal{P}(z_2|y_2). \tag{19}$$

Next, define the intermediate channel $\mathcal{P}^* : \mathcal{Y}^2 \to \mathcal{Z}^2$ as follows. For all $(y_1, y_2) \in \mathcal{Y}^2$ and $(z_1, z_2) \in \mathcal{Z}^2$,

$$\mathcal{P}^*\Big((z_1, z_2) \,|\, (y_1, y_2)\Big) \stackrel{\text{def}}{=} \mathcal{P}(z_1|y_1)\mathcal{P}(z_2|y_2).$$

It is easy to see that $\mathcal{P}^*$ is indeed a channel: we get a probability distribution on $\mathcal{Z}^2$ for every fixed $(y_1, y_2) \in \mathcal{Y}^2$. With this, (19) reduces to

$$\mathcal{Q}_{\boxplus}\big(z_1, z_2|u_1\big) = \sum_{(y_1, y_2) \in \mathcal{Y}^2} \mathcal{W}_{\boxplus}(y_1, y_2|u_1)\mathcal{P}^*\Big((z_1, z_2)|(y_1, y_2)\Big),$$

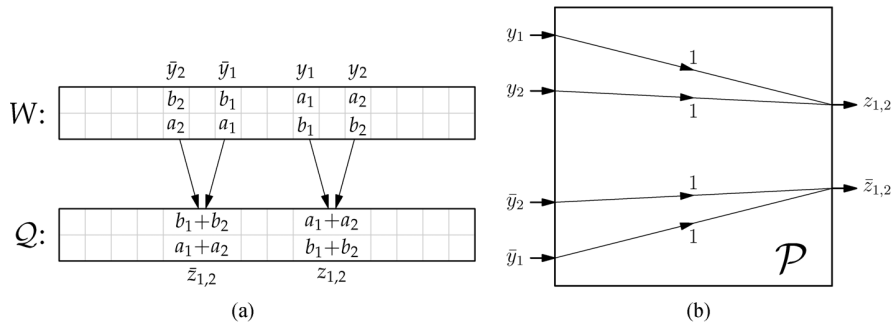
and we get that $\mathcal{Q}_{\boxplus} \preccurlyeq \mathcal{W}_{\boxplus}$, according to the definition (7). The claim $\mathcal{Q}_{\circledast} \preccurlyeq \mathcal{W}_{\circledast}$ can be proved in much the same way. ■

Proposition 6: The output of Algorithm A (respectively, Algorithm B) is a BMS channel that is degraded (respectively, upgraded) with respect to the bit-channel $\mathcal{W}_i$.

Proof: Follows from Lemma 5, by induction on j. ■

Recall that, ideally, a polar code can be constructed as follows. We are given an underlying channel $W : \mathcal{X} \to \mathcal{Y}$, a specified code length $n = 2^m$, and a target block error rate P_{block}. We choose the largest possible subset of bit-channels $\mathcal{W}_i$ such that the sum of their error probabilities $P_e(\mathcal{W}_i)$ does not exceed P_{block}. The resulting polar code is then spanned by the rows of $B_n G^{\otimes m}$ that correspond to the chosen subset of bit-channels. Denote the rate of this code as R_{exact}.

However, since we do not have a computational handle on the bit-channels themselves, we must resort to their approximations. Let $\mathcal{Q}_i$ be the result of running Algorithm A on W and i. Since $\mathcal{Q}_i \preccurlyeq \mathcal{W}_i$, we have that $P_e(\mathcal{Q}_i) \geqslant P_e(\mathcal{W}_i)$ for all i, by Lemma 3. Note that since the output alphabet of $\mathcal{Q}_i$ is small



(a) (b)

Fig. 4. Degrading $\mathcal{W}$ to $\mathcal{Q}$. (a) Degrading merge operation: the entry in the first/second row of a channel is the probability of receiving the corresponding symbol, given that a 0/1 was transmitted. (b) Intermediate channel $\mathcal{P}$.

(at most μ), we can actually compute $P_e(\mathcal{Q}_i)$. We now mimic the "ideal" construction by choosing the largest possible subset of indices i for which the sum of $P_e(\mathcal{Q}_i)$ is at most P_{block}. Observe that, for this subset, the sum of $P_e(\mathcal{W}_i)$ is at most P_{block} as well. Therefore, the code spanned by the corresponding rows of $B_n G^{\otimes m}$ is guaranteed to have block error probability no higher than P_{block}. Denote the rate of this code by $R_\downarrow$. It is easy to see that $R_\downarrow \leqslant R_{\text{exact}}$. In order to gauge the difference between the two rates, we compute a third rate $R_\uparrow$, such that $R_\uparrow \geqslant R_{\text{exact}}$ and consider the difference $R_\uparrow - R_\downarrow$. The rate $R_\uparrow$ is computed in the same way as $R_\downarrow$, but instead of using Algorithm A we use Algorithm B. That is, we choose the largest possible subset of indices i for which the sum of the error probabilities $P_e(\mathcal{Q}'_i)$ of the upgraded channels $\mathcal{Q}'_i$ is at most P_{block}. Recall from Fig. 2 that $R_\downarrow$ and $R_\uparrow$ are typically very close.

We conclude this section by establishing a point that will be needed later in the proof of Theorem 14. Let us consider the running time needed in order to approximate all n bit-channels. Assume that each invocation of either `degrading_merge` or `upgrading_merge` takes time $\tau = \tau(\mu)$. Hence, the time needed to approximate a single bit-channel using either Algorithm A or Algorithm B is $O(m\tau)$. Thus, a naive analysis suggests that the time needed in order to approximate all n bit-channels is $O(nm\tau)$. However, significant savings are obtained by observing that intermediate calculations can be shared between bit-channels. For example, in a naive implementation, we would approximate $W \boxplus W$ over and over again, $n/2$ times instead of only once. A quick calculation shows that the number of *distinct channels* one needs to approximate is only $2n-2$. Indeed, following both branches of the "if" statement of Algorithm A would produce 2^j channels at each level $j = 1, 2, \ldots, m$, for a total of

$$\sum_{j=1}^{m} 2^j \;=\; 2^{m+1} - 2 \;=\; 2n - 2.$$

Thus, the running time needed to approximate all n bit-channels can be reduced to $O\big((2n-2)\tau\big)$, which is simply $O(n\tau)$.

V. Merging Functions

In this section, we specify the degrading and upgrading functions used to reduce the output alphabet size. These functions are called `degrading_merge` and `upgrading_merge` in Algorithms A and B, respectively. For now, let us treat our functions as heuristic (deferring their analysis to Section VIII).

A. Degrading-Merge Function

We first note that the problem of degrading a binary-input channel to a channel with a prescribed output alphabet size was independently considered by Kurkoski and Yagi [15]. The main result in [15] is an optimal degrading strategy, in the sense that the capacity of the resulting channel is the largest possible. In this respect, the method we now introduce is suboptimal. However, as we will show, the complexity of our method is superior to that presented in [15].

The next lemma shows how one can reduce the output alphabet size by 2, and get a degraded channel. It is our first step towards defining a valid `degrading_merge` function.

Lemma 7: Let $\mathcal{W} : \mathcal{X} \to \mathcal{Y}$ be a BMS channel, and let y_1 and y_2 be symbols in the output alphabet $\mathcal{Y}$. Define the channel $\mathcal{Q} : \mathcal{X} \to \mathcal{Z}$ as follows [see Fig. 4(a)]. The output alphabet $\mathcal{Z}$ is given by

$$\mathcal{Z} = \mathcal{Y} \setminus \{y_1, \bar{y}_1, y_2, \bar{y}_2\} \cup \{z_{1,2}, \bar{z}_{1,2}\}.$$

For all $x \in \mathcal{X}$ and $z \in \mathcal{Z}$, define

$$\mathcal{Q}(z|x) = \begin{cases} \mathcal{W}(z|x), & \text{if } z \notin \{z_{1,2}, \bar{z}_{1,2}\}, \\ \mathcal{W}(y_1|x) + \mathcal{W}(y_2|x), & \text{if } z = z_{1,2}, \\ \mathcal{W}(\bar{y}_1|x) + \mathcal{W}(\bar{y}_2|x), & \text{if } z = \bar{z}_{1,2}. \end{cases}$$

Then, $\mathcal{Q} \preccurlyeq \mathcal{W}$. That is, $\mathcal{Q}$ is degraded with respect to $\mathcal{W}$.

Proof: Take the intermediate channel $\mathcal{P} : \mathcal{Y} \to \mathcal{Z}$ as the channel that maps with probability 1 as follows [see Fig. 4(b)]: both y_1 and y_2 map to $z_{1,2}$, both $\bar{y}_1$ and $\bar{y}_2$ map to $\bar{z}_{1,2}$, other symbols map to themselves. Recall that we have assumed that $\mathcal{W}$ does not contain an erasure symbol, and this continues to hold for $\mathcal{Q}$. ∎

We now define the `degrading_merge` function we have used. It gives good results in practice and is amenable to a fast implementation. Assume we are given a BMS channel $\mathcal{W} : \mathcal{X} \to \mathcal{Y}$ with an alphabet size of $2L$ (recall our assumption of no self-conjugates), and wish to transform $\mathcal{W}$ into a degraded version of itself while reducing its alphabet size to μ. If $2L \leqslant \mu$, then we are done, since we can take the degraded version of $\mathcal{W}$ to be $\mathcal{W}$ itself. Otherwise, we do the following. Recall that for each y, we have that $\mathrm{LR}(y) = 1/\mathrm{LR}(\bar{y})$, where in this context $1/0 = \infty$ and $1/\infty = 0$. Thus, our first step is to choose from each pair $(y, \bar{y})$ a representative such that $\mathrm{LR}(y) \geqslant 1$. Next, we order these L representative such that

$$1 \leqslant \mathrm{LR}(y_1) \leqslant \mathrm{LR}(y_2) \leqslant \cdots \leqslant \mathrm{LR}(y_L). \tag{20}$$

We now ask the following: for which index $1 \leqslant i \leqslant L-1$ does the channel resulting from the application of Lemma 7 to $\mathcal{W}$, y_i, and y_{i+1} result in a channel with largest capacity? Note that instead of considering $\binom{L}{2}$ merges, we consider only $L-1$. After finding the maximizing index i we indeed apply Lemma 7 and get a degraded channel $\mathcal{Q}$ with an alphabet size smaller

by 2 than that of $\mathcal{W}$. The same process is applied to $\mathcal{Q}$, until the output alphabet size is not more than μ.

Algorithm C: The `degrading_merge` function

input: A BMS channel $\mathcal{W}: \mathcal{X} \to \mathcal{Y}$ where $|\mathcal{Y}| = 2L$, a bound $\mu = 2\nu$ on the output alphabet size.
output: A degraded channel $\mathcal{Q}: \mathcal{X} \to \mathcal{Y}'$, where $|\mathcal{Y}'| \leqslant \mu$.

```
// Assume 1 ⩽ LR(y_1) ⩽ LR(y_2) ⩽ ··· ⩽ LR(y_L)
for i = 1, 2, ..., L − 1 do
    d ← new data element
    d.a ← W(y_i|0) ,   d.b ← W(ȳ_i|0)
    d.a′ ← W(y_{i+1}|0) ,   d.b′ ← W(ȳ_{i+1}|0)
    d.deltaI ← calcDeltaI(d.a, d.b, d.a′, d.b′)
    insertRightmost(d)
ℓ = L
while ℓ > ν do
    d ← getMin()
    a⁺ = d.a + d.a′ ,   b⁺ = d.b + d.b′
    dLeft ← d.left
    dRight ← d.right
    removeMin()
    ℓ ← ℓ − 1
    if dLeft ≠ null then
        dLeft.a′ = a⁺
        dLeft.b′ = b⁺
        dLeft.deltaI ← calcDeltaI(dLeft.a, dLeft.b, a⁺, b⁺)
        valueUpdated(dLeft)
    if dRight ≠ null then
        dRight.a = a⁺
        dRight.b = b⁺
        dRight.deltaI ←
        calcDeltaI(a⁺, b⁺, dRight.a′, dRight.b′)
        valueUpdated(dRight)
Construct Q according to the probabilities in the data structure and return it.
```

In light of Lemma 7 and (20), a simple yet important point to note is that if y_i and y_{i+1} are merged to z, then

$$\mathrm{LR}(y_i) \leqslant \mathrm{LR}(z) \leqslant \mathrm{LR}(y_{i+1}). \tag{21}$$

Namely, the original LR ordering is essentially preserved by the merging operation. Algorithm C contains an implementation of our merging procedure. It relies on the above observation in order to improve complexity and runs in $O(L \cdot \log L)$ time. Thus, assuming L is at most $2\mu^2$, the running time of our algorithm is $O(\mu^2 \log \mu)$. In contrast, had we used the degrading method presented in [15], the running time would have been $O(\mu^5)$.

Our implementation assumes an underlying data structure and data elements as follows. Our data structure stores data elements, where each data element corresponds to a pair of adjacent letters y_i and y_{i+1}, in the sense of the ordering in (20). Each data element has the following fields:

$$a, \quad b, \quad a', \quad b', \quad \mathrm{deltaI}, \quad \mathsf{dLeft}, \quad \mathsf{dRight}, \quad h.$$

The fields a, b, a', and b' store the probabilities $\mathcal{W}(y_i|0)$, $\mathcal{W}(\bar{y}_i|0)$, $\mathcal{W}(y_{i+1}|0)$, and $\mathcal{W}(\bar{y}_{i+1}|0)$, respectively. The field deltaI contains the difference in capacity that would result from applying Lemma 7 to y_i and y_{i+1}. Note that deltaI is only a function of the above four probabilities, and thus the function `calcDeltaI` used to initialize this field is given by

$$\texttt{calcDeltaI}(a, b, a', b') = C(a, b) + C(a', b') - C(a^+, b^+),$$

Before the merge of y_i and y_{i+1}:

$$\ldots \leftrightarrow \overbrace{(y_{i-1}, y_i)}^{\mathsf{dLeft}} \leftrightarrow \underbrace{(y_i, y_{i+1})}_{\text{merged to } z} \leftrightarrow \overbrace{(y_{i+1}, y_{i+2})}^{\mathsf{dRight}} \leftrightarrow \ldots$$

After the merge, a new symbol z:

$$\ldots \leftrightarrow (y_{i-1}, z) \leftrightarrow (z, y_{i+2}) \leftrightarrow \ldots$$

Fig. 5. Graphical depiction of the doubly linked list before and after a merge.

where

$$C(a, b) = -(a + b) \log_2((a + b)/2) + a \log_2(a) + b \log_2(b),$$

we use the shorthand

$$a^+ = a + a', \quad b^+ = b + b',$$

and $0 \log_2 0$ is defined as 0. The field dLeft is a pointer to the data element corresponding to the pair y_{i-1} and y_i (or "null", if $i = 1$). Likewise, dRight is a pointer to the element corresponding to the pair y_{i+1} and y_{i+2} (see Fig. 5 for a graphical depiction). Apart from these, each data element contains an integer field h, which will be discussed shortly.

We now discuss the functions that are the interface to our data structure: `insertRightmost`, `getMin`, `removeMin`, and `valueUpdated`. Our data structure combines the attributes of a doubly linked list [7, Sec. 10.2] and a heap [3][7, Ch. 6]. The doubly linked list is implemented through the dLeft and dRight fields of each data element, as well as a pointer to the rightmost element of the list. Our heap will have the "array" implementation, as described in [7, Sec. 6.1]. Thus, each data element will have a corresponding index in the heap array, and this index is stored in the field h. The doubly linked list will be ordered according to the corresponding LR value, while the heap will be sorted according to the deltaI field.

The function `insertRightmost` inserts a data element as the rightmost element of the list and updates the heap accordingly. The function `getMin` returns the data element with smallest deltaI. Namely, the data element corresponding to the pair of symbols we are about to merge. The function `removeMin` removes the element returned by `getMin` from both the linked-list and the heap. The function `valueUpdated` updates the heap due to a change in deltaI resulting from a merge, but does not change the linked list in view of (21).

The running time of `getMin` is $O(1)$, and this is obviously also the case for `calcDeltaI`. Due to the need of updating the heap, the running time of `removeMin`, `valueUpdated`, and `insertRightmost` is $O(\log L)$. The time needed for the initial sort of the LR pairs is $O(L \cdot \log L)$. Hence, since the initializing for-loop in Algorithm C has L iterations and the while-loop has $L - \nu$ iterations, the total running time of Algorithm C is $O(L \cdot \log L)$.

Note that at first sight, it may seem as though there might be an even better heuristic to employ. As before, assume that the y_i are ordered according to their likelihood ratios, and all of these are at least 1. Instead of limiting the application of Lemma 7 to y_i and y_{i+1}, we can broaden our search and consider the penalty

[3]In short, a heap is a data structure that supports four operations: "insert", "getMin," "removeMin," and "valueUpdated." In our implementation, the running time of "getMin" is constant, while the running time of the other operations is logarithmic in the heap size.

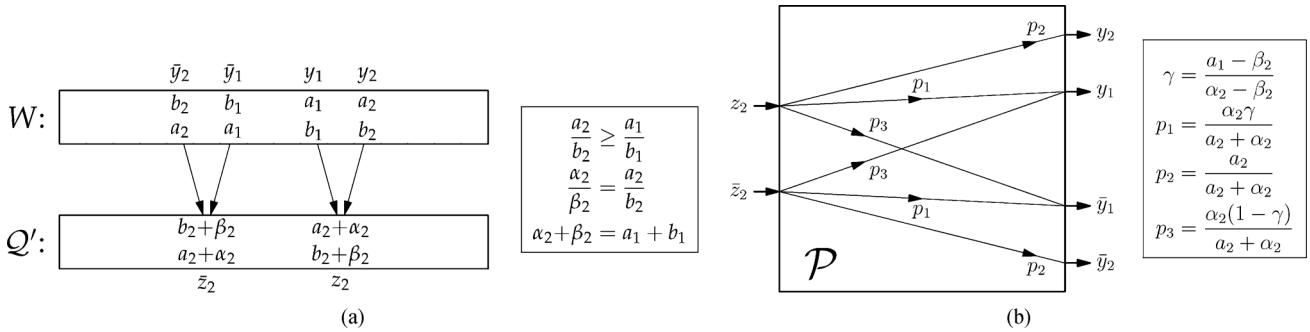


Fig. 6. First method of Upgrading $\mathcal{W}$ to $\mathcal{Q}'$. (a) Upgrading merge operation. (b) Intermediate channel $\mathcal{P}$.

in capacity incurred by merging arbitrary y_i and y_j, where $i \neq j$. Indeed, we could further consider merging arbitrary y_i and $\bar{y}_j$, where $i \neq j$. Clearly, this broader search will result in worse complexity. However, as the next theorem shows, we will essentially gain nothing by it.

Theorem 8: Let $\mathcal{W} : \mathcal{X} \to \mathcal{Y}$ be a BMS channel, with

$$\mathcal{Y} = \{y_1, y_2, \ldots, y_L, \bar{y}_1, \bar{y}_2, \ldots, \bar{y}_L\}.$$

Assume that

$$1 \leqslant \mathrm{LR}(y_1) \leqslant \mathrm{LR}(y_2) \leqslant \cdots \leqslant \mathrm{LR}(y_L).$$

For symbols $w_1, w_2 \in \mathcal{Y}$, denote by $I(w_1, w_2)$ the capacity of the channel one gets by the application of Lemma 7 to w_1 and w_2. Then, for all distinct $1 \leqslant i \leqslant L$ and $1 \leqslant j \leqslant L$,

$$I(\bar{y}_i, \bar{y}_j) = I(y_i, y_j) \geqslant I(y_i, \bar{y}_j) = I(\bar{y}_i, y_j). \tag{22}$$

Moreover, for all $1 \leqslant i < j < k \leqslant L$, we have that either

$$I(y_i, y_j) \geqslant I(y_i, y_k),$$

or

$$I(y_j, y_k) \geqslant I(y_i, y_k).$$

We note that Theorem 8 seems very much related to [15, Lemma 5]. However, one important difference is that Theorem 8 deals with the case in which the degraded channel is constrained to be symmetric, while [15, Lemma 5] does not. At any rate, for completeness, we will prove Theorem 8 in Appendix A.

B. Upgrading-Merge Functions

The fact that one can merge symbol pairs and get a degraded version of the original channel should come as no surprise. However, it turns out that we can also merge symbol pairs and get an *upgraded* version of the original channel. We first show a simple method of doing this. Later on, we will show a slightly more complex method, and compare between the two.

As in the degrading case, we show how to reduce the output alphabet size by 2, and then apply this method repeatedly as much as needed. The following lemma shows how the core reduction can be carried out. The intuition behind it is simple. Namely, now we "promote" a pair of output symbols to have a higher LR value and then merge with an existing pair having that LR.

Lemma 9: Let $\mathcal{W} : \mathcal{X} \to \mathcal{Y}$ be a BMS channel, and let y_2 and y_1 be symbols in the output alphabet $\mathcal{Y}$. Denote $\lambda_2 = \mathrm{LR}(y_2)$ and $\lambda_1 = \mathrm{LR}(y_1)$. Assume that

$$1 \leqslant \lambda_1 \leqslant \lambda_2. \tag{23}$$

Next, let $a_1 = \mathcal{W}(y_1|0)$ and $b_1 = \mathcal{W}(\bar{y}_1|0)$. Define α_2 and β_2 as follows. If $\lambda_2 < \infty$

$$\alpha_2 = \lambda_2 \frac{a_1 + b_1}{\lambda_2 + 1}, \qquad \beta_2 = \frac{a_1 + b_1}{\lambda_2 + 1}. \tag{24}$$

Otherwise, we have $\lambda_2 = \infty$, and so define

$$\alpha_2 = a_1 + b_1, \qquad \beta_2 = 0. \tag{25}$$

We note that the subscript "2" in α_2 and β_2 is meant to suggest a connection to λ_2, since $\alpha_2/\beta_2 = \lambda_2$.

For real numbers α, β, and $x \in \mathcal{X}$, define

$$t(\alpha, \beta|x) = \begin{cases} \alpha, & \text{if } x = 0, \\ \beta, & \text{if } x = 1. \end{cases}$$

Define the channel $\mathcal{Q}' : \mathcal{X} \to \mathcal{Z}'$ as follows [see Fig. 6(a)]. The output alphabet $\mathcal{Z}'$ is given by

$$\mathcal{Z}' = \mathcal{Y} \setminus \{y_2, \bar{y}_2, y_1, \bar{y}_1\} \cup \{z_2, \bar{z}_2\}.$$

For all $x \in \mathcal{X}$ and $z \in \mathcal{Z}'$,

$$\mathcal{Q}'(z|x) = \begin{cases} \mathcal{W}(z|x), & \text{if } z \notin \{z_2, \bar{z}_2\}, \\ \mathcal{W}(y_2|x) + t(\alpha_2, \beta_2|x), & \text{if } z = z_2, \\ \mathcal{W}(\bar{y}_2|x) + t(\beta_2, \alpha_2|x), & \text{if } z = \bar{z}_2. \end{cases}$$

Then, $\mathcal{Q}' \succcurlyeq \mathcal{W}$. That is, $\mathcal{Q}'$ is upgraded with respect to $\mathcal{W}$.

Proof: Denote $a_2 = \mathcal{W}(y_2|0)$ and $b_2 = \mathcal{W}(\bar{y}_2|0)$. First, note that

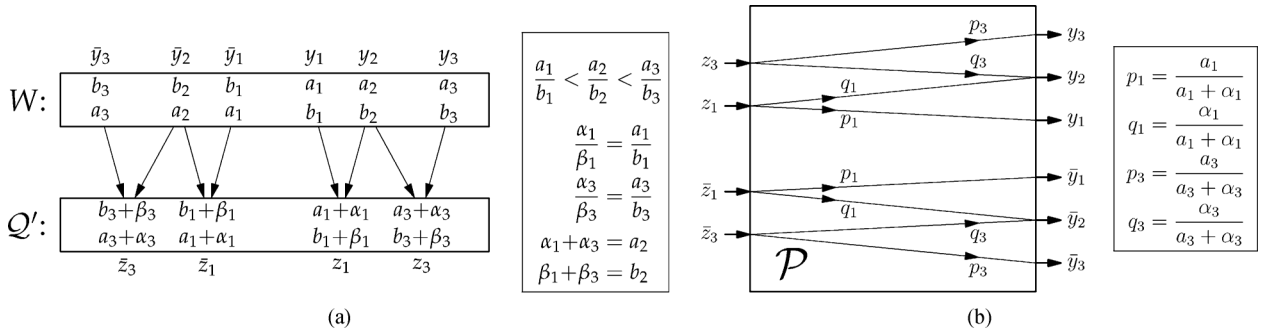
$$a_1 + b_1 = \alpha_2 + \beta_2.$$

Next, let γ be defined as follows. If $\lambda_2 > 1$, let

$$\gamma = \frac{a_1 - \beta_2}{\alpha_2 - \beta_2} = \frac{b_1 - \alpha_2}{\beta_2 - \alpha_2},$$

and note that (23) implies that $0 \leqslant \gamma \leqslant 1$. Otherwise ($\lambda_1 = \lambda_2 = 1$), let

$$\gamma = 1.$$



(a) (b)

Fig. 7. Second method of Upgrading $\mathcal{W}$ to $\mathcal{Q}'$. (a) Upgrading merge operation. (b) Intermediate channel $\mathcal{P}$.

Define the intermediate channel $\mathcal{P} : \mathcal{Z}' \to \mathcal{Y}$ as follows.

$$\mathcal{P}(y|z) = \begin{cases} 1, & \text{if } z \notin \{z_2, \bar{z}_2\} \text{ and } y = z, \\ \frac{\alpha_2\gamma}{a_2+\alpha_2}, & \text{if } (z,y) \in \{(z_2, y_1), (\bar{z}_2, \bar{y}_1)\}, \\ \frac{a_2}{a_2+\alpha_2}, & \text{if } (z,y) \in \{(z_2, y_2), (\bar{z}_2, \bar{y}_2)\}, \\ \frac{\alpha_2(1-\gamma)}{a_2+\alpha_2}, & \text{if } (z,y) \in \{(z_2, \bar{y}_1), (\bar{z}_2, y_1)\}, \\ 0, & \text{otherwise.} \end{cases}$$

Notice that when $\lambda_2 < \infty$, we have that

$$\frac{a_2}{a_2+\alpha_2} = \frac{b_2}{b_2+\beta_2} \quad \text{and} \quad \frac{\alpha_2}{a_2+\alpha_2} = \frac{\beta_2}{b_2+\beta_2}.$$

Some simple calculations finish the proof. ∎

The following corollary shows that we do not "lose anything" when applying Lemma 9 to symbols y_1 and y_2 such that $\mathrm{LR}(y_1) = \mathrm{LR}(y_2)$. Thus, intuitively, we do not expect to lose much when applying Lemma 9 to symbols with "close" LR values.

Corollary 10: Let $\mathcal{W}$, $\mathcal{Q}'$, y_1, and y_2 be as in Lemma 9. If $\mathrm{LR}(y_1) = \mathrm{LR}(y_2)$, then $\mathcal{Q}' \equiv \mathcal{W}$. That is, $\mathcal{W}$ and $\mathcal{Q}'$ are equivalent. Moreover, all of the above holds if we replace "Lemma 9" by "Lemma 7".

Proof: The proof follows by noticing that the channel $\mathcal{Q}'$ we get by applying Lemma 9 to $\mathcal{W}$, y_1, and y_2, is exactly the same channel we get if we apply Lemma 7 instead. Thus, we have both $\mathcal{Q}' \succcurlyeq \mathcal{W}$ and $\mathcal{Q}' \preccurlyeq \mathcal{W}$. ∎

In Lemma 9, we have essentially transferred the probability $\mathcal{W}(y_1|0) + \mathcal{W}(\bar{y}_1|0)$ onto a symbol pair with a higher LR value. We now show a different method of merging that involves dividing the probability $\mathcal{W}(y_1|0) + \mathcal{W}(\bar{y}_1|0)$ between a symbol pair with higher LR value and a symbol pair with lower LR value. As we will prove later on, this new approach is generally preferable.

Lemma 11: Let $\mathcal{W} : \mathcal{X} \to \mathcal{Y}$ be a BMS channel, and let y_1, y_2, and y_3 be symbols in the output alphabet $\mathcal{Y}$. Denote $\lambda_1 = \mathrm{LR}(y_1)$, $\lambda_2 = \mathrm{LR}(y_2)$, and $\lambda_3 = \mathrm{LR}(y_3)$. Assume that

$$1 \leqslant \lambda_1 < \lambda_2 < \lambda_3.$$

Next, let $a_2 = \mathcal{W}(y_2|0)$ and $b_2 = \mathcal{W}(\bar{y}_2|0)$. Define α_1, β_1, α_3, β_3 as follows. If $\lambda_3 < \infty$

$$\alpha_1 = \lambda_1 \frac{\lambda_3 b_2 - a_2}{\lambda_3 - \lambda_1}, \qquad \beta_1 = \frac{\lambda_3 b_2 - a_2}{\lambda_3 - \lambda_1}, \tag{26}$$

$$\alpha_3 = \lambda_3 \frac{a_2 - \lambda_1 b_2}{\lambda_3 - \lambda_1}, \qquad \beta_3 = \frac{a_2 - \lambda_1 b_2}{\lambda_3 - \lambda_1}. \tag{27}$$

Otherwise, we have $\lambda_3 = \infty$, and so define

$$\alpha_1 = \lambda_1 b_2, \qquad \beta_1 = b_2, \tag{28}$$

$$\alpha_3 = a_2 - \lambda_1 b_2, \qquad \beta_3 = 0. \tag{29}$$

Let $t(\alpha, \beta|x)$ be as in Lemma 9, and define the BMS channel $\mathcal{Q}' : \mathcal{X} \to \mathcal{Z}'$ as follows [see Fig. 7(a)]. The output alphabet $\mathcal{Z}'$ is given by

$$\mathcal{Z}' = \mathcal{Y} \setminus \{y_1, \bar{y}_1, y_2, \bar{y}_2, y_3, \bar{y}_3\} \cup \{z_1, \bar{z}_1, z_3, \bar{z}_3\}.$$

For all $x \in \mathcal{X}$ and $z \in \mathcal{Z}'$, define

$$\mathcal{Q}'(z|x) = \begin{cases} \mathcal{W}(z|x), & \text{if } z \notin \{z_1, \bar{z}_1, z_3, \bar{z}_3\}, \\ \mathcal{W}(y_1|x) + t(\alpha_1, \beta_1|x), & \text{if } z = z_1, \\ \mathcal{W}(\bar{y}_1|x) + t(\beta_1, \alpha_1|x), & \text{if } z = \bar{z}_1, \\ \mathcal{W}(y_3|x) + t(\alpha_3, \beta_3|x), & \text{if } z = z_3, \\ \mathcal{W}(\bar{y}_3|x) + t(\beta_3, \alpha_3|x), & \text{if } z = \bar{z}_3. \end{cases}$$

Then, $\mathcal{Q}' \succcurlyeq \mathcal{W}$. That is, $\mathcal{Q}'$ is upgraded with respect to $\mathcal{W}$.

Proof: Denote $a_1 = \mathcal{W}(y_1|0)$, $b_1 = \mathcal{W}(\bar{y}_1|0)$, $a_3 = \mathcal{W}(y_3|0)$, and $b_3 = \mathcal{W}(\bar{y}_3|0)$. Define the intermediate channel $\mathcal{P} : \mathcal{Z}' \to \mathcal{Y}$ as follows:

$$\mathcal{P}(y|z) = \begin{cases} 1, & \text{if } z \notin \{z_3, \bar{z}_3, z_1, \bar{z}_1\} \text{ and } y = z, \\ \frac{a_1}{a_1+\alpha_1} = \frac{b_1}{b_1+\beta_1}, & \text{if } (z,y) \in \{(z_1, y_1), (\bar{z}_1, \bar{y}_1)\}, \\ \frac{\alpha_1}{a_1+\alpha_1} = \frac{\beta_1}{b_1+\beta_1}, & \text{if } (z,y) \in \{(z_1, y_2), (\bar{z}_1, \bar{y}_2)\}, \\ \frac{a_3}{a_3+\alpha_3}, & \text{if } (z,y) \in \{(z_3, y_3), (\bar{z}_3, \bar{y}_3)\}, \\ \frac{\alpha_3}{a_3+\alpha_3}, & \text{if } (z,y) \in \{(z_3, y_2), (\bar{z}_3, \bar{y}_2)\}, \\ 0, & \text{otherwise.} \end{cases}$$

Notice that when $\lambda_3 < \infty$, we have that

$$\frac{a_3}{a_3+\alpha_3} = \frac{b_3}{b_3+\beta_3} \quad \text{and} \quad \frac{\alpha_3}{a_3+\alpha_3} = \frac{\beta_3}{b_3+\beta_3}.$$

The proof follows by observing that, whatever the value of λ_3,

$$\alpha_1 + \alpha_3 = a_2 \quad \text{and} \quad \beta_1 + \beta_3 = b_2.$$

∎

The following lemma formalizes why Lemma 11 results in a merging operation that is better than that of Lemma 9.

Lemma 12: Let $\mathcal{W}$, y_1, y_2, and y_3 be as in Lemma 11. Denote by $\mathcal{Q}'_{123} : \mathcal{X} \to \mathcal{Z}'_{123}$ the result of applying Lemma 11 to $\mathcal{W}$, y_1, y_2, and y_3. Next, denote by $\mathcal{Q}'_{23} : \mathcal{X} \to \mathcal{Z}'_{23}$ the result of applying Lemma 9 to $\mathcal{W}$, y_2, and y_3. Then $\mathcal{Q}'_{23} \succcurlyeq \mathcal{Q}'_{123} \succcurlyeq \mathcal{W}$. Namely, in a sense, $\mathcal{Q}'_{123}$ is a more faithful representation of $\mathcal{W}$ than $\mathcal{Q}'_{23}$ is.

Proof: Recall that the two alphabets $\mathcal{Z}'_{123}$ and $\mathcal{Z}'_{23}$ satisfy

$$\mathcal{Z}'_{123} = \{z_1, \bar{z}_1, z_3, \bar{z}_3\} \cup \mathcal{A},$$
$$\mathcal{Z}'_{23} = \{y_1, \bar{y}_1, z_3, \bar{z}_3\} \cup \mathcal{A},$$

where

$$\mathcal{A} = \mathcal{Y} \setminus \{y_1, \bar{y}_1, y_2, \bar{y}_2, y_3, \bar{y}_3\}$$

is the set of symbols not participating in either merge operation.

In order to prove that $\mathcal{Q}'_{123}$ is degraded with respect to $\mathcal{Q}'_{23}$, we must supply a corresponding intermediate channel $\mathcal{P} : \mathcal{Z}'_{23} \to \mathcal{Z}'_{123}$. To this end, let

$$\lambda_3 = \frac{\mathcal{Q}'_{123}(z_3|0)}{\mathcal{Q}'_{123}(z_3|1)} = \frac{\mathcal{Q}'_{23}(z_3|0)}{\mathcal{Q}'_{23}(z_3|1)} = \frac{\mathcal{W}(y_3|0)}{\mathcal{W}(y_3|1)}$$

and

$$\gamma = \frac{\mathcal{Q}'_{123}(z_3|0)}{\mathcal{Q}'_{23}(z_3|0)} = \frac{\mathcal{Q}'_{123}(\bar{z}_3|1)}{\mathcal{Q}'_{23}(\bar{z}_3|1)}.$$

Note that in both Lemma 11 and 9 we have that $\alpha_3/\beta_3 = \lambda_3$. Next, we recall that in Lemma 9, we have that $\alpha_3+\beta_3 = a_2+b_2$, whereas in Lemma 11, we have $\alpha_3 + \beta_3 = a_2 + b_2 - \alpha_1 - \beta_1$. Thus, we conclude that $0 \leqslant \gamma \leqslant 1$. Moreover, since $0 \leqslant \gamma \leqslant 1$, we conclude that the following definition of an intermediate channel is indeed valid

$$\mathcal{P}(z_{123}|z_{23}) = \begin{cases} 1, & \text{if } z_{123} = z_{23} \text{ and } z_{123} \in \mathcal{A}, \\ 1, & \text{if } (z_{23}, z_{123}) \in \{(y_1, z_1), (\bar{y}_1, \bar{z}_1)\}, \\ \gamma, & \text{if } (z_{23}, z_{123}) \in \{(z_3, z_3), (\bar{z}_3, \bar{z}_3)\}, \\ \frac{(1-\gamma)\lambda_1}{\lambda_1+1}, & \text{if } (z_{23}, z_{123}) \in \{(z_3, z_1), (\bar{z}_3, \bar{z}_1)\}, \\ \frac{(1-\gamma)}{\lambda_1+1}, & \text{if } (z_{23}, z_{123}) \in \{(z_3, \bar{z}_1), (\bar{z}_3, z_1)\}, \\ 0, & \text{otherwise.} \end{cases}$$

A short calculation shows that $\mathcal{P}$ is indeed an intermediate channel that degrades $\mathcal{Q}'_{23}$ to $\mathcal{Q}'_{123}$. ∎

At this point, the reader may be wondering why we have chosen to state Lemma 9 at all. Namely, it is clear what disadvantages it has with respect to Lemma 11, but we have yet to indicate any advantages. Recalling the conditions of Lemma 11, we see that it cannot be employed when the set $\{\lambda_1, \lambda_2, \lambda_3\}$ contains nonunique elements. In fact, more is true. Ultimately, when one wants to implement the algorithms outlined in this paper, one will most probably use floating point numbers. Recall that a major source of numerical instability stems from subtracting two floating point numbers that are too close. By considering the denominator in (26) and (27) we see that λ_1 and λ_3 should not be too close. Moreover, by considering the numerators, we conclude that λ_2 should not be too close to both λ_1 and λ_3. So, when these cases do occur, our only option is Lemma 9.

We now define the merge-upgrading procedure we have used. Apart from an initial step, it is very similar to the merge-degrading procedure we have previously outlined. Assume we are given a BMS channel $\mathcal{W} : \mathcal{X} \to \mathcal{Y}$ with an alphabet size of $2L$ and wish to reduce its alphabet size to μ, while transforming $\mathcal{W}$ into a upgraded version of itself. If $2L \leqslant \mu$, then, as before, we are done. Otherwise, as in the merge-degrading procedure, we choose L representatives $y_1, y_2, \ldots, y_L$, and order them according to their LR values, all of which are greater than or equal to 1. We now specify the preliminary step: for some specified parameter epsilon (we have used $\epsilon = 10^{-3}$), we check if there exists an index $1 \leqslant i < L$ such that the ratio $\mathrm{LR}(y_{i+1})/\mathrm{LR}(y_i)$ is less than $1+\epsilon$. If so, we apply Lemma 9 repeatedly, until no such index exists. Now comes the main step. We ask the following question: for which index $1 \leqslant i \leqslant L-1$ does the channel resulting from the application of Lemma 11 to $\mathcal{W}$, y_i, y_{i+1}, and y_{i+2} result in a channel with smallest capacity increase? After finding the minimizing index i, we indeed apply Lemma 11 and get an upgraded channel $\mathcal{Q}'$ with an alphabet size smaller by 2 than that of $\mathcal{W}$. The same process is applied to $\mathcal{Q}'$, until the output alphabet size is not more than μ. As before, assuming the output alphabet size of $\mathcal{W}$ is at most $2\mu^2$, an implementation similar to that given in Algorithm C will run in $O(\mu^2 \log \mu)$ time.

As was the case for degrading, the following theorem proves that no generality is lost by only considering merging of consecutive triplets of the form y_i, y_{i+1}, y_{i+2} in the main step. The proof is given in Appendix B.

Theorem 13: Let $\mathcal{W} : \mathcal{X} \to \mathcal{Y}$ be a BMS channel. Denote by $I_{\mathcal{W}}$ the capacity of $\mathcal{W}$ and by $I(y_1, y_2, y_3)$ the capacity one gets by the application of Lemma 11 to $\mathcal{W}$ and symbols $y_1, y_2, y_3 \in \mathcal{Y}$ such that

$$1 \leqslant \mathrm{LR}(y_1) \leqslant \mathrm{LR}(y_2) \leqslant LR(y_3).$$

Let $\mathrm{LR}(y_1) = \lambda_1$, $\mathrm{LR}(y_2) = \lambda_2$, $\mathrm{LR}(y_3) = \lambda_3$, $\pi_2 = \mathcal{W}(y_2|0) + \mathcal{W}(y_2|1)$, and denote the difference in capacities as

$$\Delta[\lambda_1; \lambda_2, \pi_2; \lambda_3] = I(y_1, y_2, y_3) - I_{\mathcal{W}}.$$

Then, for all $\lambda'_1 \leqslant \lambda_1$ and $\lambda'_3 \geqslant \lambda_3$,

$$\Delta[\lambda_1; \lambda_2, \pi_2; \lambda_3] \leqslant \Delta[\lambda'_1; \lambda_2, \pi_2; \lambda'_3]. \tag{30}$$

We end this section by considering the running time of Algorithms A and B.

Theorem 14: Let an underlying BMS channel W, a fidelity parameter μ, and codelength $n = 2^m$ be given. Assume that the output alphabet size of the underlying channel W is at most μ. The running time of either Algorithm A or Algorithm B is as follows. Approximating a single bit-channel takes $O(m \cdot \mu^2 \log \mu)$ time; approximating all n bit-channels takes $O(n \cdot \mu^2 \log \mu)$ time.

Proof: Without loss of generality, we consider Algorithm A. Recall that the output alphabet size of W is at most μ. Thus,

by induction, at the start of each loop the size of the output alphabet of $\mathcal{Q}$ is at most μ. Therefore, at each iteration, calculating $\mathcal{W}$ from $\mathcal{Q}$ takes $O(\mu^2)$ time, since the output alphabet size of $\mathcal{W}$ is at most $2\mu^2$. Next, we have seen that each invocation of `degrading_merge` takes $O(\mu^2 \log \mu)$ time. The number of times we loop in Algorithm A is m. Thus, for a single bit-channel, the total running time is $O(m \cdot \mu^2 \log \mu)$.

As was explained at the end of Section IV, when approximating all n bit channels, the number of distinct channels that need to be approximated is $2n - 2$. Thus, the total running time in this case is $O(n \cdot \mu^2 \log \mu)$. ■

VI. Channels With Continuous Output

Recall that in order to apply either Algorithm A or B to an underlying BMS channel W, we had to thus far assume that W has a finite output alphabet. In this section, we show two transforms (one degrading and the other upgrading) that transform a BMS channel with a continuous alphabet to a BMS channel with a specified finite output alphabet size. Thus, after applying the degrading (upgrading) transform we will shortly specify to W, we will be in a position to apply Algorithm A (B) and get a degraded (upgraded) approximation of $\mathcal{W}_i$. Moreover, we prove that both degraded and upgraded versions of our original channels have a guaranteed closeness to the original channel, in terms of difference of capacity.

Let W be a given BMS channel with a continuous alphabet. We will make a few assumptions on W. First, we assume that the output alphabet of W is the reals $\mathbb{R}$. Thus, for $y \in \mathbb{R}$, let $f(y|0)$ and $f(y|1)$ be the p.d.f. functions of the output given that the input was 0 and 1, respectively. Next, we require that the symmetry of W manifest itself as

$$f(y|0) = f(-y|1), \quad \text{for all } y \in \mathbb{R}.$$

Also, for notational convenience, we require that

$$f(y|0) \geqslant f(y|1), \quad \text{for all } y \geqslant 0. \tag{31}$$

Note that all of the above holds for the BAWGN channel (after renaming the input 0 as -1).

We now introduce some notation. For $y \geqslant 0$, define the likelihood ratio of y as

$$\lambda(y) = \frac{f(y|0)}{f(y|1)}. \tag{32}$$

As usual, if $f(y|1)$ is zero while $f(y|0)$ is not, we define $\lambda(y) = \infty$. Also, if both $f(y|0)$ and $f(y|1)$ are zero, then we arbitrarily define $\lambda(y) = 1$. Note that by (31), we have that $\lambda(y) \geqslant 1$.

Under these definitions, a short calculation shows that the capacity of W is

$$I(W) = \int_0^\infty \left(f(y|0) + f(y|1)\right) C[\lambda(y)]\, dy,$$

where for $1 \leqslant \lambda < \infty$

$$C[\lambda] = 1 - \frac{\lambda}{\lambda+1} \log_2\left(1 + \frac{1}{\lambda}\right) - \frac{1}{\lambda+1} \log_2\left(\lambda + 1\right),$$

and (for continuity) we define $C[\infty] = 1$.

Let $\mu = 2\nu$ be the specified size of the degraded/upgraded channel output alphabet. An important property of $C[\lambda]$ is that it is strictly increasing in λ for $\lambda \geqslant 1$. This property is easily proved, and will now be used to show that the following sets form a partition of the nonnegative reals. For $1 \leqslant i \leqslant \nu - 1$, let

$$A_i = \left\{ y \geqslant 0 : \frac{i-1}{\nu} \leqslant C[\lambda(y)] < \frac{i}{\nu} \right\}. \tag{33}$$

For $i = \nu$ we similarly define (changing the second inequality to a weak inequality)

$$A_\nu = \left\{ y \geqslant 0 : \frac{\nu-1}{\nu} \leqslant C[\lambda(y)] \leqslant 1 \right\}. \tag{34}$$

As we will see later on, we must assume that the sets A_i are sufficiently "nice." This will indeed be the case of for BAWGN channel.

A. Degrading Transform

Essentially, our degrading procedure will consist of ν applications of the continuous analog of Lemma 7. Denote by $\mathcal{Q} : \mathcal{X} \to \mathcal{Z}$ the degraded approximation of W we are going to produce, where

$$\mathcal{Z} = \{z_1, \bar{z}_1, z_2, \bar{z}_2, \ldots, z_\nu, \bar{z}_\nu\}.$$

We define $\mathcal{Q}$ as follows:

$$\mathcal{Q}(z_i|0) = \mathcal{Q}(\bar{z}_i|1) = \int_{A_i} f(y|0)\, dy, \tag{35}$$

$$\mathcal{Q}(\bar{z}_i|0) = \mathcal{Q}(z_i|1) = \int_{A_i} f(-y|0)\, dy. \tag{36}$$

Lemma 15: The channel $\mathcal{Q} : \mathcal{X} \to \mathcal{Z}$ is a BMS channel such that $\mathcal{Q} \preccurlyeq W$.

Proof: It is readily seen that $\mathcal{Q}$ is a BMS channel. To prove $\mathcal{Q} \preccurlyeq W$, we now supply intermediate channel $\mathcal{P} : \mathbb{R} \to \mathcal{Z}$.

$$\mathcal{P}(z|y) = \begin{cases} 1, & \text{if } z = z_i \text{ and } y \in A_i, \\ 1, & \text{if } z = \bar{z}_i \text{ and } -y \in A_i, \\ 0, & \text{otherwise.} \end{cases}$$

■

The following lemma bounds the loss in capacity incurred by the degrading operation.

Lemma 16: The difference in capacities of $\mathcal{Q}$ and W can be bounded as follows:

$$0 \leqslant I(W) - I(\mathcal{Q}) \leqslant \frac{1}{\nu} = \frac{2}{\mu}. \tag{37}$$

Proof: The first inequality in (37) is a consequence of the degrading relation and (16). We now turn our attention to the second inequality.

Recall that since the A_i partition the nonnegative reals, the capacity of W equals

$$I(W) = \sum_{i=1}^{\nu} \int_{A_i} \left(f(y|0) + f(y|1)\right) C[\lambda(y)] dy. \tag{38}$$

As for $\mathcal{Q}$, we start by defining for $1 \leqslant i \leqslant \nu$ the ratio

$$\theta_i = \frac{\mathcal{Q}(z_i|0)}{\mathcal{Q}(z_i|1)},$$

where the cases of the numerator and/or denominator equaling zero are as in the definition of $\lambda(y)$. By this definition, similarly to the continuous case, the capacity of $\mathcal{Q}$ is equal to

$$I(\mathcal{Q}) = \sum_{i=1}^{\nu} \left(\mathcal{Q}(z_i|0) + \mathcal{Q}(z_i|1)\right) C[\theta_i]. \tag{39}$$

Recall that by the definition of A_i in (33) and (34), we have that for all $y \in A_i$,

$$\frac{i-1}{\nu} \leqslant C[\lambda(y)] \leqslant \frac{i}{\nu}.$$

Thus, by the definition of $\mathcal{Q}(z_i|0)$ and $\mathcal{Q}(z_i|1)$ in (35) and (36), respectively, we must have by the strict monotonicity of C that

$$\frac{i-1}{\nu} \leqslant C[\theta_i] \leqslant \frac{i}{\nu}, \quad \text{if } \mathcal{Q}(z_i|0) > 0.$$

Thus, for all $y \in A_i$,

$$|C[\theta_i] - C[\lambda(y)]| \leqslant \frac{1}{\nu}, \quad \text{if } \mathcal{Q}(z_i|0) > 0.$$

Next, note that $\mathcal{Q}(z_i|0) > 0$ implies $\mathcal{Q}(z_i|0) + \mathcal{Q}(z_i|1) > 0$. Thus, we may bound $I(\mathcal{Q})$ as follows:

$$I(\mathcal{Q}) = \sum_{i=1}^{\nu} \left(\mathcal{Q}(z_i|0) + \mathcal{Q}(z_i|1)\right) C[\theta_i] =$$
$$\sum_{i=1}^{\nu} \int_{A_i} \left(f(y|0) + f(y|1)\right) C[\theta_i] dy \geqslant$$
$$\sum_{i=1}^{\nu} \int_{A_i} \left(f(y|0) + f(y|1)\right) \left(C[\lambda(y)] - \frac{1}{\nu}\right) dy =$$
$$\left(\sum_{i=1}^{\nu} \int_{A_i} \left(f(y|0) + f(y|1)\right) C[\lambda(y)] dy\right) - \frac{1}{\nu} = I(W) - \frac{1}{\nu},$$

which proves the second inequality. ■

B. Upgrading Transform

In parallel with the degrading case, our upgrading procedure will essentially consist of ν applications of the continuous analog of Lemma 9. Denote by $\mathcal{Q}' : \mathcal{X} \to \mathcal{Z}'$ the upgraded approximation of W we are going to produce, where

$$\mathcal{Z}' = \{z_1, \bar{z}_1, z_2, \bar{z}_2, \ldots, z_\nu, \bar{z}_\nu\}.$$

As before, we will show that the loss in capacity due to the upgrading operation is at most $1/\nu$.

Let us now redefine the ratio θ_i. Recalling that the function $C[\lambda]$ is strictly increasing in $\lambda \geqslant 1$, we deduce that it has an inverse in that range. Thus, for $1 \leqslant i \leqslant \nu$, we define $\theta_i \geqslant 1$ as follows:

$$\theta_i = C^{-1}\left[\frac{i}{\nu}\right]. \tag{40}$$

Note that for $i = \nu$, we have that $\theta_\nu = \infty$. Also, note that for $y \in A_i$ we have by (33) and (34) that

$$1 \leqslant \lambda(y) \leqslant \theta_i. \tag{41}$$

We now define $\mathcal{Q}'$. For $1 \leqslant i \leqslant \nu$, let,

$$\pi_i = \int_{A_i} \left(f(\alpha|0) + f(-\alpha|0)\right) d\alpha. \tag{42}$$

Then,

$$\mathcal{Q}'(z|0) = \begin{cases} \frac{\theta_i \pi_i}{\theta_i + 1}, & \text{if } z = z_i \text{ and } \theta_i \neq \infty, \\ \frac{\pi_i}{\theta_i + 1}, & \text{if } z = \bar{z}_i \text{ and } \theta_i \neq \infty, \\ \pi_i, & \text{if } z = z_i \text{ and } \theta_i = \infty, \\ 0, & \text{if } z = \bar{z}_i \text{ and } \theta_i = \infty, \end{cases} \tag{43}$$

and

$$\mathcal{Q}'(z_i|1) = \mathcal{Q}'(\bar{z}_i|0), \quad \mathcal{Q}'(\bar{z}_i|1) = \mathcal{Q}'(z_i|0). \tag{44}$$

Lemma 17: The channel $\mathcal{Q}' : \mathcal{X} \to \mathcal{Z}'$ is a BMS channel such that $\mathcal{Q}' \succcurlyeq W$.

Proof: As before, the proof that $\mathcal{Q}'$ is a BMS channel is easy. To show that $\mathcal{Q}' \succcurlyeq W$, we must supply the intermediate channel $\mathcal{P}$. The proof follows easily if we define $\mathcal{P} : \mathcal{Z}' \to \mathbb{R}$ as the cascade of two channels, $\mathcal{P}_1 : \mathcal{Z}' \to \mathbb{R}$ and $\mathcal{P}_2 : \mathbb{R} \to \mathbb{R}$.

The channel $\mathcal{P}_1 : \mathcal{Z}' \to \mathbb{R}$ is essentially a renaming channel. Denote by $g(\alpha|z)$ the p.d.f. of the output of $\mathcal{P}_1$ given that the input was z. Then, for $1 \leqslant i \leqslant \nu$,

$$g(\alpha|z) = \begin{cases} \frac{f(\alpha|0)+f(-\alpha|0)}{\pi_i}, & \text{if } z = z_i \text{ and } \alpha \in A_i, \\ \frac{f(\alpha|0)+f(-\alpha|0)}{\pi_i}, & \text{if } z = \bar{z}_i \text{ and } -\alpha \in A_i, \\ 0, & \text{otherwise.} \end{cases} \tag{45}$$

Note that by (42), the function $g(\alpha|z)$ is indeed a p.d.f. for every fixed value of $z \in \mathcal{Z}'$.

Next, we turn to $\mathcal{P}_2 : \mathbb{R} \to \mathbb{R}$, the LR reducing channel. Let $\alpha \in A_i$ and recall the definition of $\lambda(y)$ given in (32). Define the quantity p_α as follows:

$$p_\alpha = \begin{cases} \frac{\theta_i - \lambda(\alpha)}{(\lambda(\alpha)+1)(\theta_i - 1)}, & \text{if } 1 < \theta_i < \infty, \\ \frac{1}{2}, & \text{if } \theta_i = 1, \\ \frac{1}{\lambda(\alpha)+1}, & \text{if } \theta_i = \infty \text{ and } \lambda(\alpha) < \infty, \\ 0, & \text{if } \lambda(\alpha) = \infty. \end{cases} \tag{46}$$

By (41) with α in place of y we deduce that $0 \leqslant p_\alpha \leqslant 1/2$. We define the channel $\mathcal{P}_2 : \mathbb{R} \to \mathbb{R}$ as follows. For $y \geqslant 0$,

$$\mathcal{P}_2(y|\alpha) = \begin{cases} 1 - p_\alpha, & \text{if } y = \alpha, \\ p_\alpha, & \text{if } y = -\alpha, \end{cases} \tag{47}$$

and

$$\mathcal{P}_2(-y| - \alpha) = \mathcal{P}_2(y|\alpha).$$

Consider the random variable Y, which is defined as the output of the concatenation of channels $\mathcal{Q}'$, $\mathcal{P}_1$, and $\mathcal{P}_2$, given that the

input to $\mathcal{Q}'$ was 0. We must show that the p.d.f. of Y is $f(y|0)$. To do this, we consider the limit

$$\lim_{\epsilon\to 0}\frac{\text{Prob}(y\leqslant Y\leqslant y+\epsilon)}{\epsilon}.$$

Consider first a y such that $y\in A_i$, and assume further that ϵ is small enough so that the whole interval between y and $y+\epsilon$ is in A_i. In this case, the above can be expanded to

$$\lim_{\epsilon\to 0}\frac{1}{\epsilon}\Big[\mathcal{Q}'(z_i|0)\cdot\int_y^{y+\epsilon} g(\alpha|z_i)(1-p_\alpha)\,d\alpha + \mathcal{Q}'(\bar{z}_i|0)\cdot\int_{-y-\epsilon}^{-y} g(\alpha|\bar{z}_i)p_{-\alpha}\,d\alpha\Big].$$

Assuming that the two integrands are indeed integrable, this reduces to

$$\mathcal{Q}'(z_i|0)\cdot g(y|z_i)(1-p_y)+\mathcal{Q}'(\bar{z}_i|0)\cdot g(-y|\bar{z}_i)p_y.$$

From here, simple calculations indeed reduce the above to $f(y|0)$. The other cases are similar. ∎

As in the degrading case, we can bound the loss in capacity incurred by the upgrading operation.

Lemma 18: The difference in capacities of $\mathcal{Q}'$ and W can be bounded as follows:

$$0\leqslant I(\mathcal{Q}')-I(W)\leqslant\frac{1}{\nu}=\frac{2}{\mu}. \tag{48}$$

Proof: The first inequality in (48) is a consequence of the upgrading relation and (16). We now turn our attention to the second inequality.

For all $y\in A_i$, by (33), (34), and (40), we have that

$$C[\theta_i]-C[\lambda(y)]\leqslant\frac{1}{\nu},\quad\text{if } \mathcal{Q}'(z_i|0)>0.$$

Next, notice that by (43),

$$\theta_i=\frac{\mathcal{Q}'(z_i|0)}{\mathcal{Q}'(z_i|1)},\quad\text{if } \mathcal{Q}'(z_i|0)>0.$$

As in the degrading case, we have that $\mathcal{Q}'(z_i|0)>0$ implies $\mathcal{Q}'(z_i|0)+\mathcal{Q}'(z_i|1)>0$. Thus, we may bound $I(\mathcal{Q}')$ as follows:

$$I(\mathcal{Q}')=\sum_{i=1}^{\nu}\left(\mathcal{Q}'(z_i|0)+\mathcal{Q}'(z_i|1)\right)C[\theta_i]=\sum_{i=1}^{\nu}\int_{A_i}\left(f(y|0)+f(y|1)\right)C[\theta_i]dy\leqslant\sum_{i=1}^{\nu}\int_{A_i}\left(f(y|0)+f(y|1)\right)\left(C[\lambda(y)]+\frac{1}{\nu}\right)dy=\left(\sum_{i=1}^{\nu}\int_{A_i}\left(f(y|0)+f(y|1)\right)C[\lambda(y)]dy\right)+\frac{1}{\nu}=I(W)+\frac{1}{\nu},$$

which proves the second inequality. ∎

VII. Variations of Our Algorithms

As one might expect, Algorithms A and B can be tweaked and modified. As an example, we now show an improvement to Algorithm A for a specific case. As we will see in Section VIII, this improvement is key to proving Theorem 1. Also, it turns out that Algorithm A is compatible with the result by Guruswami and Xia [11], in the following sense: if we were to use Algorithm A with the same n and μ dictated by [11], then we would be guaranteed a resulting code with parameters as least as good as those promised by [11].

Recall our description of how to construct a polar code given at the end of Section IV: obtain a degraded approximation of each bit channel through the use of Algorithm A, and then select the k best channels when ordered according to the upper bound on the probability of error. Note that Algorithm A returns a channel, but in this case only one attribute of that channel interests us, namely, the probability of error. In this section, we show how to specialize Algorithm A accordingly and benefit.

The specialized algorithm is given as Algorithm D. We note that the plots in this paper having to do with an upper bound on the probability of error were produced by running this algorithm. The key observation follows from in [3, eqs. (26) and (27)], which we now restate. Recall that $Z(\mathcal{W})$ is the Bhattacharyya parameter of the channel $\mathcal{W}$. Then,

$$Z(\mathcal{W}\boxplus\mathcal{W})\leqslant 2Z(\mathcal{W})-Z(\mathcal{W})^2 \tag{49}$$

$$Z(\mathcal{W}\circledast\mathcal{W})=Z(\mathcal{W})^2. \tag{50}$$

Algorithm D: An upper bound on the error probability

input: An underlying BMS channel W, a bound $\mu=2\nu$ on the output alphabet size, a code length $n=2^m$, an index i with binary representation $i=\langle b_1,b_2,\ldots,b_m\rangle_2$.
output: An upper bound on $P_e(\mathcal{W}_i)$.

$\mathsf{Z}\leftarrow Z(W)$
$\mathcal{Q}\leftarrow$ `degrading_merge`(W,μ)
for $j=1,2,\ldots,m$ **do**
  if $b_j=0$ **then**
    $\mathcal{W}\leftarrow\mathcal{Q}\boxplus\mathcal{Q}$
    $\mathsf{Z}\leftarrow\min\{Z(\mathcal{W}),2\mathsf{Z}-\mathsf{Z}^2\}$
  else
    $\mathcal{W}\leftarrow\mathcal{Q}\circledast\mathcal{Q}$
    $\mathsf{Z}\leftarrow\mathsf{Z}^2$
  $\mathcal{Q}\leftarrow$ `degrading_merge`$(\mathcal{W},\mu)$
return $\min\{P_e(\mathcal{Q}),\mathsf{Z}\}$

Theorem 19: Let a code length $n=2^m$, an index $0\leqslant i<n$, an underlying channel W, and a fidelity parameter $\mu=2\nu$ be given. Denote by $\hat{p}_A$ and $\hat{p}_D$ the outputs of Algorithms A and D, respectively. Then,

$$\hat{p}_A\geqslant\hat{p}_D\geqslant P_e(\mathcal{W}_i).$$

That is, the bound produced by Algorithm D is always as least as good as that produced by Algorithm A.

Proof: Denote by $\mathcal{W}^{(j)}$ the channel we are trying to approximate during iteration j. That is, we start with $\mathcal{W}^{(0)}=W$. Then, iteratively $\mathcal{W}^{(j+1)}$ is gotten by transforming $\mathcal{W}^{(j)}$ using

TABLE I
UPPER AND LOWER BOUNDS ON $P_{W,n}(k)$ FOR $W = \mathrm{BSC}(0.11)$, CODE LENGTH $n = 2^{20}$, AND RATE $k/n = 445340/2^{20} = 0.42471$

	Algorithm A	**Algorithm D**	**Algorithm B**
$\mu = 8$	5.096030e-03	1.139075e-04	1.601266e-11
$\mu = 16$	6.926762e-05	2.695836e-05	4.296030e-08
$\mu = 64$	1.808362e-06	1.801289e-06	7.362648e-07
$\mu = 128$	1.142843e-06	1.142151e-06	8.943154e-07
$\mu = 256$	1.023423e-06	1.023423e-06	9.382042e-07
$\mu = 512$		9.999497e-07	9.417541e-07

either $\boxplus$ or $\circledast$, according to the value of b_j. Ultimately, we have $\mathcal{W}^{(m)}$, which is simply the bit-channel $\mathcal{W}_i$.

The heart of the proof is to show that after iteration j has completed (just after line 10 has executed), the variable Z is such that

$$Z(\mathcal{W}^{(j)}) \leqslant \mathsf{Z} \leqslant 1.$$

The proof is by induction. For the basis, note that before the first iteration starts (just after line 2 has executed), we have $\mathsf{Z} = Z(\mathcal{W}^{(0)})$. For the induction step, first note that $2\mathsf{Z} - \mathsf{Z}^2$ is both an increasing function of Z and is between 0 and 1, when $0 \leqslant \mathsf{Z} \leqslant 1$. Obviously, this is also true for Z^2. Now, note that at the end of iteration j we have that the variable $\mathcal{W}$ is degraded with respect to $\mathcal{W}^{(j)}$. Recalling (15), (49), and (50), the induction step is proved. ∎

We end this section by referring to Table I. In the table, we fix the underlying channel, the code length, and the code rate. Then, we compare upper and lower bounds on $P_{W,n}(k)$, for various values of μ. For a given μ, the lower bound is obtained by running Algorithm B, while the two upper bounds are obtained by running Algorithms A and D. As can be seen, the upper bound supplied by Algorithm D is always superior.

VIII. ANALYSIS

As we have seen in previous sections, we can build polar codes by employing Algorithm D and gauge how far we are from the optimal construction by running Algorithm B. As can be seen in Fig. 2, our construction turns out to be essentially optimal, for moderate sizes of μ. However, we are still to prove Theorem 1, which gives analytic justification to our method of construction. We do so in this section.

As background to Theorem 1, recall from [5] that for a polar code of length $n = 2^m$, the fraction of bit channels with probability of error less than 2^{-n^β} tends to the capacity of the underlying channel as n goes to infinity, for $\beta < 1/2$. Moreover, the constraint $\beta < 1/2$ is tight in that the fraction of such channels is strictly less than the capacity, for $\beta > 1/2$. Thus, in this context, the restriction on β imposed by Theorem 1 cannot be eased.

In order to prove Theorem 1, we make use of the results of Pedarsani *et al.* [19], in particular [19, Th. 1] given below. We also point out that many ideas used in the proof of Theorem 1 appear—in one form or another—in [19, Th. 2] and its proof.

Theorem 20 (Restatement of [19, Th. 1]): Let an underlying BMS channel W be given. Let $n = 2^m$ be the code length, and denote by $\mathcal{W}_i^{(m)}$ the corresponding ith bit channel, where $0 \leqslant i < n$. Next, denote by $\mathcal{Q}_i^{(m)}(\nu)$ the degraded approximation of $\mathcal{W}_i^{(m)}$ returned by running Algorithm A with parameters W, $\mu = 2\nu$, i, and m. Then,

$$\frac{\left|\left\{i : I(\mathcal{W}_i^{(m)}) - I(\mathcal{Q}_i^{(m)}(\nu)) \geqslant \sqrt{\tfrac{m}{\nu}}\right\}\right|}{n} \leqslant \sqrt{\frac{m}{\nu}}.$$

With respect to the above, we remark the following. Recall that in Section VI-A, we introduced a method of degrading a continuous channel to a discrete one with at most $\mu = 2\nu$ symbols. In fact, there is nothing special about the continuous case: a slight modification can be used to degrade an arbitrary discrete channel to a discrete channel with at most μ symbols. Thus, we have an alternative to the merge-degrading method introduced in Section V-A. Thus, it follows easily that Theorem 20 and thus Theorem 1 would still hold had we used that alternative.

We now break the proof of Theorem 1 into several lemmas. Put simply, the first lemma states that a laxer requirement than that in Theorem 1 on the probability of error can be met.

Lemma 21: Let $\mathcal{Q}_i^{(m)}(\nu)$ be as in Theorem 20. Then, for every $\delta > 0$ and $\epsilon > 0$, there exists an m_0 and a large enough $\mu = 2\nu$ such that

$$\frac{\left|\left\{i_0 : Z\left(\mathcal{Q}_{i_0}^{(m_0)}(\nu)\right) \leqslant \delta\right\}\right|}{n_0} \geqslant I(W) - \epsilon, \qquad (51)$$

where

$$n_0 = 2^{m_0} \quad \text{and} \quad 0 \leqslant i_0 < n_0.$$

We first note that Lemma 21 has a trivial proof: By [3, Th. 2], we know that there exists an m_0 for which (51) holds, if $\mathcal{Q}_{i_0}^{(m_0)}(\nu)$ is replaced by $\mathcal{W}_{i_0}^{(m_0)}$. Thus, we may take μ large enough so that the pair-merging operation defined in Lemma 7 is never executed, and so $\mathcal{Q}_{i_0}^{(m_0)}(\nu)$ is in fact equal to $\mathcal{W}_{i_0}^{(m_0)}$.

This proof—although valid—implies a value of μ which is doubly exponential in m_0. We now give an alternative proof, which—as we have recently learned—is a precursor to the result of Guruswami and Xia [11]. Namely, we state this alternative proof since we have previously conjectured and now know by [11] that it implies a value of m_0 which is not too large.

Proof of Lemma 21: For simplicity of notation, let us drop the subscript 0 from i_0, n_0, and m_0. Recall that by [3, Th. 1], we have that the capacity of bit channels polarizes. Specifically, for each $\epsilon_1 > 0$ and $\delta_1 > 0$, there exists an m such that

$$\frac{\left|\left\{i : I\left(\mathcal{W}_i^{(m)}\right) \geqslant 1 - \delta_1\right\}\right|}{n} \geqslant I(W) - \epsilon_1. \qquad (52)$$

We can now combine the above with Theorem 20 and deduce that

$$\frac{\left|\left\{i : I\left(\mathcal{Q}_i^{(m)}(\nu)\right) \geqslant 1 - \delta_1 - \sqrt{\tfrac{m}{\nu}}\right\}\right|}{n} \geqslant I(W) - \epsilon_1 - \sqrt{\frac{m}{\nu}}. \qquad (53)$$

Next, we claim that for each $\delta_2 > 0$ and $\epsilon_2 > 0$, there exist m and $\mu = 2\nu$ such that

$$\frac{\left|\left\{i : I\left(\mathcal{Q}_i^{(m)}(\nu)\right) \geqslant 1-\delta_2\right\}\right|}{n} \geqslant I(W) - \epsilon_2. \tag{54}$$

To see this, take $\epsilon_1 = \epsilon_2/2$, $\delta_1 = \delta_2/2$, and let m be the guaranteed constant such that (52) holds. Now, we can take ν big enough so that, in the context of (53), we have that both

$$\delta_1 + \sqrt{\frac{m}{\nu}} < \delta_2$$

and

$$\epsilon_1 + \sqrt{\frac{m}{\nu}} < \epsilon_2.$$

By [3, eq. (2)], we have that

$$Z\left(\mathcal{Q}_i^{(m)}(\nu)\right) \leqslant \sqrt{1 - I^2\left(\mathcal{Q}_i^{(m)}(\nu)\right)}.$$

Thus, if (54) holds, then

$$\frac{\left|\left\{i : Z\left(\mathcal{Q}_i^{(m)}(\nu)\right) \leqslant \sqrt{2\delta_2 - \delta_2^2}\right\}\right|}{n} \geqslant I(W) - \epsilon_2.$$

So, as before, we deduce that for every $\delta_3 > 0$ and $\epsilon_3 > 0$, there exist m and μ such that

$$\frac{\left|\left\{i : Z\left(\mathcal{Q}_i^{(m)}(\nu)\right) \leqslant \delta_3\right\}\right|}{n} \geqslant I(W) - \epsilon_3.$$

■

The next lemma will be used later to bound the evolution of the variable Z in Algorithm D.

Lemma 22: For every $m \geqslant 0$ and index $0 \leqslant i < 2^m$, let there be a corresponding real $0 \leqslant \zeta(i,m) \leqslant 1$. Denote the binary representation of i by $i = \langle b_1, b_2, \ldots, b_m\rangle_2$. Assume that the $\zeta(i,m)$ satisfy the following recursive relation. For $m > 0$ and $i' = \langle b_1, b_2, \ldots, b_{m-1}\rangle_2$,

$$\zeta(i,m) \leqslant \begin{cases} 2\zeta(i', m-1) - \zeta^2(i', m-1), & \text{if } b_m = 0, \\ \zeta^2(i', m-1), & \text{otherwise}. \end{cases} \tag{55}$$

Then, for every $\beta < 1/2$, we have that

$$\liminf_{m\to\infty} \frac{\left|\left\{i : \zeta(i,m) < 2^{-n^\beta}\right\}\right|}{n} \geqslant 1 - \zeta(0,0), \tag{56}$$

where $n = 2^m$.

Proof: First, note that both $f_1(\zeta) = \zeta^2$ and $f_2(\zeta) = 2\zeta - \zeta^2$ strictly increase from 0 to 1 when ζ ranges from 0 to 1. Thus, it suffices to prove the claim for the worst case in which the inequality in (55) is replaced by an equality. Assume from now on that this is indeed the case.

Consider an underlying BEC with probability of erasure (as well as Bhattacharyya parameter) $\zeta(0,0)$. Next, note that the ith bit channel, for $0 \leqslant i < n = 2^m$, is also a BEC, with probability of erasure $\zeta(i,m)$. Since the capacity of the underlying BEC is $1 - \zeta(0,0)$, we deduce (56) by [5, Th. 2]. ■

We are now in a position to prove Theorem 1.

Proof of Theorem 1: Let us first specify explicitly the code construction algorithm used, and then analyze it. As expected, we simply run Algorithm D with parameters W and n to produce upper bounds on the probability of error of all n bit channels. Then, we sort the upper bounds in ascending order. Finally, we produce a generator matrix G, with k rows. The rows of G correspond to the first k bit channels according to the sorted order, and k is the largest integer such that the sum of upper bounds is strictly less than $2^{n-\beta}$. By Theorem 14, the total running time is indeed $O(n \cdot \mu^2 \log \mu)$.

Recall our definition of $\mathcal{W}_i^{(m)}$ and $\mathcal{Q}_i^{(m)}$ from Theorem 20. Denote the upper bound on the probability of error returned by Algorithm D for bit channel i by $\overline{P_e}(\mathcal{W}_i^{(m)}, \mu)$. The theorem will follow easily once we prove that for all $\epsilon > 0$ and $0 < \beta < 1/2$, there exists an even μ_0 such that for all $\mu = 2\nu \geqslant \mu_0$, we have

$$\liminf_{m\to\infty} \frac{\left|\left\{i : \overline{P_e}(\mathcal{W}_i^{(m)}, \mu) < 2^{-n^\beta}\right\}\right|}{n} \geqslant I(W) - \epsilon. \tag{57}$$

By Lemma 21, there exist constants m_0 and ν such that

$$\frac{\left|\left\{i_0 : Z\left(\mathcal{Q}_{i_0}^{(m_0)}(\nu)\right) \leqslant \frac{\epsilon}{2}\right\}\right|}{n_0} \geqslant I(W) - \frac{\epsilon}{2}, \tag{58}$$

where

$$n_0 = 2^{m_0} \quad \text{and} \quad 0 \leqslant i_0 < n_0.$$

Denote the code length as $n = 2^m$, where $m = m_0 + m_1$ and $m_1 > 0$. Consider an index $0 \leqslant i < n$ having binary representation

$$i = \langle b_1, b_2, \ldots, b_{m_0}, b_{m_0+1}, \ldots, b_m\rangle_2,$$

where b_1 is the most significant bit. We split the run of Algorithm D on i into two stages. The first stage will have j going from 1 to m_0, while the second stage will have j going from $m_0 + 1$ to m.

We start by considering the end of the first stage. Namely, we are at iteration $j = m_0$ and line 10 has just finished executing. Recall that we denote the value of the variable $\mathcal{Q}$ after the line has executed by $\mathcal{Q}_{i_0}^{(m_0)}(\nu)$, where

$$i_0 = \langle b_1, b_2, \ldots, b_{m_0}\rangle_2.$$

Similarly, define $\mathsf{Z}_{i_0}^{(m_0)}(\nu)$ as the value of the variable Z at that point. Since, by (15), degrading increases the Bhattacharyya parameter, we have then that the Bhattacharyya parameter of the variable $\mathcal{W}$ is less than or equal to that of the variable $\mathcal{Q}$. So, by the minimization carried out in either line 6 or 9, we conclude the following: at the end of line 10 of the algorithm, when $j = m_0$,

$$\mathsf{Z} = \mathsf{Z}_{i_0}^{(m_0)}(\nu) \leqslant Z\left(\mathcal{Q}_{i_0}^{(m_0)}(\nu)\right) = Z(\mathcal{Q}).$$

We can combine this observation with (58) to conclude that

$$\frac{\left|\left\{i_0 : \mathsf{Z}_{i_0}^{(m_0)}(\nu) \leqslant \frac{\epsilon}{2}\right\}\right|}{n_0} \geqslant I(W) - \frac{\epsilon}{2}. \tag{59}$$

We now move on to consider the second stage of the algorithm. Fix an index $i = \langle b_1, b_2, \ldots, b_{m_0}, b_{m_0+1}, \ldots, b_m\rangle_2$. That is, let i have i_0 as a binary prefix of length m_0. Denote by $\mathsf{Z}[t]$ the value of Z at the end of line 10, when $j = m_0 + t$. By lines 6 and 9 of the algorithm we have, similarly to (55), that

$$\mathsf{Z}[t+1] \leqslant \begin{cases} 2\mathsf{Z}[t] - \mathsf{Z}^2[t], & \text{if } b_{m_0+t+1} = 0, \\ \mathsf{Z}^2[t], & \text{otherwise.} \end{cases}$$

We now combine our observations about the two stages. Let γ be a constant such that

$$\beta < \gamma < \frac{1}{2}.$$

Considering (59), we see that out of the $n_0 = 2^{m_0}$ possible prefixes of length m_0, the fraction for which

$$\mathsf{Z}[0] \leqslant \frac{\epsilon}{2} \tag{60}$$

is at least $I(W) - \frac{\epsilon}{2}$. Next, by Lemma 22, we see that for each such prefix, the fraction of suffixes for which

$$\mathsf{Z}[m_1] \leqslant 2^{-(n_1)^\gamma} \tag{61}$$

is at least $1 - \mathsf{Z}[0]$, as $n_1 = 2^{m_1}$ tends to infinity. Thus, for each such prefix, we get by (60) that (in the limit) the fraction of such suffixes is at least $1 - \frac{\epsilon}{2}$. We can now put all our bounds together and claim that as m_1 tends to infinity, the fraction of indices $0 \leqslant i < 2^m$ for which (61) holds is at least

$$\left(I(W) - \frac{\epsilon}{2}\right) \cdot \left(1 - \frac{\epsilon}{2}\right) \geqslant I(W) - \epsilon.$$

By line (11) of Algorithm D, we see that $\mathsf{Z}[m_1]$ is an upper bound on the return value $\overline{P_e}(\mathcal{W}_i^{(m)})$. Thus, we conclude that

$$\liminf_{m\to\infty} \frac{\left|\left\{i : \overline{P_e}(\mathcal{W}_i^{(m)}, \mu) < 2^{-(n_1)^\gamma}\right\}\right|}{n} = I(W) - \epsilon.$$

With the above at hand, the only thing left to do in order to prove (57) is to show that for m_1 large enough, we have that

$$2^{-(n_1)^\gamma} \leqslant 2^{-n^\beta},$$

which reduces to showing that

$$(n_1)^{\gamma-\beta} \geqslant (n_0)^\beta.$$

Since $\gamma > \beta$ and $n_0 = 2^{m_0}$ is constant, this is indeed the case. ∎

We end this section by pointing out a similarity between the analysis used here and the analysis carried out in [12]. In both papers, there are two stages. The first stage (prefix of length m_0 in our paper) makes full use of the conditional probability distribution of the channel, while the second stage uses a simpler rule (evolving the bound on the Bhattacharyya parameter in our paper and using an RM rule in [12]).

Appendix A
Proof of Theorem 8

This appendix is devoted to the proof of Theorem 8. Although the initial lemmas needed for the proof are rather intuitive, the latter seem to be a lucky coincidence (probably due to a lack of a deeper understanding on the authors' part). The prime example seems to be (69) in the proof of Lemma 27.

We start by defining some notation. Let $\mathcal{W} : \mathcal{X} \to \mathcal{Y}$, ν, $y_1, y_2, \ldots, y_\nu$ and $\bar{y}_1, \bar{y}_2, \bar{y}_\nu$ be as in Theorem 8. Let $w \in \mathcal{Y}$ and $\bar{w} \in \mathcal{Y}$ be a symbol pair, and denote by (a, b) the corresponding probability pair, where

$$a = p(w|0) = p(\bar{w}|1), \quad b = p(w|1) = p(\bar{w}|0).$$

The contribution of this probability pair to the capacity of $\mathcal{W}$ is denoted by

$$C(a,b) = -(a+b)\log_2((a+b)/2) + a\log_2(a) + b\log_2(b) = -(a+b)\log_2(a+b) + a\log_2(a) + b\log_2(b) + (a+b),$$

where $0\log_2 0 = 0$.

Next, suppose we are given two probability pairs: (a_1, b_1) and (a_2, b_2) corresponding to the symbol pair $w_1, \bar{w}_1$ and $w_2, \bar{w}_2$, respectively. The capacity difference resulting from the application of Lemma 7 to w_1 and w_2 is denoted by

$$\Delta(a_1, b_1; a_2, b_2) = C(a_1, b_1) + C(a_2, b_2) - C(a_1 + a_2, b_1 + b_2).$$

For reasons that will become apparent later on, we henceforth relax the definition of a probability pair to two nonnegative numbers, the sum of which may be greater than 1. Note that $C(a, b)$ is still well defined with respect to this generalization, as is $\Delta(a_1, b_1; a_2, b_2)$. Furthermore, to exclude trivial cases, we require that a probability pair (a, b) has at least one positive element.

The following lemma states that we lose capacity by performing a downgrading merge.

Lemma 23: Let (a_1, b_1) and (a_2, b_2) be two probability pairs. Then,

$$\Delta(a_1, b_1; a_2, b_2) \geqslant 0.$$

Proof: Assume first that a_1, b_1, a_2, b_2 are all positive. In this case, $\Delta(a_1, b_1; a_2, b_2)$ can be written as follows:

$$(a_1 + a_2)\left(\frac{-a_1}{a_1+a_2}\log_2\frac{(a_1+b_1)(a_1+a_2)}{a_1(a_1+b_1+a_2+b_2)} + \frac{-a_2}{a_1+a_2}\log_2\frac{(a_2+b_2)(a_1+a_2)}{a_2(a_1+b_1+a_2+b_2)}\right) + (b_1 + b_2)\left(\frac{-b_1}{b_1+b_2}\log_2\frac{(a_1+b_1)(b_1+b_2)}{b_1(a_1+b_1+a_2+b_2)} + \frac{-b_2}{b_1+b_2}\log_2\frac{(a_2+b_2)(b_1+b_2)}{b_2(a_1+b_1+a_2+b_2)}\right).$$

By Jensen's inequality, both the first two lines and the last two lines can be lower bounded be 0. The proof for cases in which some of the variables equal zero is much the same. ∎

The intuition behind the following lemma is that the order of merging does matter in terms of total capacity lost.

Lemma 24: Let (a_1, b_1), (a_2, b_2), and (a_3, b_3) be three probability pairs. Then,

$$\Delta(a_1, b_1; a_2, b_2) + \Delta(a_1 + a_2, b_1 + b_2; a_3, b_3) = \Delta(a_2, b_2; a_3, b_3) + \Delta(a_1, b_1; a_2 + a_3, b_2 + b_3).$$

Proof: Both sides of the equation equal

$$C(a_1, b_1)+C(a_2, b_2)+C(a_3, b_3)-C(a_1+a_2+a_3, b_1+b_2+b_3).$$

■

Instead of working with a probability pair (a, b), we find it easier to work with a probability sum $\pi = a + b$ and likelihood ratio $\lambda = a/b$. Of course, we can go back to our previous representation as follows. If $\lambda = \infty$ then $a = \pi$ and $b = 0$. Otherwise, $a = \frac{\lambda \cdot \pi}{\lambda+1}$ and $b = \frac{\pi}{\lambda+1}$. Recall that our relaxation of the term "probability pair" implies that π is positive and it may be greater than 1.

Abusing notation, we define the quantity C through λ and π as well. For $\lambda = \infty$ we have $C[\infty, \pi] = \pi$. Otherwise,

$$C[\lambda, \pi] = \pi \left(-\frac{\lambda}{\lambda+1} \log_2 \left(1 + \frac{1}{\lambda} \right) - \frac{1}{\lambda+1} \log_2(1 + \lambda) \right) + \pi.$$

Let us next consider merging operations. The merging of the symbol pair corresponding to $[\lambda_1, \pi_1]$ with that of $[\lambda_2, \pi_2]$ gives a symbol pair with $[\lambda_{1,2}, \pi_{1,2}]$, where

$$\pi_{1,2} = \pi_1 + \pi_2$$

and

$$\lambda_{1,2} = \bar{\lambda}[\pi_1, \lambda_1; \pi_2, \lambda_2] = \frac{\lambda_1 \pi_1 (\lambda_2 + 1) + \lambda_2 \pi_2 (\lambda_1 + 1)}{\pi_1 (\lambda_2 + 1) + \pi_2 (\lambda_1 + 1)}. \quad (62)$$

Abusing notation, define

$$\Delta[\lambda_1, \pi_1; \lambda_2, \pi_2] = C[\lambda_1, \pi_1] + C[\lambda_2, \pi_2] - C[\lambda_{1,2}, \pi_{1,2}].$$

Clearly, we have that the new definition of Δ is symmetric:

$$\Delta[\lambda_1, \pi_1; \lambda_2, \pi_2] = \Delta[\lambda_2, \pi_2; \lambda_1, \pi_1]. \quad (63)$$

Lemma 25: $\Delta[\lambda_1, \pi_1; \lambda_2, \pi_2]$ is monotonic increasing in both π_1 and π_2.

Proof: Recall from (63) that Δ is symmetric, and so it suffices to prove the claim for π_1. Thus, our goal is to prove the following for all $\rho > 0$,

$$\Delta[\lambda_1, \pi_1 + \rho; \lambda_2, \pi_2] \geqslant \Delta[\lambda_1, \pi_1; \lambda_2, \pi_2].$$

At this point, we find it useful to convert back from the likelihood ratio/probability sum representation $[\lambda, \pi]$ to the probability pair representation (a, b). Denote by (a_1, b_1), (a_2, b_2), and (a', b') the probability pairs corresponding to $[\lambda_1, \pi_1]$, $[\lambda_2, \pi_2]$, and $[\lambda_1, \pi_1 + \rho]$, respectively. Let $a_3 = a' - a_1$ and $b_3 = b' - b_1$. Next, since both (a_1, b_1) and (a', b') have the same likelihood ratio, we deduce that both a_3 and b_3 are nonnegative. Under our new notation, we must prove that

$$\Delta(a_1 + a_3, b_1 + b_3; a_2, b_2) \geqslant \Delta(a_1, b_1; a_2, b_2).$$

Since both (a_1, b_1) and (a', b') have likelihood ratio λ_1, this is also the case for (a_3, b_3). Thus, a simple calculation shows that

$$\Delta(a_1, b_1; a_3, b_3) = 0.$$

Hence,

$$\Delta(a_1 + a_3, b_1 + b_3; a_2, b_2) = \Delta(a_1 + a_3, b_1 + b_3; a_2, b_2) + \Delta(a_1, b_1; a_3, b_3).$$

Next, by Lemma 24,

$$\Delta(a_1 + a_3, b_1 + b_3; a_2, b_2) + \Delta(a_1, b_1; a_3, b_3) = \Delta(a_1, b_1; a_2, b_2) + \Delta(a_1 + a_2, b_1 + b_2; a_3, b_3).$$

Since, by Lemma 23, we have that $\Delta(a_1 + a_2, b_1 + b_2; a_3, b_3)$ is nonnegative, we are done. ■

We are now at the point in which our relaxation of the term "probability pair" can be put to good use. Namely, we will now see how to reduce the number of variables involved by one, by taking a certain probability sum to infinity.

Lemma 26: Let λ_1, π_1, and λ_2 be given. Assume that $0 < \lambda_2 < \infty$. Define

$$\Delta[\lambda_1, \pi_1; \lambda_2, \infty] = \lim_{\pi_2 \to \infty} \Delta[\lambda_1, \pi_1; \lambda_2, \pi_2].$$

If $0 < \lambda_1 < \infty$, then

$$\Delta[\lambda_1, \pi_1; \lambda_2, \infty] = \pi_1 \left(-\frac{\lambda_1}{\lambda_1 + 1} \log_2 \left(\frac{1 + \frac{1}{\lambda_1}}{1 + \frac{1}{\lambda_2}} \right) - \frac{1}{\lambda_1 + 1} \log_2 \left(\frac{\lambda_1 + 1}{\lambda_2 + 1} \right) \right). \quad (64)$$

If $\lambda_1 = \infty$, then

$$\Delta[\lambda_1, \pi_1; \lambda_2, \infty] = \pi_1 \left(-\log_2 \left(\frac{1}{1 + \frac{1}{\lambda_2}} \right) \right). \quad (65)$$

If $\lambda_1 = 0$, then

$$\Delta[\lambda_1, \pi_1; \lambda_2, \infty] = \pi_1 \left(-\log_2 \left(\frac{1}{\lambda_2 + 1} \right) \right). \quad (66)$$

Proof: Consider first the case $0 < \lambda_1 < \infty$. We write out $\Delta[\lambda_1, \pi_1; \lambda_2, \pi_2]$ in full and after rearrangement get

$$\pi_1 \left(\frac{1}{\frac{1}{\lambda_{1,2}} + 1} \log_2 \left(1 + \frac{1}{\lambda_{1,2}} \right) - \frac{1}{\frac{1}{\lambda_1} + 1} \log_2 \left(1 + \frac{1}{\lambda_1} \right) \right) + \pi_1 \left(\frac{1}{\lambda_{1,2} + 1} \log_2 (1 + \lambda_{1,2}) - \frac{1}{\lambda_1 + 1} \log_2 (1 + \lambda_1) \right) + \pi_2 \left(\frac{1}{\frac{1}{\lambda_{1,2}} + 1} \log_2 \left(1 + \frac{1}{\lambda_{1,2}} \right) - \frac{1}{\frac{1}{\lambda_2} + 1} \log_2 \left(1 + \frac{1}{\lambda_2} \right) \right) + \pi_2 \left(\frac{1}{\lambda_{1,2} + 1} \log_2 (1 + \lambda_{1,2}) - \frac{1}{\lambda_2 + 1} \log_2 (1 + \lambda_2) \right), \quad (67)$$

where $\lambda_{1,2}$ is given in (62). Next, note that

$$\lim_{\pi_2\to\infty}\lambda_{1,2}=\lambda_2.$$

Thus, applying $\lim_{\pi_2\to\infty}$ to the first two lines of (67) is straightforward. Next, consider the third line of (67), and write its limit as

$$\lim_{\pi_2\to\infty}\frac{\frac{1}{\frac{1}{\lambda_{1,2}}+1}\log_2\left(1+\frac{1}{\lambda_{1,2}}\right)-\frac{1}{\frac{1}{\lambda_2}+1}\log_2\left(1+\frac{1}{\lambda_2}\right)}{\frac{1}{\pi_2}}.$$

Since $\lim_{\pi_2\to\infty}\lambda_{1,2}=\lambda_2$, we get that both numerator and denominator tend to 0 as $\pi_2\to\infty$. Thus, we apply l'Hôpital's rule and get

$$\lim_{\pi_2\to\infty}\frac{\frac{1}{(\lambda_{1,2}+1)^2}\left(\log_2 e-\log_2\left(1+\frac{1}{\lambda_{1,2}}\right)\right)\frac{\partial\lambda_{1,2}}{\partial\pi_2}}{\frac{1}{(\pi_2)^2}}=$$
$$\lim_{\pi_2\to\infty}\frac{1}{(\lambda_{1,2}+1)^2}\left(\log_2 e-\log_2\left(1+\frac{1}{\lambda_{1,2}}\right)\right)\cdot$$
$$\frac{\pi_1(\lambda_2+1)(\lambda_1+1)(\lambda_2-\lambda_1)}{(\frac{\pi_1(\lambda_1+1)+\pi_2(\lambda_1+1)}{\pi_2})^2}=$$
$$\frac{\pi_1(\lambda_2-\lambda_1)}{(\lambda_1+1)(\lambda_2+1)}\left(\log_2 e-\log_2\left(1+\frac{1}{\lambda_2}\right)\right),$$

where $e=2.71828\ldots$ is Euler's number. Similarly, taking the $\lim_{\pi_2\to\infty}$ of the fourth line of (67) gives

$$\frac{\pi_1(\lambda_2-\lambda_1)}{(\lambda_1+1)(\lambda_2+1)}\left(-\log_2 e-\log_2(1+\lambda_2)\right).$$

Thus, a short calculations finishes the proof for this case. The cases $\lambda_1=\infty$ and $\lambda_1=0$ are handled much the same way. ■

The utility of the next Lemma is that it asserts a stronger claim than the "Moreover" part of Theorem 8, for a specific value of λ_2.

Lemma 27: Let probability pairs (a_1,b_1) and (a_3,b_3) have likelihood ratios λ_1 and λ_3, respectively. Assume $\lambda_1\leqslant\lambda_3$. Denote $\pi_1=a_1+b_1$ and $\pi_3=a_3+b_3$. Let

$$\lambda_2=\lambda_{1,3}=\bar{\lambda}[\pi_1,\lambda_1;\pi_3,\lambda_3], \tag{68}$$

as defined in (62). Then,

$$\Delta[\lambda_1,\pi_1;\lambda_2,\infty]\leqslant\Delta[\lambda_1,\pi_1;\lambda_3,\pi_3]$$

and

$$\Delta[\lambda_3,\pi_3;\lambda_2,\infty]\leqslant\Delta[\lambda_1,\pi_1;\lambda_3,\pi_3].$$

Proof: We start by taking care of a trivial case. Note that if it is not the case that $0<\lambda_2<\infty$, then $\lambda_1=\lambda_2=\lambda_3$, and the proof follows easily.

So, we henceforth assume that $0<\lambda_2<\infty$, as was done in Lemma 26. Let

$$\Delta_{(1,3)}=\Delta[\lambda_1,\pi_1;\lambda_3,\pi_3],$$
$$\Delta'_{(1,2)}=\Delta[\lambda_1,\pi_1;\lambda_2,\infty],$$

and

$$\Delta'_{(2,3)}=\Delta[\lambda_3,\pi_3;\lambda_2,\infty].$$

Thus, we must prove that $\Delta'_{(1,2)}\leqslant\Delta_{(1,3)}$ and $\Delta'_{(2,3)}\leqslant\Delta_{(1,3)}$. Luckily, Lemma 26 and a bit of calculation yields that

$$\Delta'_{(1,2)}+\Delta'_{(2,3)}=\Delta_{(1,3)}. \tag{69}$$

Recall that $\Delta'_{(1,2)}$ and $\Delta'_{(2,3)}$ must be nonnegative by Lemmas 23 and 25. Thus, we are done. ■

The next lemma shows how to discard the restraint put on λ_2 in Lemma 27.

Lemma 28: Let the likelihood ratios λ_1, λ_3 and the probability sums π_1, π_3 be as in be as in Lemma 27. Fix

$$\lambda_1\leqslant\lambda_2\leqslant\lambda_3. \tag{70}$$

Then either

$$\Delta[\lambda_1,\pi_1;\lambda_2,\infty]\leqslant\Delta[\lambda_1,\pi_1;\lambda_3,\pi_3] \tag{71}$$

or

$$\Delta[\lambda_3,\pi_3;\lambda_2,\infty]\leqslant\Delta[\lambda_1,\pi_1;\lambda_3,\pi_3]. \tag{72}$$

Proof: Let $\lambda_{1,3}$ be as in (68), and note that

$$\lambda_1\leqslant\lambda_{1,3}\leqslant\lambda_3.$$

Assume w.l.o.g. that λ_2 is such that

$$\lambda_1\leqslant\lambda_2\leqslant\lambda_{1,3}.$$

From Lemma 27 we have that

$$\Delta[\lambda_1,\pi_1;\lambda_{1,3},\infty]\leqslant\Delta[\lambda_1,\pi_1;\lambda_3,\pi_3].$$

Thus, we may assume that $\lambda_2<\lambda_{1,3}$ and aim to prove that

$$\Delta[\lambda_1,\pi_1;\lambda_2,\infty]\leqslant\Delta[\lambda_1,\pi_1;\lambda_{1,3},\infty]. \tag{73}$$

Next, notice that

$$\Delta[\lambda_1,\pi_1;\lambda_2,\infty]=0,\quad\text{if }\lambda_2=\lambda_1. \tag{74}$$

Thus, let us assume that

$$\lambda_1<\lambda_2<\lambda_{1,3}.$$

Specifically, it follows that

$$0<\lambda_2<\infty$$

and thus the assumption in Lemma 26 holds.

Define the function f as follows:

$$f(\lambda'_2)=\Delta[\lambda_1,\pi_1;\lambda'_2,\infty].$$

Assume first that $\lambda_1=0$, and thus by (66), we have that

$$\frac{\partial f(\lambda'_2)}{\partial\lambda'_2}\geqslant 0.$$

On the other hand, if $\lambda_1 \neq 0$ we must have that $0 \leqslant \lambda_1 < \infty$. Thus, by (64), we have that

$$\frac{\partial f(\lambda_2')}{\partial \lambda_2'} = \frac{\pi_1}{(\lambda_1+1)(\lambda_2'+1)}\left(1-\frac{\lambda_1}{\lambda_2'}\right),$$

which is also nonnegative for $\lambda_2' \geqslant \lambda_1$. Thus, we have proved that the derivative is nonnegative in both cases, and this together with (74) proves (73). ■

We are now in a position to prove Theorem 8.

Proof of Theorem 8: We first consider the "Moreover" part of the theorem. Let $[\lambda_1, \pi_1]$, $[\lambda_2, \pi_2]$, and $[\lambda_1, \pi_1]$ correspond to y_i, y_j, and y_k, respectively. From Lemma 28, we have that either (71) or (72) holds. Assume w.l.o.g. that (71) holds. By Lemma 25, we have that

$$\Delta[\lambda_1, \pi_1; \lambda_2, \pi_2] \leqslant \Delta[\lambda_1, \pi_1; \lambda_2, \infty].$$

Thus,

$$\Delta[\lambda_1, \pi_1; \lambda_2, \pi_2] \leqslant \Delta[\lambda_1, \pi_1; \lambda_3, \pi_3],$$

which is equivalent to

$$I(y_j, y_k) \geqslant I(y_i, y_k).$$

Having finished the "Moreover" part, we now turn our attention to the proof of (22). The two equalities in (22) are straightforward, so we are left with proving the inequality. For $\lambda \geqslant 0$ and $\pi > 0$, the following are easily verified:

$$C[\lambda, \pi] = C[1/\lambda, \pi], \tag{75}$$

and

$$C[\lambda, \pi] \text{ increases with } \lambda \geqslant 1. \tag{76}$$

Also, for $\bar{\lambda}$ as given in (62), $\lambda_1, \lambda_2 \geqslant 0$, and $\pi_1 > 0, \pi_2 > 0$, it is easy to show that

$$\bar{\lambda}[\lambda_1, \pi_1, \lambda_2, \pi_2] \text{ increases with both } \lambda_1 \text{ and } \lambda_2. \tag{77}$$

Let $[\lambda_1, \pi_1]$ and $[\lambda_2, \pi_2]$ correspond to y_i and y_j, respectively. Denote

$$\gamma = \bar{\lambda}[\lambda_1, \pi_1; \lambda_2, \pi_2]$$

and

$$\delta = \bar{\lambda}[1/\lambda_1, \pi_1; \lambda_2, \pi_2].$$

Hence, our task reduces to showing that

$$C[\gamma, \pi_1+\pi_2] \geqslant C[\delta, \pi_1+\pi_2]. \tag{78}$$

Assume first that $\delta \geqslant 1$. Recall that both $\lambda_1 \geqslant 1$ and $\lambda_2 \geqslant 1$. Thus, by (77) we conclude that $\gamma \geqslant \delta \geqslant 1$. This, together with (76), finishes the proof.

Conversely, assume that $\delta \leqslant 1$. Since

$$\bar{\lambda}[\lambda_1, \pi_1; \lambda_2, \pi_2] = \bar{\lambda}[1/\lambda_1, \pi_1; 1/\lambda_2, \pi_2]$$

we now get from (77) that $\gamma \leqslant \delta \leqslant 1$. This, together with (75) and (76), finishes the proof. ■

Appendix B
Proof of Theorem 13

As a preliminary step toward the proof of Theorem 13, we convince ourselves that the notation $\Delta[\lambda_1; \lambda_2, \pi_2; \lambda_3]$ used in the theorem is indeed valid. Specifically, the next lemma shows that knowledge of the arguments of Δ indeed suffices to calculate the difference in capacity. The proof is straightforward.

Lemma 29: For $i = 1, 2, 3$, let y_i and λ_i, as well as π_2 be as in Theorem 13. If $\lambda_3 < \infty$, then

$$\begin{aligned}\Delta[\lambda_1; \lambda_2, \pi_2; \lambda_3] = \frac{\pi_2}{(\lambda_2+1)(\lambda_1-\lambda_3)}\Bigg[\\ (\lambda_3-\lambda_2)\left(\lambda_1 \log_2\left(1+\frac{1}{\lambda_1}\right)+\log_2(1+\lambda_1)\right)+ \\ (\lambda_2-\lambda_1)\left(\lambda_3 \log_2\left(1+\frac{1}{\lambda_3}\right)+\log_2(1+\lambda_3)\right)+ \\ (\lambda_1-\lambda_3)\left(\lambda_2 \log_2\left(1+\frac{1}{\lambda_2}\right)+\log_2(1+\lambda_2)\right)\Bigg].\end{aligned} \tag{79}$$

Otherwise, $\lambda_3 = \infty$ and

$$\begin{aligned}\Delta[\lambda_1; \lambda_2, \pi_2; \lambda_3 = \infty] = \\ \frac{\pi_2}{\lambda_2+1}\Bigg[-\lambda_1 \log_2\left(1+\frac{1}{\lambda_1}\right)-\log_2(1+\lambda_1) \\ +\lambda_2 \log_2\left(1+\frac{1}{\lambda_2}\right)+\log_2(1+\lambda_2)\Bigg].\end{aligned} \tag{80}$$

Having the above calculations at hand, we are in a position to prove Theorem 13.

Proof of Theorem 13: First, let consider the case $\lambda_3 < \infty$. Since our claim does not involve changing the values of λ_2 and π_2, let us fix them and denote

$$f(\lambda_1, \lambda_3) = \Delta[\lambda_1; \lambda_2, \pi_2; \lambda_3].$$

Under this notation, it suffices to prove that $f(\lambda_1, \lambda_3)$ is decreasing in λ_1 and increasing in λ_3, where $\lambda_1 < \lambda_2 < \lambda_3$. A simple calculation shows that

$$\begin{aligned}\frac{\partial f(\lambda_1, \lambda_3)}{\partial \lambda_1} = \frac{-\pi_2(\lambda_3-\lambda_2)}{(1+\lambda_2)(\lambda_3-\lambda_1)^2}\Bigg[\\ \lambda_3 \log\left(\frac{1+\frac{1}{\lambda_1}}{1+\frac{1}{\lambda_3}}\right)+\log\left(\frac{1+\lambda_1}{1+\lambda_3}\right)\Bigg].\end{aligned} \tag{81}$$

So, in order to show that $f(\lambda_1, \lambda_3)$ is decreasing in λ_1, it suffices to show that the term inside the square brackets is positive for all $\lambda_1 < \lambda_3$. Indeed, if we denote

$$g(\lambda_1, \lambda_3) = \lambda_3 \log\left(\frac{1+\frac{1}{\lambda_1}}{1+\frac{1}{\lambda_3}}\right)+\log\left(\frac{1+\lambda_1}{1+\lambda_3}\right),$$

then is readily checked that

$$g(\lambda_1, \lambda_1) = 0,$$

while

$$\frac{\partial g(\lambda_1, \lambda_3)}{\partial \lambda_1} = \frac{\lambda_3 - \lambda_1}{\lambda_1(\lambda_1 + 1)}$$

is positive for $\lambda_3 > \lambda_1$. The proof of $f(\lambda_1, \lambda_3)$ increasing in λ_3 is exactly the same, up to a change of variable names.

Let us now consider the second case, $\lambda_3 = \infty$. Similarly to what was done before, let us fix λ_2 and π_2, and consider $\Delta[\lambda_1; \lambda_2, \pi_2; \lambda_3 = \infty]$ as a function of λ_1. Denote

$$h(\lambda_1) = \Delta[\lambda_1; \lambda_2, \pi_2; \lambda_3 = \infty].$$

Under this notation, our aim is to prove that $h(\lambda_1)$ is decreasing in λ_1. Indeed,

$$\frac{\partial h(\lambda_1)}{\partial \lambda_1} = \frac{-\pi_2 \log_2 \left(1 + \frac{1}{\lambda_1}\right)}{\lambda_2 + 1}$$

is easily seen to be negative. ■

ACKNOWLEDGMENT

We thank E. Abbe, E. Arıkan, H. Hassani, R. Pedarsani, U. Pereg, E. Şaşoğlu, A. Sharov, E. Telatar, and R. Urbanke for helpful discussions. We are especially grateful to U. Pereg and A. Sharov for going over many incarnations of this paper and offering valuable comments.

REFERENCES

[1] E. Abbe, Extracting Randomness and dependencies via a matrix polarization 2011 [Online]. Available: arXiv:1102.1247v1

[2] E. Abbe and E. Telatar, Polar codes for the m-user MAC and matroids 2010 [Online]. Available: arXiv:1002.0777v2

[3] E. Arıkan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. Inf Theory*, vol. 55, no. 7, pp. 3051–3073, Jul. 2009.

[4] E. Arıkan, Source polarization 2010 [Online]. Available: arXiv:1001.3087v2

[5] E. Arıkan and E. Telatar, "On the rate of channel polarization," in *Proc. IEEE Symp. Inf. Theory*, Seoul, South Korea, 2009, pp. 1493–1495.

[6] D. Burshtein and A. Strugatski, Polar write once memory codes 2012 [Online]. Available: arXiv:1207.0782v2

[7] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. Cambridge, MA, USA: MIT Press, 2001.

[8] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. New York, NY, USA: Wiley, 2006.

[9] R. G. Gallager, *Information Theory and Reliable Communications*. New York, NY, USA: Wiley, 1968.

[10] A. Goli, S. H. Hassani, and R. Urbanke, "Universal bounds on the scaling behavior of polar codes," in *Proc. IEEE Symp. Inf. Theory*, Cambridge, MA, USA, 2012, pp. 1957–1961.

[11] V. Guruswami and P. Xia, Polar Codes: speed of polarization and polynomial gap to capacity 2013 [Online]. Available: http://eccc.hpi-web.de/report/2013/050/

[12] S. H. Hassani, R. Mori, T. Tanaka, and R. Urbanke, Rate-dependent analysis of the asymptotic behavior of channel polarization 2011 [Online]. Available: arXiv:1110.0194v2

[13] S. B. Korada, "Polar Codes for Channel and Source Coding," Ph.D. dissertation, Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, 2009.

[14] S. B. Korada, E. Şaşoğlu, and R. Urbanke, "Polar codes: Characterization of exponent, bounds, and constructions," *IEEE Trans. Inf. Theory*, vol. 56, no. 12, pp. 6253–6264, Dec. 2010.

[15] B. M. Kurkoski and H. Yagi, Quantization of binary-input discrete memoryless channels with applications to LDPC decoding 2011 [Online]. Available: arXiv:11107.5637v1

[16] H. Mahdavifar and A. Vardy, "Achieving the secrecy capacity of wiretap channels using polar codes," *IEEE Trans. Inf. Theory*, vol. 57, no. 10, pp. 6428–6443, Oct. 2011.

[17] R. Mori, "Properties and construction of polar codes," Master's thesis, Kyoto Univ., Kyoto, Japan, 2010.

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

[19] R. Pedarsani, S. H. Hassani, I. Tal, and E. Telatar, "On the construction of polar codes," in *Proc. IEEE Symp. Inf. Theory*, Saint Petersburg, Russia, 2011, pp. 11–15.

[20] T. Richardson and R. Urbanke, *Modern Coding Theory*. Cambridge, U.K.: Cambridge Univ. Press, 2008.

[21] E. Şaşoğlu, "Polarization in the presence of memory," in *Proc. IEEE Symp. Inf. Theory*, Saint Petersburg, Russia, 2011, pp. 189–193.

[22] E. Şaşoğlu, E. Telatar, and E. Arıkan, Polarization for arbitrary discrete memoryless channels 2009 [Online]. Available: arXiv:0908.0302v1

[23] E. Şaşoğlu, E. Telatar, and E. Yeh, Polar codes for the two-user multiple-access channel 2010 [Online]. Available: arXiv:1006.4255v1

[24] I. Tal and A. Vardy, "List decoding of polar codes," in *Proc. IEEE Symp. Inf. Theory*, Saint Petersburg, Russia, 2011, pp. 1–5.

Ido Tal was born in Haifa, Israel, in 1975. He received the B.Sc., M.Sc., and Ph.D. degrees in computer science from Technion—Israel Institute of Technology, Haifa, Israel, in 1998, 2003 and 2009, respectively. During 2010–2012 he was a postdoctoral scholar at the University of California at San Diego. In 2012 he joined the Electrical Engineering Department at Technion. His research interests include constrained coding and error-control coding.

Alexander Vardy (S'88–M'91–SM'94–F'98) was born in Moscow, U.S.S.R., in 1963. He earned his B.Sc. (summa cum laude) from the Technion, Israel, in 1985, and Ph.D. from the Tel-Aviv University, Israel, in 1991. During 1985–1990 he was with the Israeli Air Force, where he worked on electronic counter measures systems and algorithms. During the years 1992 and 1993 he was a Visiting Scientist at the IBM Almaden Research Center, in San Jose, CA. From 1993 to 1998, he was with the University of Illinois at Urbana-Champaign, first as an Assistant Professor then as an Associate Professor. Since 1998, he has been with the University of California San Diego (UCSD), where he is the Jack Keil Wolf Endowed Chair Professor in the Department of Electrical and Computer Engineering, with joint appointments in the Department of Computer Science and the Department of Mathematics. While on sabbatical from UCSD, he has held long-term visiting appointments with CNRS, France, the EPFL, Switzerland, and the Technion, Israel.

His research interests include error-correcting codes, algebraic and iterative decoding algorithms, lattices and sphere packings, coding for digital media, cryptography and computational complexity theory, and fun math problems.

He received an IBM Invention Achievement Award in 1993, and NSF Research Initiation and CAREER awards in 1994 and 1995. In 1996, he was appointed Fellow in the Center for Advanced Study at the University of Illinois, and received the Xerox Award for faculty research. In the same year, he became a Fellow of the Packard Foundation. He received the IEEE Information Theory Society Paper Award (jointly with Ralf Koetter) for the year 2004. In 2005, he received the Fulbright Senior Scholar Fellowship, and the Best Paper Award at the IEEE Symposium on Foundations of Computer Science (FOCS). During 1995–1998, he was an Associate Editor for Coding Theory and during 1998–2001, he was the Editor-in-Chief of the IEEE TRANSACTIONS ON INFORMATION THEORY. From 2003 to 2009, he was an Editor for the SIAM *Journal on Discrete Mathematics*. He has been a member of the Board of Governors of the IEEE Information Theory Society during 1998–2006, and again starting in 2011.