Polar Codes for Channels with Insertions, Deletions, and Substitutions

Henry D. Pfister
Duke University

Ido Tal
Technion

Abstract—**This paper presents a coding scheme for an insertion deletion substitution channel. We extend a previous scheme for the deletion channel where polar codes are modified by adding "guard bands" between segments. In the new scheme, each guard band is comprised of a middle segment of '1' symbols, and left and right segments of '0' symbols. Our coding scheme allows for a regular hidden-Markov input distribution, and achieves the information rate between the input and corresponding output of such a distribution. Thus, we prove that our scheme can be used to efficiently achieve the capacity of the channel. The probability of error of our scheme decays exponentially in the cube-root of the block length.**

I. Introduction

In many communications systems, symbol-timing errors can result in insertion and deletion errors. For example, the insertion deletion substitution (IDS) channel maps a length-N input string to a finite output string by sampling an i.i.d. output process for each input that selects between insertion, deletion, and substitution. These types of channels were first studied in the 1960s [1], [2] and modern coding techniques were first applied to them in [3]. Over the past 15 years, bounds on the capacity of synchronization error channels have been significantly improved [4]–[8].

In [9], [10], a capacity-achieving coding scheme is introduced for the deletion channel based on polar codes. The construction and proof builds upon many earlier results (e.g., [11]–[19]); see [10] for a detailed description of these connections. See also [20], which considers a setting related to ours.

The construction in [9], [10] is based on generating codewords consisting of smaller blocks separated by guard bands. After reception, the overall output sequence is separated into blocks associated with the smaller input blocks. However, the separation process changes the effective channel experienced by the small blocks. In particular, the guard bands are long blocks of zeros and the separation process removes all zeros on either side of the small block. The analysis in [9], [10] shows that the resulting channel, dubbed the trimmed deletion channel (TDC), polarizes weakly and has the same mutual information rate as the original deletion channel. Due to the possibly unbounded memory in the deletion channel, the standard extension [21] to strong polarization does not work. Instead, strong polarization can be shown for the polar combining of these small blocks due to the independence provided by the guard bands. These elements complete the achievability proof for the deletion channel.

In this paper, we apply roughly the same coding scheme to the IDS channel. The main difference is that separating the overall output sequence into smaller blocks is more challenging. For the deletion channel, an input that only contains zeros always gives an output that only contains zeros. Thus, the separation process consists of parsing into small blocks and removing zeros from the edges. For the IDS channel, an input only containing zeros typically gives an output containing both zeros and ones. Fortunately, the expected fraction of zeros will be noticeably larger than the fraction of ones. This observation along with a more complicated parsing process can be used to separate the overall output sequence into blocks associated with the smaller input blocks.

The key challenge is designing the parsing process so that the effective channel experienced by the small block can be analyzed. In particular, our parsing process produces segments that can be seen roughly as the IDS output of an input consisting of a prefix of zeros, the original input data, and a suffix of zeros. For all the small output blocks, the prefix and suffix lengths are i.i.d. random variables with a known distribution. We refer to the resulting channel as the dirty zero-padded (DZP) IDS channel. To establish the coding theorem for the IDS channel, we must show three things. First, that our parsing of the IDS channel output gives small blocks whose joint input-output distribution matches that of the DZP channel. Second, that the DZP channel polarizes weakly and has the same mutual information rate as the original IDS channel. Third, that the trellis representation of the joint input-output distribution of the IDS channel [3] can be modified to give the joint input-output distribution of the DZP channel. In this work, we establish these three elements and describe the first two elements herein.

By combining the parsing process described in this paper with the results of [9], [10], one gets the following theorem. Due to space limitations, many details are deferred to the extended version of this paper.

Theorem 1. *Fix a regular hidden-Markov input process and a parameter $\nu \in (0, 1/3]$. The rate of our coding scheme approaches the mutual information rate between the input process and the binary IDS channel output. The encoding and decoding complexities are $O(\Lambda \log \Lambda)$ and $O(\Lambda^{1+3\nu})$, respectively, where Λ is the blocklength. For any $0 < \nu' < \nu$ and sufficiently large blocklength Λ, the probability of decoding error is at most $2^{-\Lambda^{\nu'}}$.*

The structure of this paper is as follows. In Section II we define the IDS channel, and also a close variant which we term the "dirty zero padding IDS channel" (DZP). Section III details how encoding is done. In Section IV, we define two decoding methods. Namely, we first define a decoding method executed by a genie, which is in possession of some extra information (it knows where the "commas" which separate the outputs corresponding to certain input blocks are). The utility of the genie's decoding method is that it is easy to analyze (the DZP channel is used in the analysis). We then define a second decoding method: Aladdin's decoding method. Since Aladdin is a mere mortal, he does not have knowledge of where the above commas lie. That is, Aladdin's method is the one we can actually implement. The main trick is to show that with very high probability, the genie's decoder and Aladdin's decoder produce the exact same result.

II. Channel models

In this section, we define the IDS and DZP channels.

A. Dobrushin's Channel and the IDS Channel

In 1967, Dobrushin introduced a general class of channels with synchronization errors and proved a random coding theorem for that class [2]. The model consists of a finite input alphabet $\mathcal{X}$ and a conditional distribution $p_{Y|X}(\cdot|x)$ over finite output strings $\mathcal{Y} \subseteq \mathcal{X}^* = \cup_{n=0}^{\infty} \mathcal{X}^n$ given $x \in \mathcal{X}$, where ϵ denotes the empty string of length 0. For the input $\mathbf{x} = (x_1, x_2, \ldots, x_N) \in \mathcal{X}^N$, the channel output is generated by drawing $Y_n \sim p_{Y|X}(\cdot|x_n)$ i.i.d. and concatenating to get

$$\mathbf{Y} = Y_1 \odot Y_2 \odot \cdots \odot Y_N.$$

For example, the binary deletion channel with deletion probability p_{d} has $\mathcal{X} = \{0,1\}$ and $\mathcal{Y} = \{\epsilon, 0, 1\}$ with non-zero probabilities $P_{Y|X}(\epsilon|x) = p_{\mathrm{d}}$ and $P_{Y|X}(x|x) = 1 - p_{\mathrm{d}}$ for all $x \in \mathcal{X}$. Similarly, the binary IDS channel we consider has IDS probabilities $(p_{\mathrm{i}}, p_{\mathrm{d}}, p_{\mathrm{s}})$, $\mathcal{X} = \{0,1\}$, and $\mathcal{Y} = \{\epsilon, 0, 1, 00, 01, 10, 11\}$ with non-zero probabilities $P_{Y|X}(\epsilon|x) = p_{\mathrm{d}}$, $P_{Y|X}(x|x) = 1 - p_{\mathrm{i}} - p_{\mathrm{d}} - p_{\mathrm{s}}$, $P_{Y|X}(\overline{x}|x) = p_{\mathrm{s}}$, $P_{Y|X}(0x|x) = P_{Y|X}(1x|x) = p_{\mathrm{i}}/2$ for all $x \in \mathcal{X}$.

While we focus on this binary IDS channel for concreteness, the approach described here should generalize to any well-behaved binary-input Dobrushin channel for which the output distribution associated with the all-zero input is distinguishable from finite shifts of the output distribution associated with the all-one input. For simplicity, we focus on the case where they are distinguishable simply by counting ones and zeros.

Define $\alpha_{0|x}$ ($\alpha_{1|x}$) as the expected number of 0 (1) symbols at the output of the channel, given that the input was $x \in \mathcal{X}$. Note that the expected length of an output, given that the input was $x \in \mathcal{X}$ is $\alpha_{0|x} + \alpha_{1|x}$. We require that this sum is independent of x, and denote it as

$$\beta = \alpha_{0|0} + \alpha_{1|0} = \alpha_{0|1} + \alpha_{1|1} \ . \tag{1}$$

We also require an "advantage" to x at the output, if the input was x. That is, we require that

$$\alpha_{0|0} > \alpha_{1|0} \qquad \text{and} \qquad \alpha_{1|1} > \alpha_{0|1} \ , \tag{2}$$

and denote

$$\gamma \triangleq \frac{\min\{\alpha_{0|0} - \alpha_{1|0}, \alpha_{1|1} - \alpha_{0|1}\}}{2} > 0 \ , \tag{3}$$

where the inequality follows by (2).

Informally, the above "advantage to the input at the output" will allow us to differentiate between a long input of 0 symbols and a long input of 1 symbols. Specifically, fix a window length $h > 0$ and an $x \in \mathcal{X}$. Then, generate an output sequence of length at least $h + 1$ and optionally remove the first output bit. Then, we count the number of 0 symbols contained in the first h positions of the string. If it is at least $h/2$, then we declare that $x = 0$; otherwise, we declare that $x = 1$. The following lemma states that we have a very high chance of guessing correctly, for large enough window length h.

Lemma 2. *Let $x \in \mathcal{X}$ be fixed, and let a window length $h \geq h_0$ be given, where h_0 is a constant dependent on the channel. Let $\mathbf{Y}$ be a string of length h generated by truncating the output associated with the all-x input where the first output bit is optionally removed. Then, the probability that $\mathbf{Y}$ contains fewer than $h/2$ bits equal to x is less than*

$$\mathrm{err}_h \triangleq e^{-h \cdot c_0} \ ,$$

where c_0 is a positive constant dependent on the channel.

Proof: See Appendix. ■

B. Dirty-Zero-Padding IDS channel

The DZP channel $W^\star$ is defined by the IDS channel W, the input blocklength N_0, and two probability distributions over $\mathcal{X}^*$, P_{padLeft} and P_{padRight}. Given the length-N_0 input string $\mathbf{x}$, we first pass $\mathbf{x}$ through the IDS channel W and let $\mathbf{y}_{\mathrm{middle}}$ denote the output. Next, we draw two independent vectors $\mathbf{y}_{\mathrm{left}}$ and $\mathbf{y}_{\mathrm{right}}$ according to the probability distributions P_{padLeft} and P_{padRight}, respectively. The output of the DZP channel is then given by

$$\mathbf{y}^\star = \mathbf{y}_{\mathrm{left}} \odot \mathbf{y}_{\mathrm{middle}} \odot \mathbf{y}_{\mathrm{right}} \ . \tag{4}$$

We will specify P_{padLeft} and P_{padRight} later. For now, let us say informally that $\mathbf{y}_{\mathrm{left}}$ and $\mathbf{y}_{\mathrm{right}}$ are the result of passing strings of '0' symbols through the channel W. Hence the name: we pad $\mathbf{y}$ from the left and right by vectors corresponding to zeros "dirtied" by passing through the channel W.

Informally, the following lemma states that, in the limit as $N_0 \to \infty$, the mutual information rates of W and $W^\star$ are equal. As will become apparent later, the maximum possible length of $\mathbf{y}_{\mathrm{left}}$ and the maximum possible length of $\mathbf{y}_{\mathrm{right}}$ both grow sub-linearly in N_0. Hence, the condition of the lemma is not vacant.

Lemma 3. *Let $\mathbf{X} \in \mathcal{X}^{N_0}$ be a random vector of length N_0. Let $\mathbf{Y}$ and $\mathbf{Y}^\star$ be the outputs gotten when $\mathbf{X}$ is input to the IDS channel W and the DZP channel $W^\star$, respectively. Let N_0 be large enough so that the maximum length that $\mathbf{y}_{\mathrm{left}}$ can take and the maximum length that $\mathbf{y}_{\mathrm{right}}$ can take are both at most $N_0 - 1$. Then,*

$$\frac{I(\mathbf{X};\mathbf{Y}) - 2\log_2 N_0}{N_0} \leq \frac{I(\mathbf{X};\mathbf{Y}^\star)}{N_0} \leq \frac{I(\mathbf{X};\mathbf{Y})}{N_0} \ .$$

Proof. It suffices to prove the inequalities for the numerators since the denominators all equal N_0. The inequality $I(\mathbf{X};\mathbf{Y}^\star) \leq I(\mathbf{X};\mathbf{Y})$ follows by the data-processing inequality, since $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Y}^\star$ form a Markov chain, in that order. We will show that

$$I(\mathbf{X};\mathbf{Y}) - 2\log_2 N_0 \leq I(\mathbf{X};\mathbf{Y}^\star) \ . \tag{5}$$

Let us first denote $\mathbf{Y}^\star = \mathbf{Y}_{\text{left}} \odot \mathbf{Y}_{\text{middle}} \odot \mathbf{Y}_{\text{right}}$, as described above. Next, note that we can assume w.l.o.g. that $\mathbf{Y} = \mathbf{Y}_{\text{middle}}$. Finally, note that $\mathbf{X}$, $(\mathbf{Y}^\star, |\mathbf{Y}_{\text{left}}|, |\mathbf{Y}_{\text{right}}|)$, and $\mathbf{Y}$ form a Markov chain, in the order, where $|\cdot|$ denotes the length of a string. Thus,

$$\begin{aligned} I(\mathbf{X};\mathbf{Y}) &\le I(\mathbf{X};\mathbf{Y}^\star, |\mathbf{Y}_{\text{left}}|, |\mathbf{Y}_{\text{right}}|) \\ &\le I(\mathbf{X};\mathbf{Y}^\star) + H(|\mathbf{Y}_{\text{left}}|) + H(|\mathbf{Y}_{\text{right}}|) \\ &\le I(\mathbf{X};\mathbf{Y}^\star) + 2\log_2(N_0) \ , \end{aligned}$$

because both $|\mathbf{Y}_{\text{left}}|$ and $|\mathbf{Y}_{\text{right}}|$ can take at most N_0 different values. Thus, (5) holds and the proof is complete. □

III. Encoding

Suppose for a moment that we were coding not for the IDS channel W, but for the DZP channel $W^\star$. First of all, recall that the channel $W^\star$ accepts a block of length N_0 bits. We choose a typically "large" N_0. However, instead of only sending a single block of length N_0, we send Φ such blocks, denoted $\mathbf{x}(1), \mathbf{x}(2), \ldots, \mathbf{x}(\Phi)$. The important point to note is the output: denote the output of $W^\star$ corresponding to $\mathbf{x}(i)$ as $\mathbf{y}^\star(i)$. We assume that the output corresponding to the above input is $(\mathbf{y}^\star(1), \mathbf{y}^\star(2), \ldots, \mathbf{y}^\star(\Phi))$, as opposed to $\mathbf{y}^\star(1) \odot \mathbf{y}^\star(2) \odot \cdots \odot \mathbf{y}^\star(\Phi)$. That is, we assume that the output blocks corresponding to the input blocks are *punctuated.* Namely, given the output corresponding to Φ blocks, we can distinguish the output corresponding to input block $\mathbf{x}(i)$. This is in stark contrast to W, in which no such punctuation is given.

For this setting, one can both encode and decode using polar codes; this is very similar to what was done in [10] with the DZP channel playing the role of the block-TDC channel. Given the information symbols and frozen indices, the information symbols are mapped to a polar codeword of length ΦN_0 using polar encoding. Also, extending the ideas in [3], [10], we can build a trellis for calculating the joint probability of $\mathbf{x}$ being the input to $W^\star$ and $\mathbf{y}^\star$ being the output (building such a trellis involves the use of P_{padLeft} and P_{padRight}). Finally, using Lemma 3 and essentially the same proof as [10], this coding scheme can approach the capacity of the IDS channel W. Due to lack of space, we do not go into further details.

Our coding scheme for the IDS channel W consists of two phases. In the first phase, we produce the blocks $\mathbf{x}(1), \mathbf{x}(2), \ldots, \mathbf{x}(\Phi)$ by taking the whole polar codeword and adding commas to separate into blocks of length N_0. Then, we imagine these blocks being transmitted over $W^\star$. In the second phase, we add guard bands (defined shortly) between the above blocks. The result is a long codeword that is transmitted over the channel W. Loosely speaking, the purpose of the guard bands is to allow the decoder to simulate the operation of $W^\star$ on the blocks $\mathbf{x}(1), \mathbf{x}(2), \ldots, \mathbf{x}(\Phi)$, even though we are in fact transmitting over the channel W.

Denote $N = 2^n$, $N_0 = 2^{n_0}$ and $\Phi = 2^{n_1} = 2^{n-n_0}$, where $n_0 = \lceil n\nu \rceil$ and ν was fixed in Theorem 1. Let

$$\mathbf{x} = \mathbf{x}(1) \odot \mathbf{x}(2) \odot \cdots \odot \mathbf{x}(\Phi) \tag{6}$$

be a vector of length N, consisting of blocks $\mathbf{x}(i)$, each of length N_0. We denote by $g(\mathbf{x})$ the result of adding guard bands to $\mathbf{x}$. For this, let us denote $\mathbf{x} = \mathbf{x}_{\text{I}} \odot \mathbf{x}_{\text{II}}$, where $\mathbf{x}_{\text{I}}$ and $\mathbf{x}_{\text{II}}$ are the left and right halves of $\mathbf{x}$, each of length $N/2$.

$$g(\mathbf{x}) \triangleq \begin{cases} \mathbf{x} & \text{if } n \le n_0 \\ g(\mathbf{x}_{\text{I}}) \odot \mathbf{g}_n \odot g(\mathbf{x}_{\text{II}}) & \text{if } n > n_0, \end{cases} \tag{7}$$

where $\mathbf{g}_n$ is termed the guard band and defined as follows. Denote by $\mathbf{0}(\ell)$ and $\mathbf{1}(\ell)$ a string of ℓ consecutive '0' symbols and a string of ℓ consecutive '1' symbols, respectively. Let

$$\ell_n \triangleq 2^{\lfloor (1-\xi)(n-1) \rfloor} \ ,$$

where $\xi \in (0, 1/2)$ is a 'small' constant determined by the difference between ν and ν' in Theorem 1. Then,

$$\mathbf{g}_n \triangleq \underbrace{\mathbf{0}(\ell_{n_0})}_{\mathbf{g}_n^{\text{left}}} \odot \overbrace{\underbrace{\mathbf{1}(\ell_n)}_{\mathbf{g}_n^{\text{midleft}}} \odot \underbrace{\mathbf{1}(\ell_n)}_{\mathbf{g}_n^{\text{midright}}}}^{\mathbf{g}_n^{\text{mid}}} \odot \underbrace{\mathbf{0}(\ell_{n_0})}_{\mathbf{g}_n^{\text{right}}} \ . \tag{8}$$

We note that $\mathbf{g}_n^{\text{left}}$ and $\mathbf{g}_n^{\text{right}}$ are not, in fact, functions of n.

IV. Decoding

We now consider two settings for decoding. In both settings, a vector $g(\mathbf{x})$ is transmitted over the IDS channel W, and the corresponding output is $\mathbf{y}$. Both settings differ only in their preliminary step, which parses the received vector $\mathbf{y}$ into Φ sub-vectors. In the first setting, which we call "genie parsing", an all-knowing genie receives the output $\mathbf{y}$ and adds commas in certain appropriate places. Recall from (6) that $\mathbf{x}$ is comprised of Φ blocks. After adding commas to the output, the genie produces for each block $\mathbf{x}(i)$ a corresponding output $\mathbf{y}^\star(i)$. The result is a series of outputs $\mathbf{y}^\star = (\mathbf{y}^\star(1), \mathbf{y}^\star(2), \ldots, \mathbf{y}^\star(\Phi))$, where for each i, the probability law of $\mathbf{y}^\star(i)$ given $\mathbf{x}^\star(i)$ is the DZP channel $W^\star(\mathbf{y}^\star(i)|\mathbf{x}(i))$. We then use the methods described in [10] to decode $\mathbf{x}$ from $\mathbf{y}^\star$.

The second setting is called "Aladdin parsing". As before, $g(\mathbf{x})$ is transmitted and $\mathbf{y}$ is received. The goal of Aladdin is to produce the same sequence $\mathbf{y}^\star$ as the genie. Since Aladdin is a mere mortal, he does not have the knowledge required to guarantee that he will add commas in the appropriate places.

This raises the question, "Why does the genie output have dirty zero-padding?". An all-knowing genie could produce the IDS output sequences $\mathbf{y}(1), \ldots, \mathbf{y}(\Phi)$. But, our genie chooses a weaker strategy (based on an i.i.d. dither sequence) so that Aladdin can hope to match the genie's parsing by making use of the guard bands. Thus, we will show that Aladdin can succeed in producing $\mathbf{y}^\star$ with very high probability.

A. Genie parsing

Recall from (7) that the codeword we transmit is comprised of blocks $\mathbf{x}(i)$, $1 \le i \le \Phi$, separated by guard bands. Let $\mathbf{g}(i)$ denote the guard band between $\mathbf{x}(i)$ and $\mathbf{x}(i+1)$, where $\mathbf{g}(i)$ equals $\mathbf{g}_{n'}$ for some $n_0 < n' \le n$ which is a function of i. Now we recall from (8) that each guard band $\mathbf{g}(i)$ is comprised of four blocks, which we denote $\mathbf{g}^{\text{left}}(i)$, $\mathbf{g}^{\text{midleft}}(i)$, $\mathbf{g}^{\text{midright}}(i)$, and $\mathbf{g}^{\text{right}}(i)$, for $1 \le i \le \Phi - 1$. The genie receives the output $\mathbf{y}$, and adds commas between all the blocks because the genie can distinguish which substring of $\mathbf{y}$ equals $\mathbf{y}(i)$, the output corresponding to $\mathbf{x}(i)$. It can also distinguish which part of $\mathbf{y}$ corresponds to $\mathbf{g}^{\square}(i)$, where $\square \in \{\text{left}, \text{midleft}, \text{midright}, \text{right}\}$. We denote the relevant part of $\mathbf{y}$ as $\mathbf{d}^{\square}(i)$, where "d" stands for "dirty".

Recall from (4) that, in order to return

$$\mathbf{y}^\star(i) = \mathbf{y}_{\text{left}}(i) \odot \mathbf{y}(i) \odot \mathbf{y}_{\text{right}}(i) \ ,$$

the DZP channel $W^\star$ must pad $\mathbf{y}(i)$ from the left and right. This padding is according to the probability distributions P_{padLeft} and P_{padRight}, which have yet to be specified. Now, we define how the genie produces $\mathbf{y}_{\text{left}}(i)$ and $\mathbf{y}_{\text{right}}(i)$ from the following punctuated segment of $\mathbf{y}^\star$,

$$\mathbf{d}^{\text{midright}}(i-1), \mathbf{d}^{\text{right}}(i-1), \mathbf{y}(i), \mathbf{d}^{\text{left}}(i), \mathbf{d}^{\text{midleft}}(i) \ .$$

In doing this, we *implicitly define* P_{padLeft} and P_{padRight} as the distributions of $\mathbf{y}_{\text{left}}(i)$ and $\mathbf{y}_{\text{right}}(i)$. Before we proceed, we encourage the reader to validate the following points: $\mathbf{y}_{\text{left}}(i)$ and $\mathbf{y}_{\text{right}}(i)$ are independent and their distributions

- depend on the channel statistics of W;
- are not functions of i;
- are not functions of either $\mathbf{x}(i)$ nor $\mathbf{y}(i)$.

Consider an index $1 < i < \Phi$ (not the first nor last block). We now describe how $\mathbf{y}_{\text{left}}(i)$ depends on $\mathbf{d}^{\text{midright}}(i-1)$ and $\mathbf{d}^{\text{right}}(i-1)$. The description of how $\mathbf{y}_{\text{right}}(i)$ depends on $\mathbf{d}^{\text{left}}(i)$ and $\mathbf{d}^{\text{midleft}}(i)$ is given by reflection symmetry. Before diving into the details, we emphasize that $\mathbf{y}_{\text{left}}(i)$ will consist of some suffix of $\mathbf{d}^{\text{right}}(i-1)$. Since $\mathbf{d}^{\text{right}}(i-1)$ is the result of sending a string of zeros, $\mathbf{g}^{\text{right}}(i-1)$, we will indeed pad $\mathbf{y}(i)$ with a string of "dirty zeros". Here are the details.

1) The genie considers the length of $\mathbf{d}^{\text{midright}}(i-1)$.
   a) If it is less than h, where
   $$h \triangleq \frac{\ell_{n_0} \cdot \beta}{4} \ ,$$
   the genie pads $\mathbf{d}^{\text{midright}}(i-1)$ from the left. This is done by conceptually drawing a string from $p_{Y|X}(\cdot|1)$ and prepending $\mathbf{d}^{\text{midright}}(i-1)$ with the string. In practice, we use independent random variables to simulate $p_{Y|X}$. This is repeated until the length of $\mathbf{d}^{\text{midright}}(i-1)$ is at least h.
2) The genie considers the concatenated string
$$\mathbf{z} = \mathbf{d}^{\text{midright}}(i-1) \odot \mathbf{d}^{\text{right}}(i-1) \ .$$
It places a window of length h at the right side of $\mathbf{d}^{\text{midright}}(i-1)$. That is, the window starts at z_s and ends at z_e, where $e = |\mathbf{d}^{\text{midright}}(i-1)|$ and $s = e - h$.
3) The genie draws a random integer $\rho = \rho^{\text{left}}(i)$ uniformly from $\{1, 2, \ldots, h\}$. We think of ρ as a "random dither".
4) The genie shifts the window by ρ positions right. That is, ρ is added to both s and e.
   a) If the window falls off $\mathbf{z}$, that is, if $e > |\mathbf{z}|$, the genie chooses $\mathbf{y}_{\text{left}} = \epsilon$, the empty string. Otherwise, the genie continues to the next step.
5) The genie counts the number of '0' symbols in the window (i.e., the cardinality of $\{s \le i \le e | z_i = 0\}$),
6) If the count is at least $h/2$, the genie sets $\mathbf{y}_{\text{left}}$ to the remainder of $\mathbf{z}$ after deleting z_1 to z_e and then finishes by returning $\mathbf{y}_{\text{left}}$.
7) Otherwise, the genie shifts the window one frame right. That is, h is added to both s and e.
   a) If the window falls off $\mathbf{z}$, that is, if $e > |\mathbf{z}|$, the genie chooses $\mathbf{y}_{\text{left}} = \epsilon$, the empty string. Otherwise, the genie continues to the next step.
8) We set $\mathbf{y}_{\text{left}}$ to the remainder of $\mathbf{z}$ after deleting z_1 to z_e and then finish by returning $\mathbf{y}_{\text{left}}$.

The rationale of above procedure will become clearer after we explain Aladdin's algorithm. For now, note that it is well defined and does indeed satisfy the requirements stated previously. The reader should also keep in mind that getting into a substep is 'bad' with respect to Aladdin's ability to mimic the genie. That is, we would like the probability of entering substeps 1a, 4a, or 7a to be 'small'.

We must address one last point: how the paddings for blocks $i = 1$ and $i = \Phi$ are handled. The right padding for $i = 1$ and the left padding for $i = \Phi$ are as above. The left padding for $i = 1$ and the right padding for $i = \Phi$ (i.e., the edge padding) are given by random sampling from P_{padLeft} and P_{padRight}. These choices are coupled so that the genie and Aladdin always choose the same realizations for these edge paddings.

B. Aladdin parsing

Aladdin receives the vector $\mathbf{y}$, and as a preliminary step adds the edge padding on the left and right (both of which are coupled to the genie's choices). We denote the resulting vector $\mathbf{y}_{\text{pad}}$. Aladdin's parsing is given by $\text{mimicGenie}(\mathbf{y}_{\text{pad}}, n)$ where the recursive function $\text{mimicGenie}(\mathbf{z}, m)$ is defined by:

- If $m = n_0$, Aladdin returns $\mathbf{z}$. Otherwise,
- Aladdin builds $\mathbf{z}_{\text{I}}$ and $\mathbf{z}_{\text{II}}$ as follows, and then return $\text{mimicGenie}(\mathbf{z}_{\text{I}}, m-1)$ (which contains 2^{m-n_0-1} vectors), followed by $\text{mimicGenie}(\mathbf{z}_{\text{II}}, m-1)$ (which also contains 2^{m-n_0-1} vectors). Namely, Aladdin returns 2^{m-n_0} vectors.
- Let $\mathbf{z}_{\text{I}}$ be the left half of $\mathbf{z}$ and $\mathbf{z}_{\text{II}}$ be the right half of $\mathbf{z}$ (in case $|\mathbf{z}|$ is odd, $\mathbf{z}_{\text{I}}$ is longer than $\mathbf{z}_{\text{II}}$, by one bit). Then, Aladdin trims $\mathbf{z}_{\text{I}}$ and $\mathbf{z}_{\text{II}}$.
- Trimming $\mathbf{z}_{\text{I}}$ is the "mirror image" of trimming $\mathbf{z}_{\text{II}}$, which is done as follows:
  – Aladdin places a window of length h at the start of $\mathbf{z}_{\text{II}}$. That is, the window starts at $s = 0$ and ends at $e = h$. If in any stage of the algorithm the window "falls off $\mathbf{z}_{\text{II}}$", meaning that $e > |\mathbf{z}_{\text{II}}|$, Aladdin declares failure.
  – Aladdin randomly and uniformly chooses a random dither ρ' uniformly from $\{1, 2, \ldots, h\}$.
  – Aladdin shifts the window ρ positions right by adding ρ to both s and e.
  – Aladdin checks if the window contains at least $h/2$ '0' symbols. If it does, the process continues to the next step. If it does not, Aladdin moves the window one frame to the right by adding h to both s and e, and then repeats this bullet point.
  – Aladdin trims $\mathbf{z}_{\text{II}}$ by removing the first e symbols.

C. Connections between Aladdin and genie parsing

Let us compare Aladdin's parsing to that of the genie. First of all, note that both decoders use $2\Phi - 2$ random dithers during their runs (recall that we've denoted a random dither as ρ for the genie and ρ' for Aladdin). To help Aladdin match the genie, these dithers can be *coupled*. That is, for each choice of random dithers the genie makes we couple a unique choice of random dithers that Aladdin makes. The utility of this coupling is that, with high probability, both the genie and Aladdin return the same vector of DZP channel outputs $(\mathbf{y}^\star(i))_{i=1}^{\Phi}$.

Before describing the coupling, we note that, given the DZP parsing, the proof of Theorem 1 follows essentially the same steps as the main result in [10]. The steps will be detailed in a forthcoming longer version of this paper [22].

For brevity, we explain the coupling in terms of just two dithers. Consider the ρ that the genie chooses for padding $\mathbf{x}(i)$ from the left, for $i = \Phi/2 + 1$. We couple this ρ with the ρ' Aladdin chooses in the topmost part of the recursion, for producing $\mathbf{z}_{\mathrm{II}}$. Typically, the midpoint of $\mathbf{z}$ is in $\mathbf{d}^{\mathrm{midright}}(i-1)$ or $\mathbf{d}^{\mathrm{midleft}}(i-1)$. Aladdin adds the dither ρ to the window, and then shift it one frame right, until the number of zeros is large enough. By Lemma 2 we conclude that the number of zeros will typically not be large enough, until the window contains some part of $\mathbf{d}^{\mathrm{right}}(i-1)$. Consider the first time this happens, and set the Genie's ρ to the number of symbols from $\mathbf{d}^{\mathrm{right}}(i-1)$. One can think of the genie as having a 'shortcut' that avoids the previous steps Aladdin took. Both Aladdin and the Genie have the same window, at this point. If it contains enough zero symbols, they return the same padding. If it does not, they both shift it one frame right. At this stage, typically, the window will only contain symbols from $\mathbf{d}^{\mathrm{right}}(i-1)$, "dirty zeros". Thus, again by Lemma 2, Aladdin will typically stop at this stage, as the genie always does, and both will return the same left padding.

Appendix

A. Proof of Lemma 2

Let $0 < \delta < 1$ be a constant, dependent on the channel, that we will fix later. Recall that the expected length of an output corresponding to a single input is β, and define

$$k = \lceil h(1-\delta)/\beta \rceil \ . \tag{9}$$

We can think of the output $\mathbf{Y}$ as being manufactured as follows. We input the first bit (x) to the channel, then input k more bits (all x), and if the output up to this point has length less than $h+1$, inputting however many bits (all x) are needed in order for the output length to be at least $h+1$. Then, we possibly remove the first bit of the output, and set $\mathbf{Y}$ to the first h bits. We will call the output corresponding to the k input bits after the first input bit the *essential output*. Our proof hinges on showing that the following two events occur with very high probability: 1) all of the essential output is contained in $\mathbf{Y}$, and 2) the essential output has more than $h/2$ bits equal to x.

Denote by $\mathbf{Z} = \mathbf{Z}_1 \odot \mathbf{Z}_2 \odot \cdots \odot \mathbf{Z}_k$ the essential output, where $\mathbf{Z}_i$ is the output corresponding to input bit $i+1$. We find it easier to define bad events: event A occurs if the length of $\mathbf{Z}$ is at least $h-1$; event B occurs if $\mathbf{Z}$ contains at most $h/2$ bits equal to x. Clearly, if neither A nor B occur, the above good events occur[1], and we correctly guess x.

Now, let us choose

$$\delta = \min\left\{ \frac{\gamma}{2\cdot\alpha_{0|0}}, \frac{\gamma}{2\cdot\alpha_{1|1}}, \frac{1}{2} \right\} \ , \tag{10}$$

[1]Recall that the output due to the first input bit has length at most 2.

where γ is defined in (3). Note that, indeed, $0 < \delta < 1$. Let

$$h'_0 = \frac{2(\beta+1)}{\delta} - 1 > 1 \ , \tag{11}$$

where the inequality follows from (10). Assume that $h \geq h'_0$.

By Hoeffding's bound [23, Theorem 4.12],

$$P(A) \leq 2e^{\frac{-2k\left(\frac{h-1}{k}-\beta\right)^2}{4}} = 2e^{\frac{-k\left(\frac{h-1}{k}-\beta\right)^2}{2}} \overset{\text{(a)}}{\leq} 2e^{\frac{-\left(\frac{h(1-\delta)}{\beta}\right)\cdot\left(\frac{h-1}{k}-\beta\right)^2}{2}} \ ,$$

where (a) follows from (9). Noting the squared term on the RHS, we next show that

$$\frac{h-1}{k} > \frac{h-1}{\frac{h(1-\delta)}{\beta}+1} \geq \frac{\beta}{1-\delta/2} > \beta \ .$$

Indeed, the first inequality follows from (9), noting that $h-1$ is positive, since $h \geq h'_0 > 1$; the second follows from (11), recalling that $h \geq h'_0$; the third follows since $\delta > 0$. Thus, from the above two displayed equations we conclude that

$$P(A) < 2e^{\frac{-h\left(\frac{(1-\delta)}{\beta}\right)\cdot\left(\frac{\beta}{1-\delta/2}-\beta\right)^2}{2}} \ . \tag{12}$$

For event B, we use Hoeffding's inequality and (9) to get

$$P(B) \leq 2e^{\frac{-2k\left(\alpha_{x|x}-\frac{h}{2k}\right)^2}{4}} = 2e^{\frac{-k\left(\alpha_{x|x}-\frac{h}{2k}\right)^2}{2}} \leq 2e^{\frac{-\left(\frac{h(1-\delta)}{\beta}\right)\cdot\left(\alpha_{x|x}-\frac{h}{2k}\right)^2}{2}} \ .$$

Focusing on the squared term on the RHS, we now prove that

$$\alpha_{x|x} \geq \frac{\beta/2}{1-\delta} \geq \frac{h}{2k} \ .$$

The second inequality follows easily from (9). For the first inequality, first recall that $\beta = \alpha_{x|x} + \alpha_{1-x|x}$, by (1). Thus, it suffices to prove that $\frac{\alpha_{x|x}-\alpha_{1-x|x}}{2} \geq \delta\alpha_{x|x}$. By (3), this will follow if we prove that $\gamma \geq \delta\alpha_{x|x}$, which holds by (10). Thus, from the above two displayed equations we conclude that

$$P(B) \leq 2e^{\frac{-\left(\frac{h(1-\delta)}{\beta}\right)\cdot\left(\alpha_{x|x}-\frac{\beta/2}{1-\delta}\right)^2}{2}} = 2e^{\frac{-\left(\frac{h(1-\delta)}{\beta}\right)\cdot\left(\frac{\frac{\alpha_{x|x}-\alpha_{1-x|x}}{2}-\delta\alpha_{x|x}}{1-\delta}\right)^2}{2}} \ .$$

Slightly refining the above arguments, we get from (1), (3), and (10), that

$$\frac{\alpha_{x|x}-\alpha_{1-x|x}}{2} \geq \gamma \geq \frac{\gamma}{2} \geq \delta\alpha_{x|x} \ .$$

Thus, from the above two displayed equations we get that

$$P(B) \leq 2e^{\frac{-\left(\frac{h(1-\delta)}{\beta}\right)\cdot\left(\frac{\gamma-\gamma/2}{1-\delta}\right)^2}{2}} = 2e^{\frac{-h\left(\frac{1-\delta}{\beta}\right)\cdot\left(\frac{\gamma/2}{1-\delta}\right)^2}{2}} \ . \tag{13}$$

In light of (12) and (13), let us define

$$c_0 \triangleq \frac{1}{2}\min\left\{ \underbrace{\frac{\left(\frac{(1-\delta)}{\beta}\right)\cdot\left(\frac{\beta}{1-\delta/2}-\beta\right)^2}{2}}_{c'_0}, \underbrace{\frac{\left(\frac{1-\delta}{\beta}\right)\cdot\left(\frac{\gamma/2}{1-\delta}\right)^2}{2}}_{c''_0} \right\}$$

and take $h_0 \geq h'_0$ large enough such that for all $h \geq h_0$,

$$2e^{-h\cdot c'_0} + 2e^{-h\cdot c''_0} \leq e^{-h\cdot c_0} \ .$$

That is, $h_0 \triangleq \max\{h'_0, \ln(4)/c_0\}$.

References

[1] R. Gallager, “Sequential decoding for binary channels with noise and synchronization errors,” 1961, Lincoln Lab Group Report.

[2] R. L. Dobrushin, “Shannon’s theorems for channels with synchronization errors,” *Problemy Peredachi Informatsii*, vol. 3, no. 4, pp. 18–36, 1967.

[3] M. C. Davey and D. J. MacKay, “Reliable communication over channels with insertions, deletions, and substitutions,” *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 687–698, 2001.

[4] M. Mitzenmacher, “A survey of results for deletion channels and related synchronization channels,” *Probability Surveys*, vol. 6, pp. 1–33, 2009.

[5] Y. Kanoria and A. Montanari, “Optimal coding for the binary deletion channel with small deletion probability,” *IEEE Trans. Inform. Theory*, vol. 59, no. 10, pp. 6192–6219, 2013.

[6] M. Rahmati and T. M. Duman, “Upper bounds on the capacity of deletion channels using channel fragmentation,” *IEEE Trans. Inform. Theory*, vol. 61, no. 1, pp. 146–156, 2015.

[7] J. Castiglione and A. Kavcic, “Trellis based lower bounds on capacities of channels with synchronization errors,” in *Information Theory Workshop*. Jeju, South Korea: IEEE, 2015, pp. 24–28.

[8] M. Cheraghchi, “Capacity upper bounds for deletion-type channels,” *Journal of the ACM (JACM)*, vol. 66, no. 2, p. 9, 2019.

[9] I. Tal, H. D. Pfister, A. Fazeli, and A. Vardy, “Polar codes for the deletion channel: Weak and strong polarization,” in *Proc. IEEE Int. Symp. Inform. Theory*, 2019, pp. 1362–1366.

[10] ——, “Polar codes for the deletion channel: weak and strong polarization,” 2020, preprint arXiv:1904.13385v2.

[11] R. Wang, R. Liu, and Y. Hou, “Joint successive cancellation decoding of polar codes over intersymbol interference channels,” 2014, preprint arXiv:1404.3001.

[12] R. Wang, J. Honda, H. Yamamoto, R. Liu, and Y. Hou, “Construction of polar codes for channels with memory,” in *2015 IEEE Information Theory Workshop*, October 2015, pp. 187–191.

[13] E. K. Thomas, V. Y. F. Tan, A. Vardy, and M. Motani, “Polar coding for the binary erasure channel with deletions,” *IEEE Communications Letters*, vol. 21, no. 4, pp. 710–713, April 2017.

[14] K. Tian, A. Fazeli, A. Vardy, and R. Liu, “Polar codes for channels with deletions,” in *55th Annual Allerton Conference on Communication, Control, and Computing*, 2017, pp. 572–579.

[15] K. Tian, A. Fazeli, and A. Vardy, “Polar coding for deletion channels: Theory and implementation,” in *IEEE International Symposium on Information Theory*, 2018, pp. 1869–1873.

[16] ——, “Polar coding for deletion channels,” 2018, submitted to IEEE Trans. Inform. Theory.

[17] Y. Li and V. Y. F. Tan, “On the capacity of channels with deletions and states,” 2019, preprint arXiv:1911.04473.

[18] E. Şaşoğlu and I. Tal, “Polar coding for processes with memory,” *IEEE Trans. Inform. Theory*, vol. 65, no. 4, pp. 1994–2003, April 2019.

[19] B. Shuval and I. Tal, “Universal polarization for processes with memory,” 2018, preprint arXiv:1811.05727v1.

[20] H. Koremura and H. Kaneko, “Successive cancellation decoding of polar codes for insertion/deletion error correction,” in *Proc. IEEE Int. Symp. Inform. Theory (ISIT’2019)*, Paris, France, 2019, pp. 1357–1361.

[21] B. Shuval and I. Tal, “Fast polarization for processes with memory,” *IEEE Trans. Inform. Theory*, vol. 65, no. 4, pp. 2004–2020, April 2019.

[22] H. D. Pfister and I. Tal, “Polar codes for channels with insertions, deletions, and substitutions,” arXiv preprint in preparation.

[23] M. Mitzenmacher and E. Upfal, *Probability and Computing: Randomizition and Probabilistic Techniques in Algorithms and Data Analysis*, 2nd ed. Cambridge, UK: Cambridge University Press, 2005.