Polar Codes for Channels with Insertions, Deletions, and Substitutions

Henry D. Pfister\textsuperscript{1} \quad Ido Tal\textsuperscript{2}

\textsuperscript{1}Duke \quad \textsuperscript{2}Technion
Big picture first

- Channel has constant insertion/deletion/substitution probabilities
  - These probabilities do not change with the codeword length
- Fix a hidden-Markov input distribution\(^1\)
- Code rate converges to mutual information rate
- can achieve capacity using a sequence of input distributions
- Error probability decays like \(2^{-\Lambda \nu'}\), where \(\nu' < \nu \leq \frac{1}{3}\) and \(\Lambda\) is the codeword length
- Decoding complexity is at most \(O(\Lambda^{1+3\nu})\)

---

\(^1\)i.e., a function of an aperiodic, irreducible, finite-state Markov chain
Big picture first

- Channel has *constant* insertion/deletion/substitution probabilities
  - These probabilities do not change with the codeword length
- Fix a hidden-Markov input distribution\(^1\)
- Code rate converges to mutual information rate
  - \(\implies\) can achieve capacity using a sequence of input distributions
- Error probability decays like \(2^{-\Lambda \nu'}\), where \(\nu' < \nu \leq \frac{1}{3}\) and \(\Lambda\) is the codeword length
- Decoding complexity is at most \(O(\Lambda^{1+3\nu})\)
- **Key ideas:**
  - Polarization operations defined for *trellises*
  - Polar codes modified to have guard bands of 0’s and 1’s

\(^1\)i.e., a function of an aperiodic, irreducible, finite-state Markov chain
Relation to our previous work on deletion channels

In our previous\(^2\) paper on deletion channels

- Use of trellises to capture deletion and polar transforms
- Proof of weak polarization for “vanilla” polar codes
- For strong polarization, guard bands must be added

Generalization to IDS channel

- First two bullets generalize naturally to IDS channel
- Not straightforward:
  - For strong polarization, different guard bands must be added
  - Our analysis uses two players: Genie who processes guard bands “perfectly”, and Aladdin, who tries to mimic the genie

\(^2\)I. Tal, H. D. Pfister, A. Fazeli, A. Vardy, “Polar Codes for the Deletion Channel: Weak and Strong Polarization”
The channel model

- Input alphabet: $\mathcal{X} = \{0, 1\}$
- Output alphabet: $\mathcal{Y} \subset \mathcal{X}^*$
  - $\mathcal{Y}$ is a finite collection of binary strings, possibly of different lengths
  - $\epsilon$, the empty string, is a valid output symbol
- Probability law, single input symbols:
  - For $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, the probability law is $P(y|x)$
- Probability law, multiple input symbols:
  - Let $Y_i$ be the output corresponding to $X_i$, for $1 \leq i \leq N$
  - The output corresponding to $X_1, X_2, \ldots, X_N$ is $Y_1 \circ Y_2 \circ \cdots \circ Y_N$, where $\circ$ denotes concatenation
  - Not $Y_1, Y_2, \ldots, Y_N$ (we don’t see the commas)

---

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

Important example

- $\mathcal{X} = \{0, 1\}$, $\mathcal{Y} = \{\epsilon, 0, 1, 00, 01, 10, 11\}$
- Deletion: $P(\epsilon|x) = p_d$
- Substitution: $P(\overline{x}|x) = p_s$
- Insertion: $P(0x|x) = P(1x|x) = \frac{p_i}{2}$
- No error: $P(x|x) = 1 - p_d - p_s - p_i$

Underlying assumptions

- The channel is memoryless
- Advantage of the input at the output:
  - For input $x$, let $\alpha_{0|x}$ ($\alpha_{1|x}$) be the expected number of 0 (1) symbols at the output
  - We require: $\alpha_{0|0} > \alpha_{1|0}$ and $\alpha_{1|1} > \alpha_{0|1}$
- Expected output length independent of input:

$$\beta = \alpha_{0|0} + \alpha_{1|0} = \alpha_{0|1} + \alpha_{1|1}$$
The code rate of our scheme approaches

\[ I(X; Y) = \lim_{N \to \infty} \frac{1}{N} H(X) - \lim_{N \to \infty} \frac{1}{N} H(X|Y), \]

- **X** = \((X_1, \ldots, X_N)\) is hidden-Markov input
- **Y** is the channel output
Theorem (Strong polarization)

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 \( \Lambda \) is the blocklength. For any \( 0 < \nu' < \nu \) and sufficiently large blocklength \( \Lambda \), the probability of decoding error is at most \( 2^{-\Lambda \nu'} \).
Weak polarization

- Fix a regular hidden-Markov input distribution
- Let $X_1, \ldots, X_N$ be inputs, where $N = 2^n$
- Let $Y = Y_1 \odot Y_2 \odot \cdots \odot Y_N$ be the corresponding output
- Let $U_1, U_2, \ldots, U_N$ be the polar transform of $X_1, X_2, \ldots, X_N$
- Can easily adapt the proof from the deletion-only paper to prove

**Theorem**

*For any $\epsilon > 0$,*

$$\lim_{N \to \infty} \frac{1}{N} \left| \left\{ i \in [N] \mid H(U_i \mid U_{i-1}, Y) \in [\epsilon, 1 - \epsilon] \right\} \right| = 0$$
Strong polarization — first attempt

- Fix a regular hidden-Markov input distribution
- Let $X_1, \ldots, X_N$ be inputs, where $N = 2^n$
- Let $X(1), X(2), \ldots, X(\Phi)$ be the inputs, separated into $\Phi$ blocks, each of length $N/\Phi$
- Let $Y(1), Y(2), \ldots, Y(\Phi)$ be the corresponding output blocks
- Let $U_1, U_2, \ldots, U_N$ be the polar transform of $X_1, X_2, \ldots, X_N$
- We can adapt the proof from the deletion-only paper to prove strong polarization, for output punctuated into blocks

That is, for appropriately chosen $\nu$ and $\Phi$,

$$
\lim_{N \to \infty} \frac{1}{N} \left| \left\{ i \in [N] \mid Z(U_i|U_i^{i-1}, Y(1), \ldots, Y(\Phi)) < 2^{-N^\nu} \right\} \right|
= 1 - \lim_{N \to \infty} \frac{1}{N} H(X_1^N|Y_1^N)
$$
Strong polarization — first attempt

- If we had a genie that could punctuate the output

\[ Y(1) \odot Y(2) \odot \cdots \odot Y(\Phi) \]

into

\[ Y(1), Y(2), Y(\Phi) \]

we would have strong polarization
Needed: just the right genie

- On the decoding side we have a mere-mortal, Aladdin
- Aladdin gets the non-punctuated output

\[ Y(1) \circ Y(2) \circ \cdots \circ Y(\Phi) \]

- Our “genie” will be a mathematical construct
- It must have two key qualifications
- Strong enough: using the genie gives us strong polarization
- Not too strong: Aladdin can, with high probability, mimic the genie
Needed: just the right genie

A genie that punctuates output into $\mathbf{Y}(1), \mathbf{Y}(2), \ldots, \mathbf{Y}(\Phi)$ is

- Strong enough (leads to strong polarization)
- Too strong (Aladdin can’t mimic)
Needed: just the right genie

Key idea:

- Split $\mathbf{x} = x_1, x_2, \ldots, x_N$ into $\Phi$ blocks, each of length $N/\Phi = 2^{n_0}$,
  \[ \mathbf{x} = \mathbf{x}(1) \odot \mathbf{x}(2) \odot \cdots \odot \mathbf{x}(\Phi) \]

- Instead of sending $\mathbf{x}$ over the channel, we send $g(\mathbf{x})$, in which the $\mathbf{x}(i)$ are interspaced by “guard bands”

- The genie will remove some part of each guard band from the corresponding output, and punctuate into $\Phi$ blocks

- Aladdin will be able to do the same, with high probability
Needed: just the right genie

Need to make sure that:

- Adding the guard band does not change the code rate by much
  - Length of guard bands must be sub-linear
- Trimming only part of the guard band does not change the block entropy by much
  - guard bands must be “simple”
Guard bands

Denote $\mathbf{x} = \mathbf{x}_I \odot \mathbf{x}_II$, where $\mathbf{x}_I$ and $\mathbf{x}_II$ are the left and right halves of $\mathbf{x}$

Define $g(\mathbf{x})$ recursively: for a vector $\mathbf{x}$ of length $2^n$,

$$g(\mathbf{x}) \triangleq \begin{cases} g(\mathbf{x}_I) \odot g_n \odot g(\mathbf{x}_II) & \text{if } n > n_0, \\ \mathbf{x} & \text{if } n \leq n_0 \end{cases} \quad (1)$$

where

$$g_n \triangleq 0(\ell_{n_0}) \odot \underbrace{1(\ell_n) \odot 1(\ell_n) \odot 0(\ell_{n_0})}_\text{g}.$$

and

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

$\xi \in (0, 1/2)$ a ‘small’ constant (determined by the parameters in the Theorem)
Genie decoding

- Denote the input to the channel as

\[ x(1) \circ g(1) \circ x(2) \circ g(2) \circ \cdots \circ g(\Phi - 1) \circ x(\Phi) \]

- Denote the corresponding output as

\[ y = y(1) \circ d(1) \circ y(2) \circ d(2) \circ \cdots \circ d(\Phi - 1) \circ y(\Phi) \]

- The genie will parse this into blocks, \( y^*(1), y^*(2), \ldots, y^*(\Phi) \) (and throw away some symbols)

- Consider the segment \( d(i - 1) \circ y(i) \circ d(i) \)

- For \( 1 < i < \Phi \), the genie will produce

\[ y^*(i) = y_{\text{left}}(i) \circ y(i) \circ y_{\text{right}}(i) \]

where

- \( y_{\text{left}}(i) \) is a suffix of \( d(i - 1) \)
- \( y_{\text{right}}(i) \) is a prefix of \( d(i) \)
Genie decoding – abridged

Producing $y_{\text{left}}(i)$ (abridged to “high probability” case)

- Denote

$$d(i-1) = d_{\text{left}}(i-1) \odot d_{\text{midleft}}(i-1) \odot d_{\text{midright}}(i-1) \odot d_{\text{right}}(i-1)$$

- We only consider

$$d_{\text{midright}}(i-1) \odot d_{\text{right}}(i-1)$$

- For a properly defined $h$:

- Place a window of length $h$ at the end of $d_{\text{midright}}(i-1)$

\[
\begin{array}{c}
\text{mostly ones} \\
\{11110110111011101110110\}
\end{array}
\quad
\begin{array}{c}
\text{mostly zeros} \\
\{100000010000\}
\end{array}
\]
Genie decoding – abridged

Producing $y_{\text{left}}(i)$ (abridged to “high probability” case)

- Shift the window $\rho$ places right, where $\rho$ chosen uniformly from $\{1, 2, \ldots, h\}$

- Does the window contain more zeros than ones?

- If so, $y_{\text{left}}(i)$ is everything to the right of the window
Genie decoding – abridged

Produce \( y_{\text{left}}(i) \) (abridged to “high probability” case)

\[
\begin{align*}
\text{d}_{\text{midright}}(i-1) &\quad \text{d}_{\text{right}}(i-1) \\
11110110111011101111 &\quad 10.100000010000,
\end{align*}
\]
mostly ones
mostly zeros

- Does the window contain more zeros than ones?
- If not, shift the window \( h \) place to the right

\[
\begin{align*}
\text{d}_{\text{midright}}(i-1) &\quad \text{d}_{\text{right}}(i-1) \\
1111011011101110111110 &\quad 100000100000,
\end{align*}
\]
mostly ones
mostly zeros

- \( y_{\text{left}}(i) \) is everything to the right of the window
- Producing \( y_{\text{right}}(i) \): similar (mirror)…
Aladdin decoding – abridged

- Aladdin gets \( y \), and must produce the same \( y^*(i) \) as the genie.
- He does so recursively, splitting \( y \) into two blocks, then each of these two blocks into two more blocks...
- We show the first step in the recursion.
- We will show how to find the right block (left block is similar, up to mirroring).
- First, choose the middle index in \( y \).
- Typically, this middle index is either in \( d_{\text{midleft}}(N/2 - 1) \) or \( d_{\text{midright}}(N/2 - 1) \).

\[
\begin{align*}
\text{mostly ones} & & \text{mostly ones} & & \text{mostly zeros} \\
110111110111111110 & & 1111011011101110111110 & & 100000010000,
\end{align*}
\]
Aladdin decoding – abridged

- $d_{\text{midleft}}(N/2-1)$: mostly ones
- $d_{\text{midright}}(N/2-1)$: mostly ones
- $d_{\text{right}}(N/2-1)$: mostly zeros

- Aladdin now picks $\rho \in \{1, 2, \ldots, h\}$, and shift the index $\rho$ places to the right
Aladdin decoding – abridged

\[ d_{\text{midleft}}(N/2-1) \]
\[ \begin{array}{cccccccccccccccccc}
\underbrace{1101111101111111110111} \\
\text{mostly ones}
\end{array}, \begin{array}{cccccccccccccccccc}
\underbrace{1111011011101110111110} \\
\text{mostly ones}
\end{array}, \begin{array}{cccccccccccccccccc}
\underbrace{100000010000} \\
\text{mostly zeros}
\end{array} \]

- Aladdin now opens a window of width \( h \), whose left is at the index previously picked

\[ d_{\text{midleft}}(N/2-1) \]
\[ \begin{array}{cccccccccccccccccccccccccc}
\underbrace{11011111011111111101111} \\
\text{mostly ones}
\end{array}, \begin{array}{cccccccccccccccccccccccccc}
\underbrace{1111011011101110111110} \\
\text{mostly ones}
\end{array}, \begin{array}{cccccccccccccccccccccccccc}
\underbrace{100000010000} \\
\text{mostly zeros}
\end{array} \]
Aladdin decoding – abridged

As long as the window contains more ones than zeros, we shift it by $h$ places right, and try again.

$\text{d}_{\text{midleft}}(N/2-1)$ mostly ones

$\text{d}_{\text{midright}}(N/2-1)$ mostly ones

$\text{d}_{\text{right}}(N/2-1)$ mostly zeros
As long as the window contains more ones than zeros, we shift it by $h$ places right, and try again.
As long as the window contains more ones than zeros, we shift it by $h$ places right, and try again.

- Mostly ones
- Mostly ones
- Mostly zeros
Aladdin decoding – abridged

As long as the window contains more ones than zeros, we shift it by $h$ places right, and try again.
Aladdin decoding – abridged

$\text{d}_{\text{midleft}}(N/2-1)$

11011111011111110111

mostly ones

$\text{d}_{\text{midright}}(N/2-1)$

1110110110111011110

mostly ones

$\text{d}_{\text{right}}(N/2-1)$

100000010000

mostly zeros

▶ As long as the window contains more ones than zeros, we shift it by $h$ places right, and try again
Aladdin decoding – abridged

As long as the window contains more ones than zeros, we shift it by $h$ places right, and try again.

$$d_{\text{midleft}}(N/2−1) \begin{cases} 1101111101111111101111 \text{ mostly ones} \\ 111101101110111110 \text{ mostly ones} \end{cases}$$

$$d_{\text{right}}(N/2−1) \begin{cases} 100000010000 \text{ mostly zeros} \\ 100000010000 \text{ mostly zeros} \end{cases}$$
Aladdin decoding – abridged

As long as the window contains more ones than zeros, we shift it by $h$ places right, and try again.
As long as the window contains more ones than zeros, we shift it by $h$ places right, and try again.
Aladdin decoding – abridged

As long as the window contains more ones than zeros, we shift it by $h$ places right, and try again.
Aladdin decoding – abridged

As long as the window contains more ones than zeros, we shift it by \( h \) places right, and try again.

The right block is everything to the right of the window.