Lossless Source Coding with Polar Codes

Harm S. Cronie
School of Computer and Communication Sciences
EPFL
Lausanne, Switzerland
email: harm.cronie@epfl.ch

Satish Babu Korada
Electrical Engineering Department
Stanford University
email: korada@stanford.edu

***Abstract*—In this paper lossless compression with polar codes is considered. A polar encoding algorithm is developed and a method to design the code and compute the average compression rate for finite lengths is given. It is shown that the scheme achieves the optimal compression rate asymptotically. Furthermore, the proposed scheme has a very good performance at finite lengths. Both the encoding and decoding operations can be accomplished with complexity $O(N \log N)$ where N denotes the length of the code.**

***Index Terms*—lossless source coding, polar codes, polarization.**

I. Introduction

Error-correcting codes can be used for the lossless compression of sources [1]. These schemes are usually more robust in a setting where compressed data is transmitted on a channel which induces errors. Several schemes based on sparse graph codes have been introduced where encoding and/or decoding algorithms are based on belief propagation (BP). Several of these schemes perform well in a practical setting. However, optimality seems hard to proof for these schemes.

Polar codes, introduced by Arıkan [2], are a family of codes that achieve the capacity of a large class of channels using low-complexity encoding and decoding algorithms. The complexity of these algorithms scales as $O(N \log N)$, where N is the block length of the code. Recently [3], [4] it has been shown that, in addition to being capacity-achieving for channel coding, polar-like codes are also optimal for lossy and lossless source coding as well as multi-terminal problems like the Slepian-Wolf, the Wyner-Ziv and the Gelfand-Pinsker problems.

In [4] the argument that polar codes are optimal in a lossless setting is based on mapping the source coding problem to a channel coding problem. The algorithm presented in [4] can sometimes fail to encode correctly and when it fails the source vector is stored as it is. In this paper we investigate the use of polar codes for lossless compression in more detail and show that we can modify the algorithm to take care of the failures. This results in an algorithm that is amenable to analysis and that performs very well in a practical finite length setting.

Harm Cronie is supported by Grant 228021-ECCSciEng of the European Research Council.

Satish Korada is supported by a fellowship form the Swiss National Science Foundation.

In a channel coding setting, polar codes seem to be competitive with other coding schemes such as low-density parity-check codes only for very large block lengths. The reason for this is the sequential nature of the decoding algorithm and the high amount of error propagation. However, we will show that in a lossless source coding setting, there is hardly an issue of error propagation and the source coding performance is close to the theoretical limit for moderate block lengths.

The outline is as follows. In Section II we introduce the notation used throughout the paper. In Section III we describe the encoding and decoding algorithm. In Section IV we give a proof of optimality of our algorithm. In Section V we discuss code design and analysis. Furthermore, simulation results are given which show that our scheme has very good finite-length performance.

II. Definitions and Notation

We denote random variables by upper case letters, e.g., U, and use the corresponding lower case letters, e.g., u, to denote their realizations. Let $\bar{U}$ denote the random vector $(U_0, \ldots, U_{N-1})$. For any finite set of integers F, $|F|$ denotes its cardinality. Furthermore, F^c denotes the complement of F with respect to the set $\{0, \ldots, N-1\}$. Let U_F denote $(U_{i_1}, \ldots, U_{i_{|F|}})$, where $\{i_k \in F : i_k \leq i_{k+1}\}$. In a similar way we define u_F. Let U_i^j denote the random vector $(U_i, \ldots, U_j)$. The binary field is denoted by $\mathbb{F}_2$. Let $\text{Ber}(p)$ denote a Bernoulli random variable X taking values in $\mathbb{F}_2$ with $\Pr(X = 1) = p$. We denote the binary entropy function by $h_2(p)$. The binary symmetric channel with flip probability p is denoted by BSC(p). We denote by $1(A)$ the indicator function which is 1 when a statement A is true and 0 otherwise. The logarithms in this paper always refer to $\log_2$.

At the core of polarization for channel and source coding is a polarizing transform and in this paper we consider a transform that is based on the following matrix

$$G_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}.$$

The transform itself is constructed as

$$G_2^{\otimes n} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}^{\otimes n},$$

where $\otimes$ denotes the Kronecker product. Polar codes based on $G^{\otimes n}$ where G is an $\ell \times \ell$ matrix were analyzed in [5]. Here, we restrict ourselves to codes based on G_2.

Definition 1 (Polar Code): The polar code $\mathtt{C}_N(F, u_F)$ of length $N = 2^n$, defined for any $F \subseteq \{0, \ldots, N-1\}$ and $u_F \in \mathbb{F}_2^{|F|}$, is a linear code given by

$$\mathtt{C}_N(F, u_F) = \{x_0^{N-1} = u_0^{N-1} G_2^{\otimes n} : u_{F^c} \in \mathbb{F}_2^{|F^c|}\}.$$

In other words the code $\mathtt{C}_N(F, u_F)$ is constructed by fixing the coordinates with indices in F to u_F and varying the indices in F^c over all possible values. Let us refer to the set F as the frozen set and the indices belonging to it as the frozen indices.

III. Polar Source Coding and Decoding

Consider a sequence of $N = 2^n$ i.i.d. source symbols X_0^{N-1} generated from a Ber(p) source. As in channel coding, the two main ingredients of polar source coding are *sequential encoding* and *polarization*. Instead of encoding X_0^{N-1} directly we consider encoding the sequence

$$U_0^{N-1} = X_0^{N-1} G_2^{\otimes n}.$$

Since the transform is invertible, we have

$$H(U_0^{N-1}) = H(X_0^{N-1}) = N h_2(p).$$

Furthermore, we have the chain rule of entropy

$$N h_2(p) = H(U_0^{N-1}) = \sum_{i=0}^{N-1} H(U_i \,|\, U_0^{i-1}). \qquad (1)$$

Polarization of entropy would imply that the terms on the right of (1) become either close to 1 or close to 0 when we increase N. Now, if we encode U_0^{N-1} in a sequential fashion from 0 to $N-1$, we do not need to encode the u_i for which $H(U_i \,|\, U_0^{i-1})$ is close to 0 since they can be estimated from the source model given U_0^{i-1}. The u_i for which $H(U_i \,|\, U_0^{i-1})$ becomes close to 1 are stored. The main difference with polar coding for a channel coding setting is that the encoder can perform the estimation and compare with the realization u_i. An appropriate action can be taken when the estimate turns out to be wrong. From (1) it follows that the fraction of bits whose conditional entropy approaches 1 is $h_2(p)$ which results in a compression rate of $h_2(p)$.

The encoder is defined by a set of frozen indices F and each $i \in F$ corresponds to a U_i for which $H(U_i \,|\, U_0^{i-1})$ is close to 1. The frozen set F and the source model are assumed to be known to the encoder as well as to the decoder.

Let $\bar{x}$ denote the realization of the source sequence. Now, we consider an encoder which performs the following steps. First, the encoder computes $\bar{u} = \bar{x} G_2^{\otimes n}$. Second, for each $i \in F^c$ the encoder computes

$$P(U_i = 1 | U_0^{i-1} = u_0^{i-1}), \qquad (2)$$

and estimates the value of u_i as

$$\hat{u}_i = \text{round}\left(P\left(U_i = 1 | U_0^{i-1} = u_0^{i-1}\right)\right).$$

Third, a set of indices T is defined as

$$T = \{i : i \in F^c, \hat{u}_i \neq u_i\}.$$

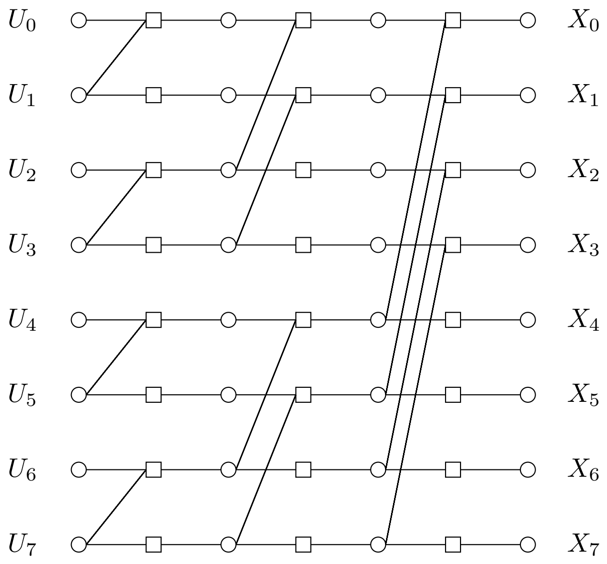

Fig. 1: Example of a factor graph for a polar code with block length of $2^3 = 8$.

The set T contains all indices for which the estimation of the corresponding u_i would be incorrect. Finally, the output of the encoder is given by (u_F, T).

In a similar way we construct a decoder. The input of the decoder is given by (u_F, T). The decoder performs the following steps. First, the decoder computes for $i \in F^c$ an estimate of u_i. In case $i \in T$ the value of the estimate is flipped. Once u_{F^c} has been recovered, the encoder knows $\bar{u}$ and can recover $\bar{x}$ as

$$\bar{x} = \bar{u}\left(G_2^{\otimes n}\right)^{-1} = \bar{u} G_2^{\otimes n}.$$

The encoding and decoding operations both involve the matrix multiplication with $G_2^{\otimes n}$ and the successive cancellation (SC) decoding algorithm in which (2) is computed for each $i \in F^c$. The computation of (2) can be performed in an efficient way by considering the factor graph between $\bar{u}$ and $\bar{x}$. Figure 1 shows an example of such a factor graph for $n = 3$. From [2] we know that both the matrix multiplication and the computation of probabilities can be accomplished with complexity $O(N \log N)$.

We will design and analyze a practical algorithm for compression in Section V where we show how to choose F in an optimal way and how to store T. In the next section we show that there exists a suitable choice of F for which the compression rate approaches the source entropy.

IV. Proof of Optimality

In this section we show that there exists a choice of F which asymptotically achieves the optimal compression rate. The proof is based on mapping the problem to a channel decoding problem.

Theorem 2 (Optimality for Lossless Compression): Let X be a Ber(p) random variable. For any rate $R > h_2(p)$ there exists a sequence of polar codes of length N and rate $R_N < R$ so that the source can be compressed losslessly with rate R_N.

Proof: Let $\epsilon > 0$, $R = h_2(p) + \epsilon$, and $0 < \beta < \frac{1}{2}$. Let F_N be a sequence of sets such that $\frac{|F_N|}{N} \le h_2(p) + \frac{\epsilon}{2}$. The probability of error $P_B(F_N)$, when using a code $\mathtt{C}_N(F_N, u_{F_N})$ for the BSC(p) under SC decoding is at most $O(2^{-N^\beta})$. Such a sequence of polar code ensembles are known to exist from [2], [6].

Let $\bar{x}$ be a sequence of N i.i.d. realizations of the source. Consider the source encoder based on the frozen set F_N. The encoder maps the source vector $\bar{x}$ to (u_{F_N}, T) where $u_{F_N} = (\bar{x} G_2^{\otimes n})_{F_N}$ and T is the set of erroneous indices as mentioned in the previous section. Therefore the average rate of compression is given by

$$\frac{1}{N}\left(|F_N| + \mathbb{E}[|T| \log N]\right) \le \frac{1}{N}\left(|F_N| + \Pr(|T| > 0) N \log N\right),$$

where the factor $\log N$ comes from the fact that we require $\log N$ bits to specify an index.

We will now show that $\Pr(|T| > 0) = P_B(F_N)$ by using the following mapping between the source compression and the channel coding problems. The task of recovering $\bar{x}$ from u_{F_N} can be formulated as the following channel decoding problem for the BSC(p). Let $\bar{x}$ be the input of a channel whose output is always $\bar{0}$. The noise during the transmission, denoted by $\bar{z}$, is given by $\bar{z} = \bar{x}$ ($\bar{0} = \bar{x} \oplus \bar{z}$). Since $\bar{x}$ is an i.i.d. Ber(p) source, it implies that the noise $\bar{z}$ is also i.i.d. Ber(p) vector. Now, consider recovering $\bar{x}$ from u_{F_N}. By construction, $\bar{x} \in \mathtt{C}_N(F_N, u_{F_N})$. Therefore, reconstructing $\bar{x}$ from $\bar{y} = \bar{0}$ is equivalent to decoding at the output of the BSC(p).

The probability $P(U_i \,|\, U_0^{i-1})$ is equal to the posterior probability $P(U_i \,|\, U_0^{i-1}, \bar{Y} = \bar{0})$ in the channel coding context. The SC algorithm indeed estimates the bits in F^c based on these posteriors. Therefore, if there is an error in the channel coding context it implies an error in the source coding context too and vice-versa. Hence,

$$\Pr(|T| > 0) = P_B(F_N).$$

Therefore, the required rate is at most

$$R_N \le \frac{|F_N| + P_B(F_N) N \log N}{N} \le \frac{|F_N|}{N} + O(2^{-(N)^\beta}),$$

which implies that for N sufficiently large, we have $R_N < R$. ∎

V. Encoder Design and Performance

In this section we consider the design of an encoder for finite lengths. Given a Ber(p) source one has to choose F and a method to encode the indices in T. The choice of F and the encoder for T determine the behavior of the encoding length L and we are interested in minimizing its expectation $\mathbb{E}[L]$. Encoding T is not difficult since the entropies corresponding to bits of F^c have low entropy and an encoder for T can easily be chosen to have a vanishing influence on $\mathbb{E}[L]$ for increasing block lengths. To encode T there are several options. First, we can choose not to use T at all and in case an encoding error occurs we can simply send the whole source block [4]. In this

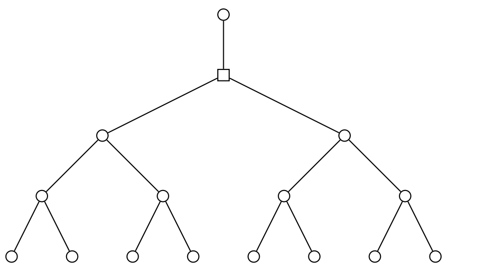

Fig. 2: Example of a computation tree for the computation of $\mathcal{E}_6$ with density evolution.

case the optimal way to choose F is not easy to estimate. In the following we consider a simple encoder for T which is amenable to finite length analysis and is good enough to achieve optimal performance asymptotically and has a good finite length performance.

A. An encoder for T

For each element in T we allocate $\log N$ bits. Since the size of T is at most N, $\log N$ bits will guarantee that we are able to encode all indices of T. Let $\mathcal{E}_i$ denote the failure probability of encoding bit U_i given the past U_0^{i-1}. We can express the expected encoding length $\mathbb{E}[L]$ as

$$\begin{aligned}\mathbb{E}[L] &= \mathbb{E}\left[|F| + \sum_{i \in F^c} 1(\hat{U}_i \ne U_i) \cdot \log N\right] \\ &= |F| + \log N \cdot \mathbb{E}\left[\sum_{i \in F^c} 1(\hat{U}_i \ne U_i)\right] \\ &= |F| + \log N \cdot \sum_{i \in F^c} \mathcal{E}_i,\end{aligned}$$

where we have made use of the linearity of expectation. Furthermore, the expected encoding rate is defined as

$$\mathbb{E}[R] = \frac{\mathbb{E}[L]}{N}.$$

Lemma 3: The expected encoding length $\mathbb{E}[L]$ is minimized when F is chosen as

$$F = \left\{ i : \mathcal{E}_i \ge \frac{1}{\log N} \right\}.$$

Proof: For each i we have the choice of including it in F or F^c. In case we include it in F it will contribute 1 bit to $\mathbb{E}[L]$ and in case we include it in F^c, it will contribute $\mathcal{E}_i \log N$ bits to $\mathbb{E}[L]$. Hence, to minimize the contribution to $\mathbb{E}[L]$ we include i in F in case $\mathcal{E}_i \log N \ge 1$. ∎

B. Computation of $\mathbb{E}[L]$

We can numerically compute $\mathbb{E}[L]$ as follows. The computation of $\mathcal{E}_i$ is equivalent to the computation of the error probabilities of the channels $W_N^{(i)}$ (mentioned in [2]) when the channel W is the BSC(p). The estimation of U_i for e.g. $i = 6$ and a code with $n = 3$ can be computed by BP decoding the tree given in Figure 2. Furthermore, the corresponding

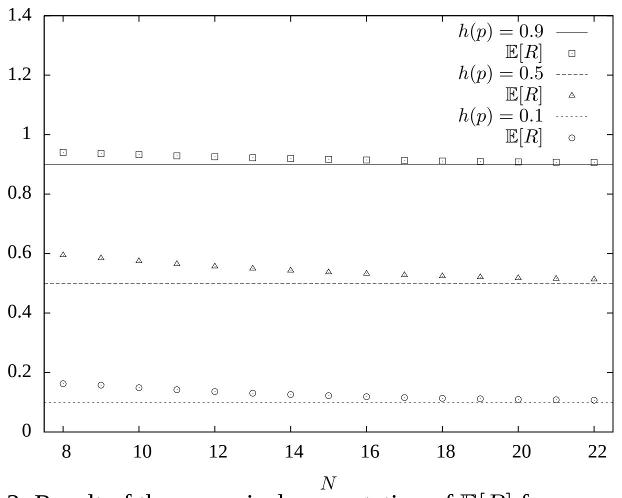

Fig. 3: Result of the numerical computation of $\mathbb{E}[R]$ for source entropies of 0.9, 0.5 and 0.1.

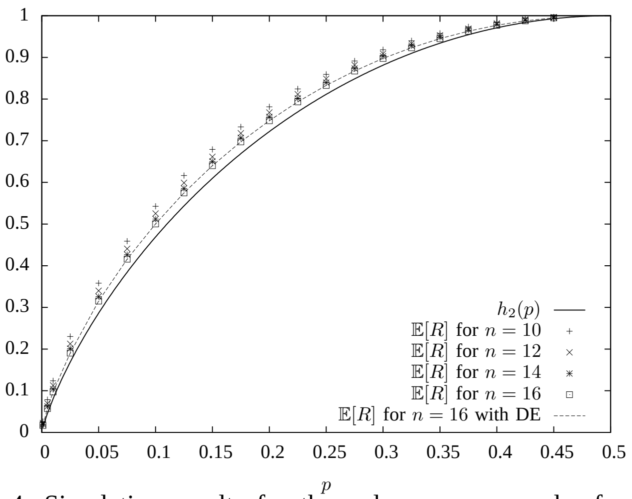

Fig. 4: Simulation results for the polar source encoder for $n = 10, 12, 14, 16$.

$\mathcal{E}_i$ can be efficiently computed using density evolution as mentioned in [7]. The leaf-variable nodes correspond to the original source random variables and depending on the binary expansion of i we observe variable or check nodes in the tree. The binary expansion of 6 is 110 and from the bottom to the top we observe variable nodes (1), variable nodes (1) and check nodes (0) in Figure 2. For density evolution, the leaf-variable nodes are initialized by a density of log-likelihood ratios (LLRs) corresponding to the source model. After a few density evolution iterations we obtain a density of LLRs from which we can compute $\mathcal{E}_i$. For the actual implementation of density evolution we refer to [8].

Figure 3 shows the expected encoding rate for a Ber(p) source with an entropy of 0.9, 0.5 and 0.1, respectively. Furthermore, n is chosen in the range $n = 8, \ldots, 22$. We observe that for increasing block lengths the average encoding rate approaches the theoretical entropy limit. Even for the smaller block lengths the gap between $\mathbb{E}[R]$ and the entropy limit is not that large.

C. Simulation results

Simulation results for an actual encoder are shown in Figure 4 for $n = 10, 12, 14, 16$ where 1000 encoding runs are simulated. The figure shows the expected encoding rate and we observe that the compression performance of the polar source encoder is close to the theoretical limit for the whole range of p. For $n = 16$ the required compression overhead is less than 5 percent.

For $n = 16$ the figure also shows the result of the computation of $\mathbb{E}[R]$ by density evolution. As expected, this curve matches with the simulation results. Hence one can use density evolution to compute the finite length performance of the encoder. This is an advantage since density evolution is in general faster and more accurate than a Monte Carlo simulation.

Although the decoding complexity scales as $O(\log N)$ per source bit, the encoder and decoder are fast. The reason for this is that the actual number of operations performed per source bits is about $\log N$. This makes our scheme quite competitive compared to other lossless compressions schemes based on sparse graph codes. The reason for this is that the complexity per source bit for these schemes is usually proportional to the number of iterations performed. In our case this quantity is $\log N$.

VI. Conclusion and Future Work

In this paper we have presented a variable rate lossless source coding scheme based on polar codes. As in a channel coding setting polar codes are asymptotically optimal for lossless source coding. However, unlike in the channel coding setting, we can ensure that there is no error propagation in the sequential encoder for the polar code. This benefits the finite length performance of the scheme.

We have shown how to design an encoder for finite lengths and how to compute its compression performance. The resulting finite length encoders have a performance close to the entropy limit while the encoding and decoding complexity are low.

The scheme can be extended in a straightforward manner to non-binary sources using polar codes for non-binary channels [9]. An alternative for non-binary sources would be a multi-level encoding approach which could have advantages in terms of encoding and decoding complexity. In this setting an extension to piecewise stationary sources is required since the properties of the source at a particular level will depend on the previous levels. Finally, the scheme might be extended to a joint source-channel coding setting. The latter setting is the case where the scheme is most relevant from a practical point of view.

References

[1] G. G. Caire, S. Shamai, and S. Verdu, “Lossless data compression with error correcting codes,” in *Proc. of the IEEE Int. Symposium on Inform. Theory*, 2003, pp. 22–26.

[2] E. Arıkan, “Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels,” *IEEE Transactions on Information Theory*, vol. 55, no. 7, pp. 3051–3073, 2009.
[3] S. B. Korada and R. Urbanke, “Polar codes are optimal for lossy source coding,” *submitted to IEEE Trans. Inform. Theory*, 2009.
[4] S. B. Korada, “Polar codes for channel and source coding,” Ph.D. dissertation, EPFL, Lausanne, Switzerland, July 2009.
[5] S. B. Korada, E. Şaşoğlu, and R. Urbanke, “Polar codes: Characterization of exponent, bounds, and constructions,” *accepted for publication in IEEE Transactions on Information Theory*, 2009.
[6] E. Arıkan and E. Telatar, “On the rate of channel polarization,” in *Proc. of the IEEE Int. Symposium on Inform. Theory*, Seoul, South Korea, July 2009, pp. 1493–1495.
[7] R. Mori and T. Tanaka, “Performance and Construction of Polar Codes on Symmetric Binary-Input Memoryless Channels,” in *Proc. of the IEEE Int. Symposium on Inform. Theory*, Seoul, South Korea, July 2009, pp. 1496–1500.
[8] T. Richardson and R. Urbanke, *Modern Coding Theory*. Cambridge University Press, 2008.
[9] E. Şaşoğlu, E. Telatar, and E. Arıkan, “Polarization for arbitrary discrete memoryless channels,” in *Proc. of the IEEE Inform. Theory Workshop*, Taormina, Italy, Oct. 2009, conference.