Joint Source-Channel Decoding of Polar Codes for Language-Based Sources

Ying Wang†, Minghai Qin§, Krishna R. Narayanan†, Anxiao (Andrew) Jiang‡, and Zvonimir Bandic§
†Department of Electrical and Computer Engineering, Texas A&M University
‡Department of Computer Science and Engineering, Texas A&M University
§Next Generation Platform Technologies, Western Digital Research
{yingwang@tamu.edu, minghai.Qin@hgst.com, krn@ece.tamu.edu,
ajiang@cse.tamu.edu, and zvonimir.bandic@wdc.com}

***Abstract*—We propose a joint list decoder and language decoder that exploits the redundancy of language-based sources during polar decoding. By judging the validity of decoded words in the decoded sequence with the help of a dictionary, the polar list decoder constantly detects erroneous paths after the decoding of every few bits. This path-pruning technique based on joint decoding has advantages over stand-alone polar list decoding in that most decoding errors in early stages are corrected. We show that if the language structure can be modeled as erasure correcting outer block codes, the rate of inner polar code can be increased while still guaranteeing a vanishing probability of error. To facilitate practical joint decoding, we first propose a construction of a dynamic dictionary using a trie and show an efficient way to trace the dictionary during decoding. Then we propose a joint decoding scheme for polar codes taking into account both information from the channel and the source. The proposed scheme has the same decoding complexity as the list decoding of polar codes. A list-size adaptive joint decoding is further implemented to largely reduce the decoding complexity. Simulation results show that the joint decoding schemes outperform stand-alone polar codes with CRC-aided successive cancellation list decoding by over 0.6 dB.**

I. Introduction

Shannon's theorem [1] shows that separate optimization of source and channel codes suffices for communicating sources over a large class of channels. However, such a separation-based scheme is often subject to impractical computational complexity and unlimited delay. It is well known that joint source and channel decoding (JSCD) can outperform separation-based schemes in the presence of complexity and delay constraints. It is based on the fact that source coding is often not perfect and that the leftover redundancy can be exploited in the channel decoder. In particular, for language-based sources, the left over redundancy from the source encoder allows us to exploit features such as the meaning of words, grammar and syntax.

Several works have considered JSCD schemes that exploit the left over redundancy from the source encoder. In [2], JSCD using a soft-output Viterbi algorithm is considered. In [3], a trellis based decoder is used as a source decoder in an iterative decoding scheme. Joint decoding of Huffman and Turbo codes is proposed in [4]. In [5], joint decoding of variable length codes (VLCs) and convolutional/Turbo codes is analyzed. Applications of turbo codes to image/video transmission are shown in [6] and [7]. Joint decoding using LDPC codes for VLCs and images are illustrated in [8] and [9], respectively. However, few works have considered JSCD specifically for language-based sources. In [10], LDPC codes are combined with a language decoder and a message passing algorithm is designed to exploit the redundancy in the source.

This work was supported in part by the National Science Foundation under grants IIP-1439722 and CCF-1217944.

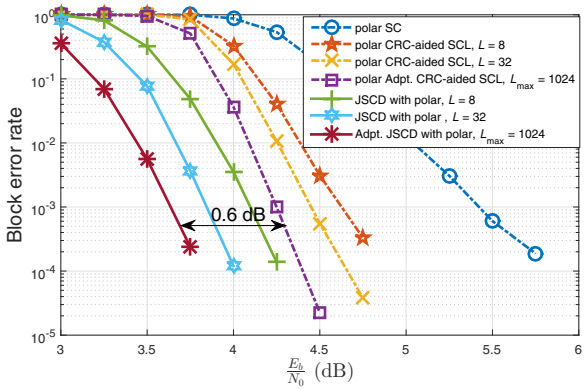


Fig. 1: Block error rate of different decoding schemes over AWGN channels: a) SC decoding; b) CRC-aided SCL decoding ($L = 8, 32$); c) List-size adaptive CRC-aided SCL decoding ($L_{\max} = 1024$); d) JSCD ($L = 8, 32$); e) List-size adaptive JSCD ($L_{\max} = 1024$). All codes have length $n = 8192$ and $k = 7561$.

Polar codes are gaining more attention due to their capacity achieving property [11] and advantages such as low encoding/decoding complexity and good error floor performance [12]. In particular, it is shown in [13] that with successive cancellation list (SCL) decoding, the concatenation of polar codes with a few bits cyclic redundancy check (CRC) can outperform some LDPC codes. We denote the concatenated codes as polar-CRC codes.

In this paper, we propose a joint source-channel decoding scheme where the source is encoded by Huffman codes and transmitted through noisy channels by polar codes. The

978-1-5090-1328-9/16/$31.00 ©2016 IEEE

proposed scheme decodes polar codes jointly with a language decoder based on a word dictionary. We assume the dictionary is only available and used on the decoder side, which is a reasonable assumption when the decoder has larger storage space and stronger calculation power, e.g., uplink channels where the source is compressed at a mobile device and uploaded to a data center. The language decoder has a similar function to that of a CRC. Instead of picking a valid path from the list, the language decoder uses the word dictionary to select most probable paths, where the word dictionary can be viewed as local constraints on the decoded subsequences. A critical advantage of the language decoder over global CRC constraints is that it can detect the validity of partially decoded paths before decoding the whole codeword. In this way, incorrect paths can be pruned at early stages, resulting in a larger probability that the correct path survives in the list. The proposed decoder exploits the fact that the language decoder and the polar decoder both work over trees and that they can be combined in a computationally efficient way. We provide an analysis of the increase in the rate of the polar code that can be obtained by modeling the language-based source as t-erasure correcting outer codes. The proposed decoder provides substantial improvement in performance over the stand-alone CRC-aided SCL decoding. Fig. 1 shows a block error rate comparison of different polar decoders for transmitting English text as sources. It can be observed that over 0.6 dB gain can be achieved by JSCD over stand-alone CRC-aided SCL decoding with comparable code parameters.

II. Background

In this section, we give a brief review of polar codes and SCL decoding. Throughout the paper, we will denote a vector $(x_i, x_{i+1}, \ldots, x_j)$ by x_i^j, denote the set of integers $\{1, 2, \ldots, n\}$ by $[n]$, denote the complement of a set F by F^c, and denote a probability measure by $P(\cdot)$.

A. Polar codes

Polar codes are recursively encoded with the generator matrix $G_n = R_n G_2^{\otimes m}$, where R_n is a $n \times n$ bit-reversal permutation matrix, $G_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$, and $\otimes$ is the Kronecker product. The length of the code is $n = 2^m$. Arıkan's channel polarization principle consists of two phases, namely channel combining and channel splitting. Let u_1^n be the bits to be encoded, x_1^n be the coded bits and y_1^n be the received sequence. Let $W(y|x)$ be the transition probability of a binary-input discrete memoryless channel (B-DMC). For channel combining, N copies of the channel are combined to create the channel

$$W_n(y_1^n|u_1^n) \triangleq W^n(y_1^n|u_1^n G_n) = \prod_{i=1}^{n} W(y_i|x_i),$$

where the last equality is due to the memoryless property of the channel. The channel splitting phase splits W_n back into a set of n bit channels

$$W_n^{(i)}(y_1^n, u_1^{i-1}|u_i) \triangleq \frac{1}{2^{n-1}} \sum_{u_{i+1}^n} W_n(y_1^n|u_1^n),\ i = 1, \ldots, n.$$

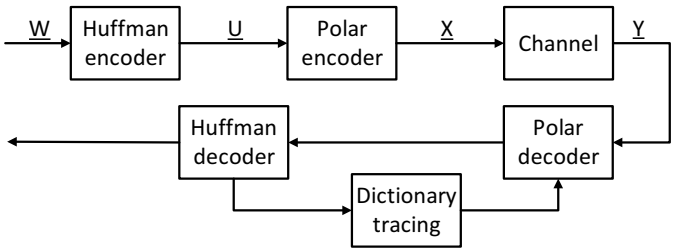


Fig. 2: A system model for joint source-channel decoding

Let $I(W)$ be the channel capacity of W. The bit channels $W_n^{(i)}$ will polarize in the sense that a fraction of bit channels will have $I(W_n^{(i)})$ converging to 1 as $n \to \infty$ and the rest will have $I(W_n^{(i)})$ converging to 0. Arıkan shows in [11] that for the binary-input discrete memoryless channels, the fraction of $I(W_n^{(i)})$ converging to 1 equals $I(W)$. Let us define the frozen set as $\mathcal{F} \triangleq \{i \in [0, 2^n - 1] : I(W_n^{(i)}) \in [0, 1 - \epsilon)\}$ where $\epsilon \in (0, 1)$. If we fix bits in $\mathcal{F}$ and transmit information bits only through the rest of bit channels, an arbitrarily small transmission error probability can be guaranteed.

B. Decoding of polar codes

Successive cancellation (SC) decoding makes decision sequentially on $u_i, i \in [n]$ based on $W_n^{(i)}(y_1^n, u_1^{i-1}|u_i)$ and the hard decision of SC decoder of polar codes can lead to severe error propagation. Instead, SCL decoder keeps a list of most probable paths. In each stage, the decoder extends a path by hypothesizing both 0 and 1 for the unfrozen bit and the number of paths doubles. Assume the list size is L. When the number of paths exceeds L, the decoder picks L most probable paths and prunes the rest. After decoding the last bit, the most probable path is picked. The complexity of SCL decoding is $O(Ln \log n)$, where n is the block length. An extra improvement can be brought by CRC, which increases the minimum distance of polar codes and helps to select the most probable path in the list. The adaptive SCL decoder with a large list size can be used to fully exploit the benefit of CRC while largely reducing the decoder complexity [14].

III. System model and joint source-channel decoding

Fig. 2 illustrates the framework of the proposed coding scheme. We consider text in English, and the extension to other languages is straightforward. In our framework, the text is first compressed by Huffman codes and then encoded by polar codes. On the decoder side, the received sequence is jointly decoded by the polar code and a language decoder. The language decoder consists of Huffman decoding and dictionary tracing. It checks the validity of the decoded sequence by recognizing words in the dictionary. A detailed description of the proposed JSCD scheme is given below.

The maximum *a posteriori* decoder aims to find $\max_{u_1^n} P(u_1^n|y_1^n)$. To avoid exponential complexity in n, we use SCL decoding to maximize $P(u_1^i|y_1^n), i = [n]$ progressively by breadth-first search of a path in the decoding tree,

where for each length-i path, a constant number, often denoted by L, of most probable paths are kept to search for length-$(i+1)$ paths. Since

$$P(u_1^i|y_1^n) = \frac{P(u_1^i, y_1^n)}{P(y_1^n)} \propto P(y_1^n|u_1^i)P(u_1^i),$$

by source-channel separation theorem, a stand-alone polar decoder calculates the first term $P(y_1^n|u_1^i) \propto P(y_1^n, u_1^{i-1}|u_i)$ by a recursive structure, assuming u_1^i are independently and identically distributed (i.i.d.) Bernoulli(0.5) random variables, and thus the second term can be obliterated since $P(u_1^i) = 2^{-i}, \forall u_1^i \in \{0,1\}^i$. However, in the language-based JSCD framework, u_1^i are no longer i.i.d., one obvious consequence of which is that u_1^i is feasible only if the decoded text, translated from u_1^i by Huffman decoder, consists words in the dictionary. Therefore, $P(u_1^i)$ contributes critically to the path metric $P(u_1^i|y_1^n)$, and in particular, if $P(u_1^i) = 0$, this path should be pruned despite the metric $P(y_1^n|u_1^i)$ obtained from the channel. This pruning technique enables early detection of decoding errors and is critical in keeping the correct path in the list. Algorithm 1 shows a high-level description of JSCD.

Algorithm 1 A high-level description of JSCD

Input: y_1^n, L
Output: u_1^n

Initialize: $i \leftarrow 1$; $l_{\text{act}} \leftarrow 1$;
while $i \leq n$ **do**
  if $i \in F$ **then**
    $u_i \leftarrow 0$ for each active path;
  else
    $k \leftarrow 1$;
    for each active path $l_j, j \in [l_{\text{act}}]$ **do**
      for $u_i = 0, 1$ **do**
        Compute $P(y_1^n, u_1^{i-1}|u_i)$;
        Update $P(u_1^i)$;
        $M_k \leftarrow P(y_1^n, u_1^{i-1}|u_i)P(u_1^i)$;
        $k \leftarrow k + 1$;
    $\rho \leftarrow \min(2l_{\text{act}}, L)$;
    Keep most probable ρ paths according to $M_1^{2l_{\text{act}}}$;
    $l_{\text{act}} \leftarrow \rho$;
  $i \leftarrow i + 1$;
Select the most probable path and output u_1^n.

IV. Rate improvement by JSCD

As seen in the previous section, compressed language-based sources still have redundancy. In this section, we show that if the redundancy can be modeled as outer codes of n_0 bits that correct t erasures, which refer to as $[n_0, t]$ outer codes, the rate of the inner polar codes can be increased while maintaining a vanishing error probability. For SC decoders, if the first $n_0 - t$ bits of the n_0 bits are decoded correctly, the last t bits can be filled in by the outer erasure code, regardless of the decisions on the last t bits made by the SC decoder. Equivalently, the rate of the polar code can be increased by sending those last t bits in frozen bit positions. We give a simple example.

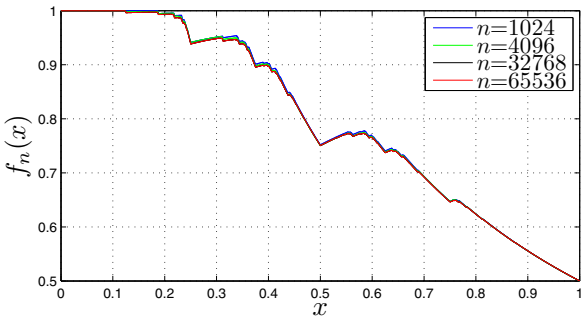


Fig. 3: Distributions of frozen bits with different code lengths

Example 1. Consider an $(8, 4)$ polar code and let the outer code be a $[2, 1]$ code. Let u_1, u_2, u_3 and u_5 be the frozen bits. We can unfreeze u_5 by grouping u_5 with u_4 into a codeword of the $[2, 1]$ outer code that corrects 1 erasure. If u_4 is decoded correctly by the polar code, the outer code can correctly decode u_5. The rate of inner polar code is therefore increased by $\frac{1}{8}$ without increasing the block error rate.

Define $f_n(x) : [0, 1] \mapsto [0, 1]$ as the proportion of frozen bits in the first nx bits of a polar code of rate R and length n. Fig. 3 shows the function $f_n(x)$ for increasing n and fixed $R = 0.5$ where polar codes are optimized at BEC (0.5). It can be seen that $f_n(x)$ appears to converge to some function $f(x) : [0, 1] \mapsto [0, 1]$ as $n \to \infty$. Based on this numerical plot, we will assume that is indeed the case for this part of the paper. A proof for this is beyond the scope of this paper.

A. $[2, 1]$ *outer codes*

We start by considering $[2, 1]$ outer codes. Among the first $n(1-R)$ bits, there are $n(1-R)\,(1 - f_n(1-R))$ unfrozen bits. This is equal to the number of frozen bits among the last nR bits. Therefore from the polar code perspective, all frozen bits in the last nR bits can be unfrozen and can be used to transmit repeated bits from the first $n(1-R)$ bits.

Theorem 2. For $[2, 1]$ outer codes, the rate of the polar code can be increased by

$$\begin{aligned}\Delta R &= \lim_{n\to\infty} \frac{n(1-R)\,(1 - f_n(1-R))}{n} \\ &= (1-R)\,(1 - f(1-R))\end{aligned}$$

B. $[n_0, t]$ *outer codes*

Theorem 3. Let $h(x)$ be an upper bound on $f(x)$. If outer codes can be modeled as length-n_0 codes that correct t erasures, the rate of polar code can be increased by $\Delta R = 1 - R - x_0 h(x_0)$, where $x_0 \in (0, 1)$ satisfies the condition

$$\frac{nx_0(1 - h(x_0))}{n_0 - t} \geq \frac{n - k - nx_0 h(x_0)}{t}. \tag{1}$$

Proof: Let $h_n(x)$ upper bounds $f_n(x)$ for each n and assume $h(x) = \lim_{n\to\infty} h_n(x)$. Let $x_0 \in (0, 1)$. Among the first nx_0 bits, there are $nx_0(1 - f_n(x_0)) \geq nx_0(1 - h_n(x_0))$ unfrozen bits. Among the last $n(1 - x_0)$ bits, there are

$n - k - nx_0 f_n(x_0) \geq n - k - nx_0 h_n(x_0)$ frozen bits. If inequality (1) holds, we can unfreeze $n-k-nx_0h_n(x_0)$ frozen bits among the last $n(1-x_0)$ bits, and take $nx_0(1-h_n(x_0))$ unfrozen bits among the first nx_0 bits and group them into length-n_0 outer codes. Then the increased rate is $\Delta R = \lim_{n\to\infty} \frac{n-k-nx_0h_n(x_0)}{n} = 1 - R - x_0h(x_0)$. ■

A simple yet useful choice of $h(x)$ is a piece-wise linear function as follows. Let $\alpha > 0$ and $\beta > \max(1, \alpha)$ be two constant real numbers. Then $h(x)$ can be chosen as

$$h(x) = \begin{cases} 1, \text{ if } 0 < x \leq \dfrac{\beta - 1}{\alpha} \\ -\alpha x + \beta, \text{ if } \dfrac{\beta - 1}{\alpha} < x \leq 1. \end{cases} \quad (2)$$

Example 4. We choose $h(x)$ in Eq. (2) with $\alpha = 0.6$ and $\beta = 1.1$. If outer codes are $[7, 2]$ codes that correct 2 erasures, $x_0 = 0.59$ satisfies inequality (1). Then we can derive $\Delta R = 0.06$. If outer codes are $[8, 3]$ codes that correct 3 erasures, we can derive that $\Delta R = 0.076$.

Note that in Theorem 2 and Theorem 3, n bits are partitioned into two segments, where frozen bits in the second segment are then unfrozen to transmit bits that are correctable by unfrozen bits in the first segment. This idea can be generalized by partitioning the n bits into $m + 1 \geq 2$ segments. Let $h(x)$ and $g(x)$ be upper and lower bounds on $f(x)$, respectively. Let $x_1, x_2, \ldots, x_m$ satisfy $\frac{\beta-1}{\alpha} < x_1 < x_2 < \cdots < x_m < 1$. They partition the codewords into $m + 1$ segments.

Theorem 5. If outer codes can be modeled as length-n_0 codes correcting t erasures, we partition the codewords into $m + 1$ segments. The rate of polar code can be increased by

$$\Delta R = \lim_{n\to\infty} \frac{\sum_{i=1}^{m} F_{i+1}}{n} = \sum_{i=1}^{m} \left(x_{i+1} g(x_{i+1}) - x_i h(x_i) \right),$$

where $x_1, \ldots, x_m$ satisfy $\frac{U_i}{n-t} = \frac{F_{i+1}}{t}, \forall i \in [m]$, F_i and U_i are defined as

$$F_i \triangleq nx_i g(x_i) - nx_{i-1} h(x_{i-1}),$$
$$U_i \triangleq nx_i - nx_{i-1} - nx_i h(x_i) + nx_{i-1} g(x_{i-1}),$$

i.e., lower bounds on number of frozen and unfrozen bits in the ith segment.

V. Implementation of JSCD

In this section we show practical implementation of JSCD. Let $\mathcal{A}$ be the alphabet of symbols in text (e.g., $\{a, b, \ldots, z\}$ for lowercase English letters, $\{0, \ldots, 127\}$ for symbols in ASCII table). Let $\mathcal{D}$ be the set of words in the dictionary. We assume a first order approximation of English words where all words are independent. The performance improvement of considering higher order Markov models of words [15] in languages is diminishing since redundancy within a Huffman-encoded word are much larger than redundancy across a sequence of words.

Proposition 6. The prior probability of source $P(u_1^i)$ can be efficiently computed from dictionary as follows:

$$P(u_1^i) = \prod_{m=1}^{j-1} P(w_m) P(l_1^k r) = \prod_{m=1}^{j-1} P(w_m) \sum_{w} P(w), \quad (3)$$

where w_1^{j-1} are $j-1$ uniquely decoded words in $\mathcal{D}$, l_1^k are k uniquely Huffman-decoded symbols in $\mathcal{A}$ and r is the remaining bit sequence. In the summation, $w \in \mathcal{D}$ satisfies that in binary Huffman-coded representation, the first k symbols equals l_1^k and r is a prefix of the remaining bit sequences.

Remark 7. The calculation of $P(u_1^i)$ should also take into account the probability of spaces (or punctuations) between words. We append a space mark to all words and omit the details of implementation due to space limitations.

Now we focus on the efficient calculation of Eq. (3). Two trees are used to facilitate the calculation, one is a tree for Huffman coding and the other is a prefix tree (i.e., a trie) for tracing a partially decoded word in the dictionary.

A. Trie representation of the dictionary

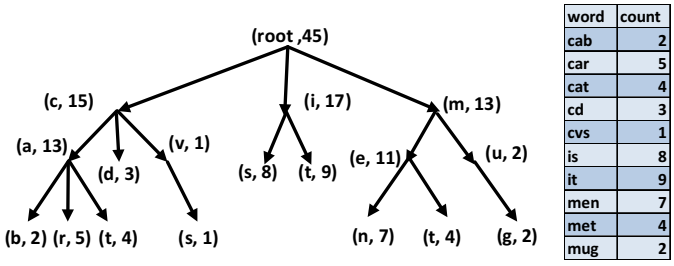
A trie is an ordered tree data structure that is used to store a dynamic set or associative array where the keys are usually strings [16]. In our implementation, each node in the trie is instantiated as an object of a class named `DictNode`. As shown in Table I, it has 4 data members, a symbol `c` (e.g., English letter), a variable `count` representing the frequency of the presence of this prefix, an indicator `is_a_word` indicating if the path from root to this node is a whole word, and a vector of pointers `child[]` pointing to their children. Fig. 4 is an illustrative example of the dictionary represented by a trie. In an established trie, if the pointer that points to the end of a word (or a partial word) w is known, then the calculation of $P(w)$ can be accomplished in $O(1)$ by dividing the count of the end node of the path associated with w by the count of the root node.

TABLE I: `DictNode` members

member	type
c	char
count	int
is_a_word	bool
child[]	DictNode*

TABLE II: `HuffNode` members

member	type
p	double
leftChild	huffNode*
rightChild	huffNode*
symSet	char*



word	count
cab	2
car	5
cat	4
cd	3
cvs	1
is	8
it	9
men	7
met	4
mug	2

Fig. 4: An illustrative example of a trie to represent the dictionary

In order to establish the trie from extracted text (e.g., from books, websites, etc.), an algorithm with an inductive process can be used. The algorithm is depicted in Algorithm 2 and detailed explanations can be found in [17].

Since searching for a symbol as a child of a node in $\mathcal{T}$ can be accomplished in $O(1)$ using a Hash table (e.g.,

Algorithm 2 Establish a trie for the dictionary from text

Input: a sequence of words $(w_1^N w_2 \dots w_N)$, each word is represented as a string
Output: a trie $\mathcal{T}$

```
1: Initialize: Create a root node of T as an object of DictNode;
2: for k = 1 to N do
3:     Let p_dict point to the root of T;
4:     for i = 1 to the length of w_k do
5:         if *p_dict has no child or w_k[i] is not in the children set of *p_dict then
6:             Create a new node as an object of DictNode with c ← w_k[i], count ← 1 and is_a_word ← False;
7:             Insert the new node as a child of *p_dict;
8:             Move p_dict to the new node;
9:             if i == the length of w_k then
10:                p_dict->is_a_word ← True;
11:        else
12:            Find j, s.t. w_k[i] ==p_dict->child[j]->c;
13:            p_dict->count++;
14:            p_dict ← p_dict->child[j];
```

unordered_map STL container in C++), the time complexity of establishing the trie would be $O(N_{\text{length}} N_{\text{word}})$, where N_{length} is the average length of a word and N_{word} is the number of words extracted from some resource.

B. Tree representation of Huffman codes

The Huffman codes for source coding are for 1-grams, namely characters, or more specifically, letters and space mark. In principle, we can also build a Huffman code for n-grams. The Huffman codes are represented as a binary tree. Each node in the tree is instantiated as an object of a class HuffNode whose members are shown in Table II. In a typical Huffman tree realization, a node m consists of three members: the probability p of the associated symbol and two pointers to their left and right children (leftChild and rightChild). In addition, we implement a fourth data member symSet, that is, a set of symbols that are descendants of m. This extra data member helps in simplifying the calculation of Eq. (3) in the following manner. Note that in Eq. (3), $P(l_1^k r)$, the probability of a partial word is required. Assume l_1^k is a path that ends in a node n_k in the trie-represented dictionary $\mathcal{T}$ and r is a path that ends in a node n_r in the Huffman tree $\mathcal{H}$. Then $P(l_1^k r)$ can be calculated by summing up the counts (or probability) of the subset of children of $n_k \in \mathcal{T}$, such that the symbols associated with this subset are all descendants of $n_r \in \mathcal{H}$. By associating all descendants of n_r as a data member to the node itself, the complexity of calculating $P(l_1^k r)$ is linear in the number of descendants of n_r, which is typically a small number and decreases exponentially in the depth of n_r.

C. Calculation of $P(u_1^i)$ with $\mathcal{T}$ and $\mathcal{H}$

Next, we will present an algorithm to calculate $P(u_1^i)$ progressively according to Eq. (3). In each of $\mathcal{T}$ and $\mathcal{H}$, two pointers, denoted by p_dict and p_huff, are used respectively to locate the current decoding stages $i \in [n]$. They are initiated to point to the root of $\mathcal{T}$ and $\mathcal{H}$, respectively. A simple description of the algorithm is as follows. Let u_1^{i-1} be represented as $(w_1^{j-1} l_1^k r)$ and suppose each term in Eq. (3) is known up to index $i-1$. Suppose p_dict and p_huff point to two nodes in $\mathcal{T}$ and $\mathcal{H}$. To update $P(u_1^i)$, first, p_huff moves to its left or right child according to u_i. Let $\mathcal{S}$ denote all descendant symbols of *p_huff. Replace $P(l_1^k r)$ by the summation of probabilities associated with a set of children, denoted by $\mathcal{C}$, of *p_dict such that $\forall a \in \mathcal{C}$, the symbols associated with a belongs to $\mathcal{S}$; If *p_huff is a leaf, then p_dict moves to its child according to the symbol *p_huff associates and p_huff is reset to point to the root of $\mathcal{H}$. If the symbol that *p_huff associates with does not exist in the children of *p_dict, that means $P(u_1^i)$ should be set to 0 and this path has a decoding error and thus be pruned. If furthermore *p_dict is an end node of a word in $\mathcal{T}$, replace $P(l_1^k r)$ by $P(w_j)$ and p_dict is reset to point to the root of $\mathcal{T}$. Let the multiplication of probabilities in Eq. (3) be denoted by P_{wd}, i.e., $P_{wd} = \prod_{m=1}^{j-1} P(w_m)$, where P_{wd} can be updated recursively. A detailed description of this algorithm is presented in Algorithm 3. Functions in Line 1 and Line 2 are literally described above and details on complexity analysis are provided in [17].

Algorithm 3 Update $P(u_1^i)$

Input: u_i, $\mathcal{T}$, $\mathcal{H}$, p_dict, p_huff, P_{wd}
Output: p_dict, p_huff, $P(u_1^i)$, P_{wd}

```
1: S ←TraceHuffmanTree(H,p_huff,u_i);
2: C ←TraceDict(T,p_dict,S);
3: P(l_1^k r) ← Σ_{w∈C} P(w);
4: P(u_1^i) ← P_wd · P(l_1^k r);
5: if p_huff points to a leaf in H then
6:     Move p_dict to its child according to p_huff ;
7:     Move p_huff to the root of H;
8:     if p_dict points to a leaf in T then
9:         P(w_j) ← P(l_1^k r);
10:        P_wd ← P_wd · P(w_j);
```

Theorem 8. The overall decoding complexity of JSCD in Algorithm 1 and 3 is $O(Ln(\log n))$.

D. List-size adaptive JSCD

To improve the efficiency of JSCD, we implement the list-size adaptive SCL decoders as in [14]. A few CRC bits are added for error detection. The adaptive SCL decoders start with $L = 1$ and computes an estimate u_1^n. If u_1^n satisfies the CRCs, then u_1^n are output as the decoded bits, otherwise, the list size doubles and the SCL JSCD is repeated. This process continues until u_1^n satisfies the CRCs for some L_{success} or the list size reaches a threshold $L_{\max}$.

VI. Numerical results

In this section, we present some numerical results that show the superiority of JSCD over the stand-alone SCL decoder.

A. Dictionary

The dictionary is built from about 10 million extracted words in Wikipedia pages. According to a word frequecy analysis in [18], the top 3000 most frequent words take 81% of the probability. In the dictionary tree implemented in this paper, there are $N_s = 180133$ nodes of type `DictNode`.

B. Polar codes and channel parameters

In our simulation, the length of polar codes is fixed to $n = 8192$ and the code rate is 0.923. Two typical B-DMCs are assumed, namely, AWGN channels and binary symmetric channels (BSCs). The polar code used for AWGN channels is constructed by density evolution in [19] at $\frac{E_b}{N_0} = 4$ dB. The polar code used for BSCs is similarly constructed for a BSC with cross-over probability 0.002, which is the same as the channel parameter for LDPC designs in [10].

C. Results

Fig. 1 shows a comparison of different decoders for AWGN channels. It can be seen that at block error rate below 10^{-3}, more than 0.6 dB gain over stand-alone CRC-aided SCL decoders with $L = 1024$ can be realized by the list-size adaptive SCL JSCD decoders. It is observed in our simulation that $L = 1024$ would be large enough such that further increase of the list size will not contribute much to the performance. The decoding complexity of the list-size adaptive SCL JSCD is much lower than that of SCL JSCD with fixed list size. Table III shows that the average list size L_{success} decreases dramatically with the increase of SNRs. We see that at $\frac{E_b}{N_0} = 4$ dB, $L_{\text{success}} = 2.24$ for all $L_{\max} = 128$ and 1024.

Fig. 5 shows a comparison of 4 decoders for BSCs. The results consistently show the superiority of JSCD over CRC-aided SCL decoding. Fig. 6 shows a comparison of joint decoding using LDPC codes [10] and our schemes. The polar code has length $n = 4096$ and rate 0.936, which is smaller in length and the same in rate as in [10]. We should note that dictionaries for two schemes are built separately. The dictionary in [10] is incomplete, i.e., not all words in sources are in the dictionary, while our dictionary is complete. Thus the comparison is not entirely fair.

TABLE III: Average list size of JSCD

E_b/N_0 (dB)	3	3.25	3.5	3.75	4
$L_{\max} = 32$	30.89	25.94	13.68	5.46	2.22
$L_{\max} = 128$	113.09	59.27	21.84	5.66	2.24
$L_{\max} = 1024$	547.66	177.57	34.12	6.08	2.24

VII. Conclusion

We exploit the redundancy in the language-based source to help polar decoding. We propose a joint list decoding scheme of polar codes taking into account the source information using a dictionary. The decoding complexity is the same as list decoding of stand-alone polar codes. Simulation results show that our scheme significantly outperforms list decoding of CRC-aided polar codes.

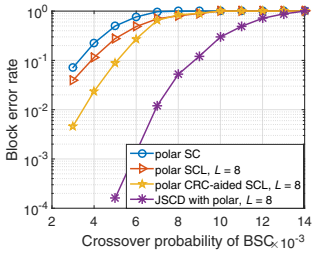


Fig. 5: Block error rate of different decoding schemes over BSCs

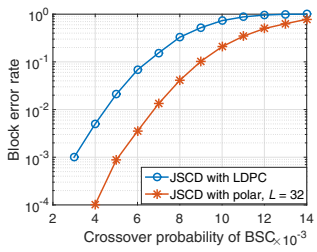


Fig. 6: Comparison of JSCD with LDPC codes [10] over BSCs

References

[1] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 379–423, 1948.

[2] J. Hagenauer, "Source-controlled channel decoding," *IEEE Trans. Commun.*, vol. 43, no. 9, pp. 2449–2457, 1995.

[3] R. Bauer and J. Hagenauer, "On variable length codes for iterative source/channel decoding," in *Proc. 2001 Data Compress. Conf.*, 2001, pp. 273–282.

[4] L. Guivarch, J.-C. Carlach, and P. Siohan, "Joint source-channel soft decoding of Huffman codes with Turbo-codes," in *Proc. DCC 2000*, pp. 83–92.

[5] M. Jeanne, J.-C. Carlach, and P. Siohan, "Joint source-channel decoding of variable-length codes for convolutional codes and Turbo codes," *IEEE Trans. Commun.*, vol. 53, no. 1, pp. 10–15, 2005.

[6] Z. Peng, Y.-F. Huang, and D. J. Costello Jr, "Turbo codes for image transmission-a joint channel and source decoding approach," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 6, pp. 868–879, 2000.

[7] A. N. Kim, S. Sesia, T. Ramstad, and G. Caire, "Combined error protection and compression using Turbo codes for error resilient image transmission," in *Proc. Int. Conf. Image Process. (ICIP)*, vol. 3, 2005, pp. III–912–15.

[8] C. Poulliat, D. Declercq, C. Lamy-Bergot, and I. Fijalkow, "Analysis and optimization of irregular LDPC codes for joint source-channel decoding," *IEEE Commun. Lett.*, vol. 9, no. 12, pp. 1064–1066, 2005.

[9] L. Pu, Z. Wu, A. Bilgin, M. W. Marcellin, and B. Vasic, "LDPC-based iterative joint source-channel decoding for JPEG2000," *IEEE Trans. Image Process.*, vol. 16, no. 2, pp. 577–581, 2007.

[10] A. Jiang, Y. Li, and J. Bruck, "Enhanced error correction via language processing," in *Proc. Non-Volatile Memories Workshop (NMVW)*, 2015.

[11] 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, 2009.

[12] A. Eslami and H. Pishro-Nik, "On finite-length performance of polar codes: stopping sets, error floor, and concatenated design," *IEEE Trans. Commun.*, vol. 61, no. 3, pp. 919–929, 2013.

[13] I. Tal and A. Vardy, "List decoding of polar codes," in *Proc. IEEE ISIT*, 2011, pp. 1–5.

[14] B. Li, H. Shen, and D. Tse, "An adaptive successive cancellation list decoder for polar codes with cyclic redundancy check," *IEEE Commun. Lett.*, vol. 16, no. 12, pp. 2044–2047, 2012.

[15] P. F. Brown, P. V. Desouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai, "Class-based n-gram models of natural language," *Computational linguistics*, vol. 18, no. 4, pp. 467–479, 1992.

[16] E. Fredkin, "Trie memory," *Communications of the ACM*, vol. 3, no. 9, pp. 490–499, 1960.

[17] Y. Wang, M. Qin, K. R. Narayanan, A. Jiang, and Z. Bandic, "Joint source-channel decoding of polar codes for language-based source," *arXiv:1601.06184*.

[18] "Word frequency data," *http://www.wordfrequency.info/free.asp*.

[19] R. Mori and T. Tanaka, "Performance of polar codes with the construction using density evolution," *IEEE Commun. Lett.*, vol. 13, no. 7, pp. 519–521, 2009.