Sparse Graphs for Belief Propagation Decoding of Polar Codes

Sebastian Cammerer, Moustafa Ebada, Ahmed Elkelesh and Stephan ten Brink
Institute of Telecommunications, Pfaffenwaldring 47, University of Stuttgart, 70569 Stuttgart, Germany
{cammerer,ebada,elkelesh,tenbrink}@inue.uni-stuttgart.de

Abstract—We describe a novel approach to interpret a polar code as a low-density parity-check (LDPC)-like code with an underlying sparse decoding graph. This sparse graph is based on the encoding factor graph of polar codes and is suitable for conventional belief propagation (BP) decoding. We discuss several pruning techniques based on the check node decoder (CND) and variable node decoder (VND) update equations, significantly reducing the size (i.e., decoding complexity) of the parity-check matrix. As a result, iterative polar decoding can then be conducted on a sparse graph, akin to the traditional well-established LDPC decoding, e.g., using a fully parallel sum-product algorithm (SPA). We show that the proposed iterative polar decoder has a negligible performance loss for short-to-intermediate codelengths compared to Arıkan’s original BP decoder. Finally, the proposed decoder is shown to benefit from a reduced complexity and reduced memory requirements and, thus, it is more suitable for hardware implementations.

I. INTRODUCTION

Polar codes introduced in [1] are the first type of channel codes that are theoretically proven to achieve the channel capacity for infinite length codes under successive cancellation (SC) decoding. Unfortunately, finite length polar codes suffer from poor error-rate performance under SC decoding. A belief propagation (BP) decoding algorithm based on [2] was proposed in [3] to enhance the finite length performance. Later, a successive cancellation list (SCL) decoder was proposed achieving the maximum likelihood (ML) bound [4].

Polar codes have been recently adopted to the 5G standard as a part of the uplink control channel, thus, practical decoding algorithms of polar codes have become a very attractive topic. On major drawback of SC and SCL decoding is a long decoding latency due to its serial decoding nature, in which the information bits are decoded one at a time, reducing the overall throughput of the decoder. Besides, the SCL decoder does not provide soft-in/soft-out information and thus not suitable for iterative decoding and detection. On the other hand, the BP decoder does not suffer from the above mentioned issues, thus it is a good candidate for high data rate demanding applications (i.e., from an implementation perspective).

The first BP decoder proposed for polar codes is performed over a factor graph corresponding to the generator matrix of the code [3]. A straightforward conversion from the generator matrix-based factor graph to a parity-check matrix-based factor graph [5] leads to a dense (i.e., non-sparse) parity-check matrix with many short cycles as shown in Fig. 2. Thus, the traditional BP algorithm as used in low-density parity-check (LDPC) decoding, with variable node decoder (VND) and check node decoder (CND), fails over the dense graph corresponding to the resulting dense parity-check matrix (see Fig. 1). A first possibility of iterative decoding over a suitable parity-check matrix and corresponding graph pruning steps have been reported in [6]. In this work, we propose a novel approach for finding a sparse parity-check matrix1 of a corresponding LDPC code suitable for iterative decoding after pruning the redundant variable nodes (VN) and check nodes (CN). Thus, a polar code can be decoded with a slightly modified generic BP decoder (e.g., an LDPC BP decoder) with negligible bit error rate (BER) performance loss (refer to Fig. 5). By viewing polar codes as an LDPC-like code, the LDPC design methods or analysis tools (e.g., EXIT charts and density evolution) may become applicable in the context of polar codes. This will facilitate the hard task of systematically analyzing polar codes under BP decoding and may be useful in finding the best polar code construction (i.e., the frozen bits) algorithm for polar codes tailored to iterative decoding. This opens door to new and, probably, more efficient hardware implementation, given the latency reduction provided here and given that polar codes can be, surprisingly, decoded iteratively on a sparse H-based factor graph of reduced size using the conventional well-established LDPC decoders.

This work has been supported by DFG, Germany, under grant BR 3205/5-1.

1Strictly speaking, the proposed matrix H is not a parity-check matrix of the polar code, as Hx requires an extended codeword x. However, for readability and with abuse of some notation we denote H as the parity-check matrix of a corresponding LDPC-like code.

Fig. 1: BER comparison between Arıkan’s original BP decoder, BP decoder performed over the naive H-based dense factor graph, and our proposed LDPC-like BP decoder performed over the pruned factor graph for P(256, 128) and P(2048, 1024) codes. N_{it, max} = 200.
II. CONVENTIONAL BP DECODING OF POLAR CODES

Polar codes are based on the concept of channel polarization in which a set of \( N \) identical channels are combined bit-by-bit. The parity-check matrix corresponding to the LDPC-like code is quite large (i.e., matrix \( \mathbf{P} \)).

The original polar BP decoder factor graph is shown in Fig. 1. The parity-check matrix of the sparse factor graph is shown in Fig. 2. The parity-check matrix \( \mathbf{H} \) of a polar code with generator matrix \( \mathbf{G} \) can be constructed from the columns of \( \mathbf{G} \) with indices in \( \mathcal{A} \), where \( \mathcal{A} \) denotes the set of frozen indices [5, Lemma 1]. The resulting \( \mathbf{H} \)-matrix, shown in Fig. 2, is a highly dense parity-check matrix (see Tab. I) with a maximum check node degree of \( N \). Consequently, this leads to a decoding failure if traditional decoding algorithms (e.g., sum-product algorithm (SPA) decoding) are performed on this dense \( \mathbf{H} \), as shown in Fig. 1. The parity-check matrix of the sparse factor graph was introduced in [5], later termed adjacency matrix in [10], and was used to construct the Linear Programming (LP) polytope which defines the LP decoder proposed in [5].

III. PRUNING TECHNIQUES FOR POLAR FACTOR GRAPHS

As shown in Fig. 2, Arikan’s BP decoding graph can be re-drawn into a bipartite graph consisting of VNs \( \mathcal{V} \) and CNs \( \mathcal{C} \) [5]. By neglecting the stage-wise decoding scheduling, the graph represents an LDPC-like code structure, or a particularly constrained LDPC code, where the last \( N \) bits of the code represent the polar codeword. Besides, there is no channel input over the first \( |\mathcal{V}| - N \) bits. However, the \( \mathbf{H} \)-matrix corresponding to the LDPC-like code is quite large (i.e., matrix
and, as a result, the frozen node can simply be removed from
the graph as it never contributes to the decoding process. 
Based on this observation, the size of the \( \mathbf{H} \)-matrix may vary depending on the code construction (i.e., \( \Delta \)).

2) Degree-1 CN: The underlying parity-check equation of a degree-1 CN can only be fulfilled if the connected VN represents the bit "0". From Eq. 1

\[
y_{c \to v} = \phi(0) \to \infty.
\]

Thus, the degree-1 CNs and the connected VN can be removed from the graph (cf. frozen bits).

3) Degree-1 VNH and degree-2 CN: Obviously, a VNH of degree-1 simply propagates the received \( L_{ch} \) into the connected CN update and never changes its message \( x \). In case the other connected VN is a VNH, the incoming message at this node always equals \( L_{ch} \) from the VNH. Therefore, the CN can be removed and the VNH must be replaced by the corresponding VNH. This is depicted in Fig. 3a.

4) Degree-1 VNH: A VNH of degree-1 (\( v_{nh} \)) always returns \( x_{v_{nh} \to c} = 0 \) to the CN (see Fig. 3b). Using Eq. (1) we find

\[
y_{c \to v} = \text{sign} \left( x_{v_{nh} \to c} \right) \cdot \prod_{v' \in V_{ch}/\{v,v_{nh}\}} \text{sign} \left( x_{v' \to c} \right)
\]

\[
\cdot \phi \left( \left[ x_{v_{nh} \to c} \right] \right) + \sum_{v' \in V_{ch}/\{v,v_{nh}\}} \phi \left( \left[ x_{v' \to c} \right] \right) \to 0,
\]

and thus, the CN never contributes to the decoding and can be removed from the graph together with the degree-1 VNH.

5) Degree-2 VNH: From Eq. (2) it follows that VNH of degree-2 are simple feed-forward nodes (cf. Fig. 3c) and the corresponding VNH update equation simplifies to

\[
x_{v_{nh} \to c} = L_{\text{init}}^{v_{nh}} + y_{c \to v_{nh}}.
\]

Thus, the VNH can be removed by merging the two CNs that are connected to it3.

6) Degree-2 CN: A CN of degree-2, which is only connected to VNH (see Fig. 3d) defines a forwarding node and can thus be removed by merging the two VNHS into one VNH.

Based on the previous observations, we define a pruning algorithm (see Algorithm 1), similar to [6], which simplifies the decoding graph without a significant impact on the decoding performance. The algorithm simplifies the graph iteratively until the size of \( \mathbf{H} \) does not change anymore. Further pruning opportunities may be possible as we here define the baseline schemes. Fig. 4 illustrates the pruning of a \( P(8, 4) \)-code. The \( \mathbf{H} \)-matrix can be pruned from \( 24 \times 32 \) to \( 5 \times 9 \). We provide the source code online to facilitate reproducing the results for further investigation and ideas [11]. Following this algorithm, the dimensions (and density) of the sparse \( \mathbf{H} \)-matrix and

3Remark: this slightly changes the scheduling, as the messages are delayed by one iteration through the VNH. However, we did not observe any significant impact due to this effect.
the degree profiles before pruning are compared to that after pruning for different code lengths, as given in Tab. II. It can be observed that the pruning algorithm leads to a significant matrix size, or graph size, reduction (with a reduction factor of $\approx 83 \text{–} 85\%$ after pruning), while maintaining the sparsity of the $H$-matrix.

One might expect that the pruning process reduces the portion of the degree-1 VNs due to removing the degree-1 VNCHs. However, while pruning the graph, the degree-1 VNCHs are not all removed/altered and thus become more dominant in the degree profile of the reduced $H$-matrix, as can be seen from the variable and check degree distribution from an edge-perspective of the $H$-matrix $(\lambda(Z)$ and $\rho(Z)$) as provided in Tab. II. It is also observed that the portions of degree-2 VNs are reduced due to condensing the degree-2 VNCHs, while the degree-2 VNCHs remain unchanged. The same can be observed, even more significantly, for degree-2 CNs because the condensed degree-2 CNs that are not connected to any VNCH represent a major percentage of the original CNs. Condensing both VNCHs and CNs results in much higher degree (i.e., complex) nodes at a relatively low percentage. One could further refine the pruning by a maximum CN degree, which only allows condensing CNs if the resulting CN degree is below a given limit.

IV. RESULTS AND COMPLEXITY ANALYSIS

The BER performance of polar codes under different decoders, namely SCL, Arıkan’s original BP and our proposed LDPC-like BP over the pruned graph is depicted in Fig. 5. As can be seen, short-to-intermediate length polar codes ($N = 256, 2048$) show a negligible BER performance loss under LDPC-like BP decoding compared to Arıkan’s original BP decoder. One potential reason for a gap is the different scheduling in both decoders. However, we observed that this gap decreases with more BP iterations.

Table III: Complexity comparison between Arıkan’s original and our proposed LDPC-like BP decoder at a fixed BER $= 10^{-4}$ and $N_{it,max} = 200$.

| Code     | BP Decoder | $I_{avg}$ | $c_{ac}$ | $m_{ac}$ | $s_{syn}$ |
|----------|------------|-----------|----------|----------|-----------|
| $P(256, 128)$ | Arıkan    | 3.8       | $1.6 \times 10^4$ | $3.9 \times 10^4$ | 60.8 |
|          | LDPC-like  | 10.9      | $0.4 \times 10^4$ | $2.7 \times 10^4$ | 21.8 |
| $P(2048, 1024)$ | Arıkan | 9.5       | $4.2 \times 10^5$ | $1 \times 10^6$ | 209 |
|          | LDPC-like  | 31.5      | $1.1 \times 10^5$ | $0.1 \times 10^6$ | 63  |

For complexity comparison, the following criteria are used:

- **Average number of iterations performed** $I_{avg}$: denotes the average number of iterations conducted, given a maximum threshold $N_{it,max}$. For the results shown in Tab. III, $N_{it,max} = 200$ is used.
- **Average number of active CNs** $c_{ac}$: denotes the average number of CNs activated over the whole BP iterations.
- **Average number of passed messages** $m_{ac}$: denotes the average number of messages exchanged between different nodes over the whole BP iterations.
- **Average number of synchronization stages** $s_{syn}$: denotes the average number of stages required for decoding (i.e., synchronization steps in a parallel implementation).

As depicted in Tab. III, BP decoding of polar codes over the pruned graphs of the LDPC-like code consumes on average more BP iterations than Arıkan’s original BP decoder of the same polar code. However, as Arıkan’s original BP factor graph is composed of $\log_2 N$ stages, this still gives our proposed decoder an advantage (i.e., the pruned graph is composed of one stage only). In the original BP decoder, $2 \cdot \log_2 N$ stages need to be synchronized per iteration, thus on average less synchronization steps are needed in our proposed decoder. Furthermore, the average number of active CNs

## Algorithm 1: Prune $H$-matrix

**Input:**
- $H_{orig}$ → original $H$-matrix

**Output:**
- $H_{pruned}$ → pruned $H$-matrix

1: $H_{pruned} \leftarrow H_{orig}$
2: $H_{pruned} \leftarrow \text{remove}_\text{Frozen}_\text{VN}(H_{pruned})$
3: while true do
4:   $H_{pruned} \leftarrow \text{prune}_\text{degree1}_\text{CN}(H_{pruned})$
5:   $H_{pruned} \leftarrow \text{condense}_\text{degree1}_\text{VNCH}(H_{pruned})$
6:   $H_{pruned} \leftarrow \text{prune}_\text{degree1}_\text{VNH}(H_{pruned})$
7:   $H_{pruned} \leftarrow \text{condense}_\text{degree2}_\text{VNH}(H_{pruned})$
8:   $H_{pruned} \leftarrow \text{condense}_\text{degree2}_\text{CN}(H_{pruned})$
9:   if size$(H_{pruned})$ does not change then
10:      return $H_{pruned}$
11:  end if
12: end while

Fig. 4: Pruning of the $P(8, 4)$-code graph with $\Lambda = \{4, 6, 7, 8\}$. 
Table II: Dimensions [and density] of the H-matrix and the degree profiles before and after pruning.

| Code | \( \dim \{ H_{\text{before}} \} \) | \( \lambda(\lambda_{\text{before}}) \) (VND) | \( \rho(\rho_{\text{before}}) \) (CND) |
|------|----------------------------------|----------------------------------|----------------------------------|
| \( \dim \{ H_{\text{after}} \} \) | \( \lambda(\lambda_{\text{after}}) \) (VND) | \( \rho(\rho_{\text{after}}) \) (CND) |
| \[1024 \times 2048\] | \[2048 \times 336\] | \[64 \times 1024\] | \[1024 \times 2048\] | \[512 \times 1024\] | \[256 \times 1024\] |
| \[\approx 0.8\%\] | \[\approx 0.1\%\] | \[\approx 0.1\%\] | \[\approx 0.1\%\] | \[\approx 0.1\%\] | \[\approx 0.1\%\] |

V. Conclusion

We show that polar codes can be decoded by a conventional LDPC decoder over a sparse graph of a corresponding LDPC-like code. By carefully pruning the resulting H-matrix, we achieve a significant reduction in the size of the H-matrix, resulting in a faster convergence behavior. The source code for the pruning steps conducted is provided online [11], for testing further possible pruning ideas. Furthermore, we show that there is only a negligible BER performance loss for short-to-intermediate length polar codes, whereas this loss diminishes for more BP iterations. We believe that this approach opens many new – theoretical and practical – research directions for polar BP decoding, as many tools such as, density evolution or EXIT charts may be applied much easier. This might also open door to new and, probably, more efficient hardware implementation, given the latency reduction provided here.

In a future step, this could help to identify improved frozen bit positions (i.e., code design) especially tailored to iterative polar decoding. Another possible research area may be to find ways of considering the CRC during iterative decoding.

REFERENCES

[1] 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, July 2009.
[2] G. D. Forney, “Codes on Graphs: Normal Realizations,” IEEE Trans. Inf. Theory, vol. 47, no. 2, pp. 520–548, Feb. 2001.
[3] E. Arıkan, “Polar codes: A pipelined implementation,” Proc. 4th ISBC, pp. 11–14, 2010.
[4] I. Tal and A. Vardy, “List Decoding of Polar Codes,” IEEE Trans. Inf. Theory, vol. 61, no. 5, pp. 2213–2226, May 2015.
[5] N. Goela, S. B. Korada, and M. Gastpar, “On LP Decoding of Polar Codes,” in IEEE Information Theory Workshop, Aug. 2010, pp. 1–5.
[6] M. Fossorier, “Polar Codes: Graph Representation and Duality,” IEEE Commun. Lett., vol. 19, no. 9, pp. 1484–1487, Sept. 2015.
[7] E. Arıkan, “Systematic Polar Coding,” IEEE Commun. Lett., vol. 15, no. 8, pp. 860–862, Aug. 2011.
[8] W. E. Ryan, “An Introduction to LDPC Codes,” CRC Handbook for Coding and Signal Processing for Recording Systems CRC Press.
[9] B. Yuan and K. K. Parhi, “Early Stopping Criteria for Energy-Efficient Low-Latency Belief-Propagation Polar Code Decoders,” IEEE Trans. Sig. Process., vol. 62, no. 24, pp. 6496–6506, Dec. 2014.
[10] V. Taraknalli and P. H. Siegel, “Adaptive Linear Programming Decoding of Polar Codes,” in IEEE Int. Symp. Inf. Th., June 2014, pp. 2982–2986.
[11] “https://github.com/sebastianca/ldpc-like-decoding-of-polar-codes.”