

Adaptive Linear Programming Decoding of Polar Codes

Veeresh Taranalli and Paul H. Siegel

University of California, San Diego, La Jolla, CA 92093, USA

Email: {vtaranalli, psiegel}@ucsd.edu

Abstract—Polar codes are high density parity check codes and hence the sparse factor graph, instead of the parity check matrix, has been used to practically represent an LP polytope for LP decoding. Although LP decoding on this polytope has the ML-certificate property, it performs poorly over a BAWGN channel. In this paper, we propose modifications to adaptive cut generation based LP decoding techniques and apply the modified-adaptive LP decoder to short blocklength polar codes over a BAWGN channel. The proposed decoder provides significant FER performance gain compared to the previously proposed LP decoder and its performance approaches that of ML decoding at high SNRs. We also present an algorithm to obtain a smaller factor graph from the original sparse factor graph of a polar code. This reduced factor graph preserves the small check node degrees needed to represent the LP polytope in practice. We show that the fundamental polytope of the reduced factor graph can be obtained from the projection of the polytope represented by the original sparse factor graph and the frozen bit information. Thus, the LP decoding time complexity is decreased without changing the FER performance by using the reduced factor graph representation.

I. INTRODUCTION

Polar codes, first introduced in [1], were shown to be capacity-achieving for binary input memoryless output symmetric channels. However, their performance on a binary additive white gaussian noise channel (BAWGNC) with successive cancellation (SC) decoding is unimpressive at practical blocklengths. Thus, improving their performance either by improved decoding algorithms or modified constructions of polar codes has been a recent topic in coding theory. The most notable improvement in error rate performance was observed using the successive cancellation list (SC-List) decoding algorithm proposed in [2]. Alternatively, a cyclic redundancy check (CRC) concatenated polar code with SC-List decoding was also shown to improve the performance significantly [2].

Linear Programming (LP) decoding has been a research topic in coding theory and is attractive mainly because of its maximum likelihood (ML)-certificate property [3]. It was introduced for polar codes in [4] where the sparse factor graph was used to represent the LP polytope instead of the high density parity check matrix. For polar codes over a binary erasure channel (BEC), it was shown that LP decoding achieves capacity and also outperforms SC decoding at finite blocklengths [4]. However, for a BAWGNC, the LP decoder in [4] is suboptimal and performs very poorly. Adaptive LP decoding techniques were proposed in [5], [6] to improve the

decoding time complexity as well as the error rate performance. Based on these techniques, we propose modifications to the adaptive cut generation based LP decoder in [6] that significantly improve its error rate performance for polar codes over a BAWGNC. We then present an algorithm to obtain a smaller factor graph representation of a polar code called the *reduced factor graph*, which decreases the representation complexity of the fundamental polytope and hence improves the decoding time complexity of the modified-adaptive LP decoder.

In Section II, we review the LP decoding of polar codes proposed in [4]. In Section III, we review the adaptive LP decoding techniques [5], [6] and describe the proposed modified-adaptive LP decoder for short blocklength polar codes, along with simulation results. In Section IV, we present the algorithm for reducing a polar code sparse factor graph.

II. LP DECODING OF POLAR CODES

Consider a binary linear code \mathcal{C}_l of length N and rate $r = \frac{k}{N}$, where $k < N$ is the number of information bits in a codeword. Let \mathbf{H} denote a parity check matrix for \mathcal{C}_l . Suppose a codeword $\mathbf{x} \in \mathcal{C}_l$ is transmitted over a binary input memoryless output symmetric channel and \mathbf{y} is the received vector. ML decoding is equivalent to solving the optimization problem [3]:

$$\text{minimize } \gamma^T \mathbf{x} \quad \text{subject to } \mathbf{x} \in \mathcal{C}_l \quad (1)$$

where $x_i \in \{0, 1\}$, $i \in 1, \dots, N$ and γ is the vector of log-likelihood ratios (LLR) defined as

$$\gamma_i = \log \left(\frac{\Pr(y_i | x_i = 0)}{\Pr(y_i | x_i = 1)} \right). \quad (2)$$

In [3], the ML decoding problem (1) was relaxed to a linear programming (LP) problem, where the relaxed polytope, also known as the fundamental polytope \mathcal{Q} has both integral and non-integral vertices. The polytope \mathcal{Q} is defined by linear inequalities, also referred to as constraints, generated from each row j of the parity-check matrix \mathbf{H} , given by

$$\begin{aligned} \sum_{i \in V} x_i - \sum_{i \in \mathcal{N}(j) \setminus V} x_i &\leq |V| - 1 \\ \forall V \subseteq \mathcal{N}(j) \text{ s.t. } |V| &\text{ is odd} \end{aligned} \quad (3)$$

where $\mathcal{N}(j)$ is the support of the row j of \mathbf{H} . This polytope \mathcal{Q} has the ML-certificate property which guarantees that an

integral solution of the LP problem would be a valid ML codeword. The number of constraints needed to define the polytope \mathcal{Q} is exponential in the maximum parity-check degree of \mathbf{H} , i.e., $O(2^{\max_j |\mathcal{N}(j)|})$.

Let a polar code \mathcal{C} be constructed using the channel polarization transform of length $N = 2^m$ proposed in [1], which is denoted by a matrix \mathbf{G}_N , where $\mathbf{G}_N = \mathbf{B}_N \mathbf{G}_2^{\otimes m}$, $\mathbf{G}_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$, the operator $\otimes m$ represents the m -times Kronecker product of \mathbf{G}_2 , and \mathbf{B}_N is the bit-reversal permutation matrix defined in [1]. Assuming all the $N - k$ frozen (non-information) bits in \mathcal{C} are set to 0, a parity check matrix \mathbf{H} for \mathcal{C} can be constructed by selecting the columns corresponding to the frozen bit indices in \mathbf{G}_N as the parity checks [4]. Thus, \mathbf{H} consists of high density rows with the maximum possible parity-check degree being N . Hence, the number of constraints needed to define the polytope \mathcal{Q} as per (3) is $O(2^{N-1})$. This is clearly impractical for all but very short length polar codes. It was also shown in [4] that LP decoding on the fundamental polytope \mathcal{Q} will fail for a BEC(ϵ), binary symmetric channel BSC(p) or a BAWGNC(σ), even if the polytope \mathcal{Q} could be represented in practice.

A sparse factor graph representation with $O(N \log N)$ auxiliary variable nodes was proposed in [1] for the polar code \mathcal{C} . An example sparse factor graph for $N = 8$ is shown in Fig. 1. It is easy to see that there are only degree-3 or degree-2 check nodes in the sparse factor graph. Let $\mathbf{H}_{\mathcal{P}}$ denote the adjacency matrix of the sparse factor graph where the rows and columns represent the check and variable nodes, respectively. The LP polytope \mathcal{P} is defined as the intersection of local minimal convex polytopes of each row (parity check) in $\mathbf{H}_{\mathcal{P}}$ and the set of cutting planes corresponding to the frozen column indices (variable nodes) in $\mathbf{H}_{\mathcal{P}}$ [4]. Using the polytope \mathcal{P} , an LP decoder for the polar code \mathcal{C} as proposed in [4] is given by

$$\begin{aligned} & \text{minimize } \gamma^T \bar{\mathbf{x}} \\ & \text{subject to } \mathbf{x} \in \mathcal{P} \subseteq [0, 1]^{N(1+\log N)} \end{aligned} \quad (4)$$

where $\bar{\mathbf{x}}$ is defined as the first N component subset of \mathbf{x} and corresponds to the codeword variable nodes, $\bar{x}_i = x_i \forall i \in \{1, \dots, N\}$. Similarly, the projection of polytope \mathcal{P} is defined [4] as

$$\bar{\mathcal{P}} = \{ \bar{\mathbf{x}} \in [0, 1]^N \mid \exists \hat{\mathbf{x}} \text{ s.t. } (\bar{\mathbf{x}}, \hat{\mathbf{x}}) \in \mathcal{P} \} \quad (5)$$

It was shown that if the projection $\bar{\mathbf{x}}$ (on $\bar{\mathcal{P}}$) of the LP decoder output vector \mathbf{x} in (4) is integral then it is guaranteed to be the ML codeword i.e., LP decoding on the polytope \mathcal{P} as defined in (4) has the ML-certificate property (Lemma 3 in [4]). It was also shown (Theorem 1 in [4]) that the projection $\bar{\mathcal{P}}$ of polytope \mathcal{P} is tighter than the fundamental polytope \mathcal{Q} .

III. ADAPTIVE LP DECODING OF POLAR CODES

A. Adaptive LP Decoding of a Binary Linear Code

An Adaptive LP (ALP) decoder for binary linear codes solves a sequence of LP decoding problems with the addition of intelligently chosen constraints called *cuts* at every iteration [5]. A cut at a point $\mathbf{x} \in [0, 1]^N$ is a violated constraint

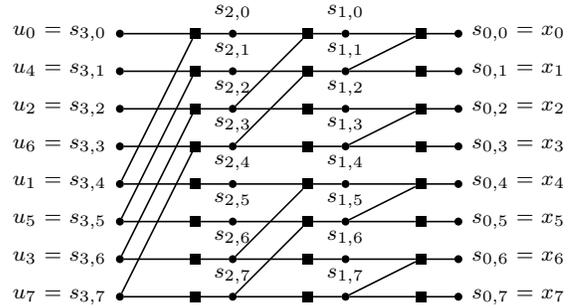


Fig. 1. Sparse factor graph representation of a length-8 polar code.

at \mathbf{x} derived from a check node. An ALP decoder starts by solving the initial LP with the constraints

$$x_i \geq 0 \text{ if } \gamma_i \geq 0; \quad x_i \leq 1 \text{ if } \gamma_i < 0 \quad (6)$$

The solution of this initial LP coincides with the output of a hard decision decoding of the received LLR values. The ALP decoder then searches constraints from all parity checks to find cuts, adds the cuts to the LP and solves the resulting LP. This procedure is repeated until an integer solution is obtained or no further cuts can be found. Violated constraints or cuts can also be generated using redundant parity checks (RPCs). RPCs are obtained by the modulo-2 addition of parity checks in the parity check matrix of the code. The addition of cuts from RPCs during the LP decoding iterations can only tighten the LP polytope and hence can only improve the error rate performance. In [6], efficient algorithms to perform the cut-search on parity checks and to find cut-inducing RPCs were proposed. Based on these algorithms, an adaptive cut generation based LP (ACG-ALP) decoder (Algorithm 2 in [6]) was proposed. Next, we present modifications to the ACG-ALP decoder which make it suitable for decoding polar codes.

B. Modified ACG-ALP Decoder for Polar Codes

A polar code can be defined using the sparse factor graph ($\mathbf{H}_{\mathcal{P}}$) with the frozen bit information or the parity check matrix (\mathbf{H}). The availability of these two representations motivates the idea of modifying the ACG-ALP decoder (Algorithm 2 in [6]) to improve its performance when compared to a LP decoder. The ACG-ALP decoder uses the parity check matrix to generate constraints and cuts. RPCs and cuts from these RPCs are also derived from the parity check matrix by the ACG-ALP decoder. Based on these observations, we investigate four ways of using the sparse factor graph and the parity check matrix representations in the ACG-ALP decoder:

1. Use the unmodified ACG-ALP decoder with the parity check matrix \mathbf{H} .
2. Use $\mathbf{H}_{\mathcal{P}}$ as the parity check matrix in the ACG-ALP decoder. Add the frozen bit constraints to the initial-LP.
3. Initialize the ACG-ALP decoder with the polytope \mathcal{P} and generate subsequent cut-inducing RPCs from $\mathbf{H}_{\mathcal{P}}$.
4. Initialize the ACG-ALP decoder with the polytope \mathcal{P} and generate subsequent cut-inducing RPCs from \mathbf{H} .

Note that the ACG-ALP decoders 1 and 2 do not use constraints from parity checks in defining the initial LP. Hence,

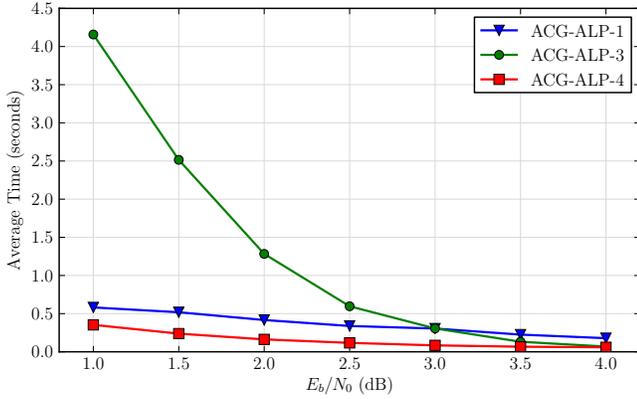


Fig. 2. Average time for decoding one codeword of a (64, 32) polar code over a BAWGN channel with ACG-ALP decoding.

these decoders are expected to have a larger average decoding time complexity compared to the ACG-ALP decoders 3 and 4, as shown in Fig. 2. Due to its large decoding time complexity, the simulation results corresponding to the ACG-ALP decoder 2 could not be obtained. We have empirically observed that the other three ACG-ALP decoders (1, 3, 4) perform equally well in terms of the frame error rate (FER) performance but the ACG-ALP decoder 4 has the smallest average decoding time complexity. Hence, we select this modified decoder for decoding polar codes (ACG-ALP-Polar) as shown in Algorithm 1.

Algorithm 1 ACG-ALP decoding algorithm for Polar codes

Input: γ , \mathbf{H}_P , frozen bit indices, \mathbf{H}

Output: Optimal solution of the current LP problem

- 1: Initialize the LP problem with the constraints obtained from \mathbf{H}_P and frozen bit information.
 - 2: Solve the current LP problem to get the solution \mathbf{x}^* .
 - 3: **if** \mathbf{x}^* is nonintegral **then**
 - 4: Construct cut-inducing RPC matrix $\tilde{\mathbf{H}}$ from \mathbf{H} [6].
 - 5: Apply the cut-search algorithm (Algorithm 1 in [6]) to each row of $\tilde{\mathbf{H}}$.
 - 6: **if** No cut is found **then**
 - 7: Terminate.
 - 8: **else**
 - 9: Add the cuts found to the LP problem, go to line 2.
-

C. Simulation Results

Fig. 3 and Fig. 4 show the FER performance over a BAWGN of rate-0.5 length-64 and length-128 polar codes, respectively. The performance of the proposed ACG-ALP-Polar decoder is compared with the previously proposed LP decoder for polar codes [4], the SC and the SC-List decoders [2]. We choose a list size = 32 for the SC-List decoder (SC-List-32) as it is known to have performance close to the ML lower bound [2]. The polar codes are constructed using the bit channel degrading merge algorithm presented in [7] optimized for a BAWGN channel with signal-to-noise

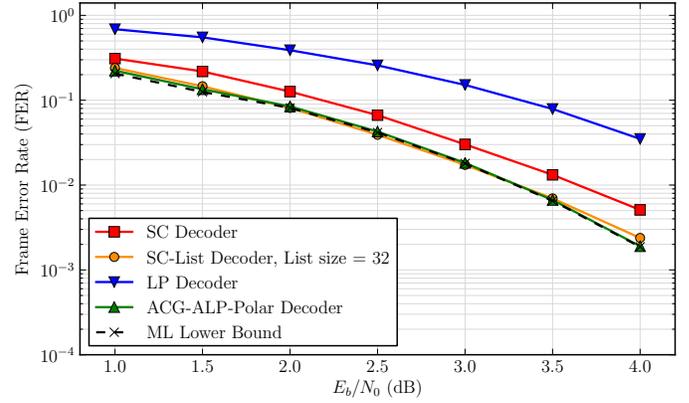


Fig. 3. FER performance of a (64, 32) polar code over a BAWGN channel.

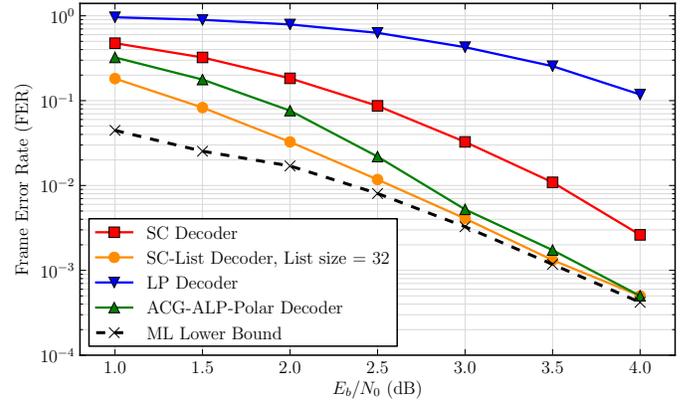


Fig. 4. FER performance of a (128, 64) polar code over a BAWGN channel.

ratio, SNR (E_s/N_0) = 3.0 dB. The proposed decoder uses the LP solver in the CVXOPT package [8]. A total of 200 frame errors were recorded at each E_b/N_0 point. We also show an ML lower bound obtained using the ML-certificate property of the proposed decoder. The ACG-ALP-Polar decoder shows a significant improvement in performance compared to the LP decoder for both polar codes. For the length-64 polar code, the ACG-ALP-Polar decoder performance is very close to the ML lower bound. However, for the length-128 polar code, there is a performance gap in the lower SNR region. The ACG-ALP-Polar decoder performs better than the SC decoder for both polar codes studied. Compared to the SC-List-32 decoder, the ACG-ALP-Polar decoder performs equally well for the (64, 32) polar code while its performance is worse in the low SNR region for the (128, 64) polar code. Although the ACG-ALP-Polar decoder shows promise in FER performance, we have empirically observed that its decoding time complexity is larger than that of the SC-List-32 decoder especially in the low SNR region and is prohibitively high for moderate to long blocklength polar codes.

IV. POLAR CODE SPARSE FACTOR GRAPH REDUCTION

Due to its recursive structure the polar code sparse factor graph has some redundant variable nodes connected to degree-2 check nodes. We use this redundancy and the frozen

bit information to propose an algorithm for reducing the number of constraints needed to represent the sparse factor graph based LP polytope \mathcal{P} . Simulation results show that the new reduced factor graph representation can be used in the ACG-ALP-Polar decoder to achieve an improvement in the time complexity.

A. Polar Code Sparse Factor Graph Reduction Algorithm

The sparse factor graph, the set of frozen bit indices and their values define the polar code. The set of frozen bit indices is obtained using a polar code construction technique [7]. We assume that all frozen bits are set to 0.

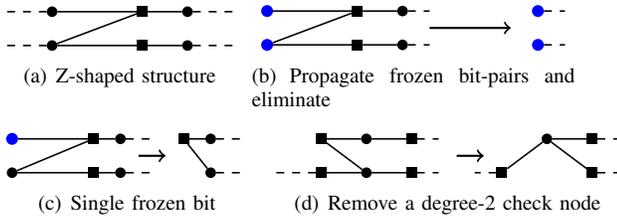


Fig. 5. Reducing a polar code sparse factor graph.

Every pair of degree-2 and degree-3 check nodes in the sparse factor graph is interconnected through a Z-shaped structure shown in Fig. 5(a), referred to as a *Z-structure*. The possible configurations of frozen variable nodes in a Z-structure are:

1. Both the variable nodes on the left are frozen.
2. Only a single variable node is frozen and due to the channel polarization principle, this must be the degree-1 variable node on the left.

In case (1), we propagate the left frozen variable node values (0's in this case) to the right variable nodes of the Z-structure and eliminate the Z-structure as shown in Fig. 5(b). We note that this step is similar to the one proposed in the simplified successive cancellation decoder [9] used to reduce the complexity of the SC decoder. We are left with Z-structures in the graph where only a single bit is frozen i.e., the case (2). We reduce such a Z-structure by replacing it with a degree-2 check node as shown in Fig. 5(c). Now, there are no more frozen variable nodes in the factor graph and hence starting at the code bit variable nodes on the right, we iteratively reduce degree-2 check nodes as shown in Fig. 5(d).

Next, we show that for LP decoding, the polar code sparse factor graph can be reduced further by eliminating degree-1 auxiliary variable nodes and their check node neighbors from the graph.

Lemma 1: The constraints from a parity check node which is connected to a degree-1 auxiliary variable node in the polar code sparse factor graph do not affect the LP decoder solution and hence the degree-1 auxiliary variable node and its check node neighbor can be deleted from the graph.

Proof: From the formulation of LP decoding for polar codes ((4) in Section II), we know that the LP decoder objective function is independent of the auxiliary variable nodes (which do not correspond to codeword bits). Hence a

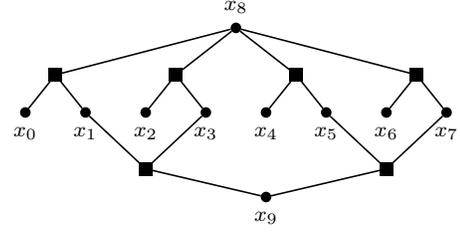


Fig. 6. Reduced factor graph (RFG) of a (8, 4) polar code.

degree-1 auxiliary variable node is free to be assigned any feasible value by the LP solver and can be deleted from the graph. ■

The steps for reducing a polar code sparse factor graph are described in Algorithm 2. Assuming u_0, u_1, u_2, u_4 (Fig. 1) are the frozen bits, the reduced factor graph (RFG) of a (8, 4) polar code sparse factor graph obtained using Algorithm 2 is illustrated in Fig. 6.

Algorithm 2 Reduce Polar Code Sparse Factor Graph

Input: Polar code sparse factor graph $\mathbf{H}_{\mathcal{P}}$, frozen bit indices
Output: Reduced factor graph

- 1: **Step 1:** Propagate frozen variable node pairs as shown in Fig. 5(b) and eliminate the corresponding Z-structures.
 - 2: **Step 2:** Replace Z-structures containing a single frozen variable node with degree-2 check nodes. (Fig. 5(c))
 - 3: **Step 3:** For each degree-2 check node, delete a variable node neighbor connecting all its neighboring check nodes to the other variable node neighbor. (Fig. 5(d))
 - 4: **Step 4:** Iteratively delete degree-1 auxiliary variable nodes and their check node neighbors until no further degree-1 auxiliary variable nodes exist.
-

Let $\mathbf{H}_{\mathcal{R}}$ be a parity-check matrix representation of the polar code reduced factor graph. We show that $\mathbf{H}_{\mathcal{R}}$ has small degree check nodes necessary to represent the LP polytope efficiently.

Lemma 2: A polar code reduced factor graph $\mathbf{H}_{\mathcal{R}}$ consists of only degree-3 check nodes.

Proof: We refer to the steps of Algorithm 2 for this proof. A polar code sparse factor graph has only degree-2 and degree-3 check nodes. Step 1 deletes check nodes in Z-structures with two frozen bits (Fig. 5(b)) and hence does not change the degree of any other check nodes. Step 2 operates on Z-structures with a single frozen bit (Fig. 5(c)) and deletes the degree-2 check node while reducing the degree of the degree-3 check node by 1. Step 3 iteratively deletes degree-2 check nodes and does not affect the degree-3 check nodes. Hence, we are left with only degree-3 check nodes in the factor graph. Step 4 deletes the degree-1 auxiliary variable nodes and their check node neighbors. Therefore, the reduced factor graph $\mathbf{H}_{\mathcal{R}}$ consists of only degree-3 check nodes. ■

Theorem 1: Let \mathcal{R} be the fundamental polytope of the reduced factor graph $\mathbf{H}_{\mathcal{R}}$ of a polar code \mathcal{C} . Then, $\mathcal{R} \subset [0, 1]^d$, where $d = f(N, r)$ is the dimension of the vectors in \mathcal{R} and is a function of the polar code blocklength N and the rate r . Let \mathcal{P} be the polytope obtained from the

original sparse factor graph and the frozen bit information of \mathcal{C} and let $\tilde{\mathcal{P}}$ be the projected polytope obtained from the projection of vectors in \mathcal{P} onto the d variables in $\mathbf{H}_{\mathcal{R}}$. Then,

$$\mathcal{R} = \tilde{\mathcal{P}} \quad (7)$$

Proof: First we show that $\tilde{\mathcal{P}} \subseteq \mathcal{R}$ i.e., every vector in polytope $\tilde{\mathcal{P}}$ is also in polytope \mathcal{R} . Consider a vector $\mathbf{u} \in \tilde{\mathcal{P}}$; its projection of length d , $\tilde{\mathbf{u}} \in \tilde{\mathcal{P}}$, can be constructed by deleting the components of \mathbf{u} corresponding to the variable nodes deleted in Algorithm 2. It is clear that no step in Algorithm 2 requires a change in the value of a variable node which is not deleted and hence $\tilde{\mathbf{u}} \in \mathcal{R}$. Next, we show that $\mathcal{R} \subseteq \tilde{\mathcal{P}}$. Let $\mathbf{v} \in \mathcal{R}$; then \mathbf{v} satisfies all the parity checks in $\mathbf{H}_{\mathcal{R}}$. From the proof of Lemma 2, we know that there are degree-3 parity checks without frozen variable node neighbors in the original sparse factor graph which cannot be reduced. In $\mathbf{H}_{\mathcal{R}}$, even though the variable node indices participating in these checks may be different from those in $\mathbf{H}_{\mathcal{P}}$, the parity checks remain unchanged because there is one representative variable node for a group of deleted variable nodes which were constrained to take on the same values. Hence, the set of parity checks in $\mathbf{H}_{\mathcal{R}}$ is a subset of the parity checks in $\mathbf{H}_{\mathcal{P}}$ and $\mathbf{v} \in \tilde{\mathcal{P}}$. Therefore, $\mathcal{R} = \tilde{\mathcal{P}}$. ■

LP decoding on the polytope \mathcal{P} has the ML-certificate property [4] and from Theorem 1 it follows that the polytope \mathcal{R} also has the ML-certificate property. We replace the matrix $\mathbf{H}_{\mathcal{P}}$ with $\mathbf{H}_{\mathcal{R}}$ in the ACG-ALP-Polar decoder (Algorithm 1). The size of the matrix $\mathbf{H}_{\mathcal{R}}$ is strictly smaller than that of $\mathbf{H}_{\mathcal{P}}$ for any polar code of rate < 1 . Hence the decoding time complexity of the ACG-ALP-Polar decoder can only decrease by using the reduced factor graph.

B. Simulation Results

We present simulation results using the reduced factor graph representation in the ACG-ALP-Polar decoder for the two polar codes discussed in Section III-C. The FER performance is unchanged (Fig. 3 and Fig. 4) and hence is not shown. However, as Fig. 7 shows, the decoding time complexity is decreased when using the reduced factor graph representation. Compared to the SC-List-32 decoder, the ACG-ALP-Polar decoder with the reduced factor graph has a lower average decoding time complexity at higher SNRs. The reduction in the representation complexity of polar codes using the reduced factor graph is shown in Fig 8.

V. CONCLUSION

We proposed modifications to the ACG-ALP decoder [6] which make it suitable for decoding short to moderate length polar codes with FER performance close to the ML performance. This indicates that with the proper polytope representation, LP decoding works well for polar codes over a BAWGNC. We also presented an algorithm to generate an efficient reduced factor graph representation of a polar code. This reduced factor graph decreases the decoding time complexity of the ACG-ALP-Polar decoder without degrading its error rate performance.

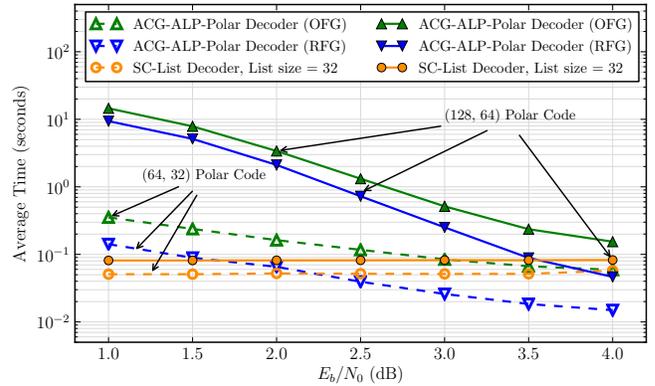


Fig. 7. Average time for decoding one codeword of a (64, 32) and (128, 64) polar code over a BAWGN channel. OFG – with original factor graph; RFG – with reduced factor graph.

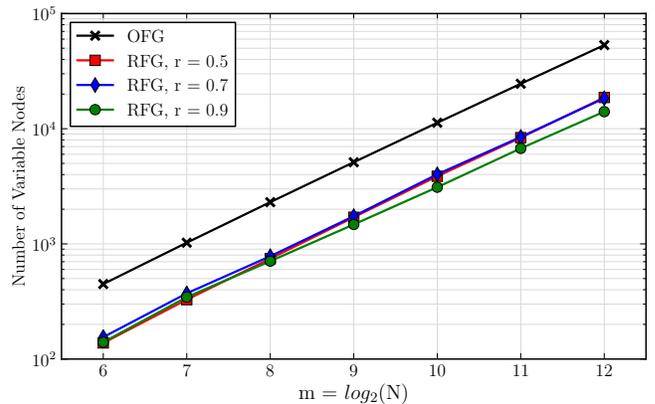


Fig. 8. Representation complexity ($d = f(N, r)$) for polar codes using the original sparse factor graph (OFG) and the reduced factor graph (RFG).

ACKNOWLEDGMENT

The authors would like to thank Aman Bhatia for helpful discussions. This work was supported in part by the Center for Magnetic Recording Research at the University of California, San Diego and the Western Digital Corporation.

REFERENCES

- [1] E. Arıkan, “Channel Polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels,” *IEEE Trans. Inform. Theory*, vol. 55, no. 7, pp. 3051–3073, July 2009.
- [2] I. Tal and A. Vardy, “List decoding of polar codes,” in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, August 2011.
- [3] J. Feldman, M. J. Wainwright, and D. R. Karger, “Using linear programming to decode binary linear codes,” *IEEE Trans. Inform. Theory*, vol. 51, no. 3, pp. 954–972, March 2005.
- [4] N. Goela, S. B. Korada, and M. Gastpar, “On LP decoding of polar codes,” in *Proc. IEEE Inf. Theory Workshop (ITW)*, August 2010.
- [5] M. H. Taghavi and P. H. Siegel, “Adaptive methods for linear programming decoding,” *IEEE Trans. Inform. Theory*, vol. 54, no. 12, pp. 5396–5410, December 2008.
- [6] X. Zhang and P. H. Siegel, “Adaptive cut generation algorithm for improved linear programming decoding of binary linear codes,” *IEEE Trans. Inform. Theory*, vol. 58, no. 10, pp. 6581–6594, October 2012.
- [7] I. Tal and A. Vardy, “How to construct polar codes,” *IEEE Trans. Inform. Theory*, vol. 59, no. 10, pp. 6562–6582, October 2013.
- [8] M. S. Andersen, J. Dahl, and L. Vandenberghe, “CVXOPT: A python package for convex optimization, version 1.1.6. Available at cvxopt.org,” 2013.
- [9] A. Alamdar-Yazdi and F. R. Kschischang, “A simplified successive-cancellation decoder for polar codes,” *IEEE Communications Letters*, vol. 15, no. 12, pp. 1378–1380, December 2011.