

Analysis of Connections between Pseudocodewords

Nathan Axvig, Deanna Dreher, Katherine Morrison, Eric Psota,
Lance C. Pérez, *Senior Member, IEEE* and Judy L. Walker, *Member, IEEE*

Abstract—The role of pseudocodewords in causing noncodeword outputs in linear programming (LP) decoding, graph cover decoding, and iterative message-passing decoding is investigated. The three main types of pseudocodewords in the literature — linear programming pseudocodewords, graph cover pseudocodewords, and computation tree pseudocodewords — are reviewed and connections between them are explored. Some discrepancies in the literature on minimal and irreducible pseudocodewords are highlighted and clarified, and a value for the minimal degree cover necessary to realize an LP pseudocodeword is found. Additionally, some conditions for the existence of connected realizations of graph cover pseudocodewords are given. This allows for further analysis of when graph cover pseudocodewords induce computation tree pseudocodewords. Finally, an example is offered that shows that existing theories on the distinction between graph cover pseudocodewords and computation tree pseudocodewords are incomplete.

Index Terms—LDPC codes, linear programming decoding, graph cover decoding, iterative message-passing decoding, pseudocodewords

I. INTRODUCTION

The discovery of turbo codes [2] and the subsequent rediscovery of low-density parity-check (LDPC) codes [6], [13] represent a major milestone in the field of coding theory. These two classes of codes can achieve bit error rates between 10^{-5} and 10^{-12} on the additive white Gaussian noise (AWGN) and related channels with signal-to-noise ratios that are only slightly above the minimum possible for a given channel and code rate established by Shannon’s original capacity theorems [17]. Perhaps the most important commonality between turbo and low density parity check codes is that each utilizes iterative message-passing decoding algorithms for practical decoding of large codes. Thus it is of primary importance to understand the behavior of iterative message-passing decoding algorithms and, in particular, to understand the noncodeword outputs that occur

in computer simulations of LDPC codes with iterative message-passing algorithms.

Motivated by empirical observations of the noncodeword output of LDPC decoders, the notion of stopping sets was first introduced by Forney, *et al.* [5] in 2001. Two years later, a formal definition of stopping sets was given by Changyan, *et al.* [3]. They demonstrated that the bit and block error probabilities of iteratively decoded LDPC codes on the binary erasure channel (BEC) can be determined exactly from the stopping sets of the parity check matrix. Work relating pseudocodewords to stopping sets for the BEC [5], the binary symmetric channel (BSC) and the AWGN channel [9] has revealed a relationship between pseudocodeword weight and stopping set size. However, the current notions of stopping sets and pseudocodewords do not completely characterize the performance and noncodeword outputs of iterative decoders on the BSC and AWGN channels.

In his dissertation [19], Wiberg provides the foundation for analyzing these errors by turning to an analysis of computation trees. Even with these insights, theoretical analyses of the convergence of iterative message-passing decoding have thus far been scarce. (A notable exception is the work done on *density evolution* [15], [16], which considers ensembles of LDPC codes rather than individual codes.) Meanwhile, linear programming (LP) decoding has strong heuristic ties to iterative message-passing decoding by way of graph cover decoding, and its analysis has proven much more attractive from a theoretical standpoint [18]. The common finding across all analyses of these decoders is that pseudocodewords play a significant role in determining convergence of the decoder and in understanding the noncodeword outputs that arise.

The focus of this paper is to further examine the three common notions of pseudocodewords — linear programming pseudocodewords, graph cover pseudocodewords, and computation tree pseudocodewords — and further elucidate relationships between these pseudocodewords. In particular, we examine properties of graph cover pseudocodewords and LP pseudocodewords that allow the translation of findings from this significant body of research to the analysis of the behavior of iterative

This work was supported in part by NSF grant DMS-0602332.

N. Axvig, D. Dreher, K. Morrison and J. L. Walker are with the Department of Mathematics, University of Nebraska, Lincoln, NE 68588-0130, USA {s-naxvig1,s-dturkl,s-kmorri11,jwalker}@math.unl.edu

E. Psota and L. C. Pérez are with the Department of Electrical Engineering, University of Nebraska, Lincoln, NE 68588-0511, USA epsota24@bigred.unl.edu, lperez@unl.edu

message-passing decoders and computation tree pseudocodewords.

The remainder of the paper is organized as follows. In Section II, we give some relevant definitions and terminology. In Section III, we motivate our analysis of pseudocodewords by demonstrating their important role in decoding; Theorem 3.2 expands upon the fact that LP decoding has the *ML-certificate property* [4] by establishing that the fundamental difference between LP decoding and maximum likelihood (ML) decoding is the result of LP pseudocodewords. Additionally in this section we review known characterizations of the sets of LP and graph cover pseudocodewords. Section IV then examines the influence of minimal and irreducible graph cover pseudocodewords and provides a counterexample to an assertion in the literature regarding the equivalence of these two kinds of pseudocodewords. We also establish the value of the minimal degree cover necessary to realize an LP pseudocodeword. Section V turns toward finding graphical realizations of graph cover pseudocodewords that may impact the performance of iterative message-passing decoders. Finally, we connect these graphical realizations back to iterative message-passing decoding in Section VI.

II. BACKGROUND AND MATHEMATICAL PRELIMINARIES

The formal study of pseudocodewords and their role in iterative message-passing decoders necessarily begins with several definitions.

Definition 2.1: A graph G is a pair (V, E) , where V is a nonempty set of elements called *vertices* and E is a (possibly empty) set of elements called *edges*, such that each edge $e \in E$ is assigned an unordered pair of vertices $\{u, v\}$ called the *endpoints* of e . The graph G is *finite* if V is a finite set. The graph G is *simple* if, for each $e \in E$, the two endpoints of E are distinct and, for any two distinct vertices u, v of G , there is at most one edge of e with endpoints $\{u, v\}$.

For the remainder of this paper, we assume our graphs are finite and simple. In particular, we can uniquely identify any edge e with its endpoints, and we write $e = (u, v)$.

Definition 2.2: Let $G = (V, E)$ be a simple graph. For $v \in V$, the *neighborhood* of v is the set $N(v)$ of vertices $u \in V$ such that $(u, v) \in E$. Elements of $N(v)$ are called *neighbors* of v , and the *degree* of v is the number of neighbors v has. We say G is *d-regular* if every vertex in G has degree d . A *path* in G is a finite sequence of distinct vertices v_0, \dots, v_k of G such that v_{i-1} and v_i are neighbors for $1 \leq i \leq k$. A *cycle* in G is a path v_0, \dots, v_k in G with $v_0 = v_k$. We say G is *connected* if, for any two vertices u, v of G , there is a

path $u = v_0, v_1, \dots, v_k = v$ from u to v in G . We say G is *bipartite* if there is a partition $V = X \cup F$ of V into nonempty disjoint sets such that each $e \in E$ has one endpoint in X and the other in F . If G is bipartite, we say it is (d_v, d_c) -*regular* if the degree of every vertex in X is d_v and the degree of every vertex in F is d_c . We say G is a *tree* if G is connected and has no cycles.

The success of low density parity codes stems from the fact that these codes come equipped with a bipartite graph on which the extremely efficient iterative message-passing algorithms operate. This graph is called the *Tanner graph* of the code, a notion whose definition we now recall.

Definition 2.3: A *Tanner graph* is a finite bipartite graph $T = (X \cup F, E)$. We call X the set of *variable nodes* of T and F the set of *check nodes* of T . A (*valid*) *configuration* on a Tanner graph T is an assignment $\mathbf{c} = (c_x)_{x \in X}$ of 0's and 1's to the variable nodes of T such that, at each check node f of T , the binary sum of the values at the neighbors of f is 0. The collection of configurations on a Tanner graph T is called the (*LDPC*) *code determined by T*.

Let $T = (X \cup F, E)$ be a Tanner graph. Since T is finite, we can identify a configuration on T with a vector in \mathbb{F}_2^n , where $n := |X|$. The code C determined by T is the collection of all such vectors, and it is easy to check that this code is linear of length n and dimension at least $n - r$, where $r := |F|$.

Given a binary linear code, i.e., a subspace $C \subseteq \mathbb{F}_2^n$, there is a one-to-one correspondence between Tanner graphs for C and parity check matrices for C . Indeed, if $H = (h_{ji})$ is an $r \times n$ binary matrix, then we associate a Tanner graph $T = T(H) = (X(H) \cup F(H), E(H))$ to H by setting

$$\begin{aligned} X(H) &= \{x_1, \dots, x_n\}, \\ F(H) &= \{f_1, \dots, f_r\}, \quad \text{and} \\ E(H) &= \{(x_i, f_j) \mid h_{ji} = 1\}. \end{aligned}$$

Note that the set of valid configurations on $T(H)$ is precisely the nullspace of H . Conversely, if $T = (\{x_1, \dots, x_n\} \cup \{f_1, \dots, f_r\}, E)$ is a Tanner graph, then we associate a binary $r \times n$ matrix $H = (h_{ji})$ to T , where $h_{ji} = 1$ if and only if $(x_i, f_j) \in E$; note that the kernel of $H(T)$ is precisely the set of valid configurations on T . Since $T = T(H(T))$ for any Tanner graph T and $H = H(T(H))$ for any binary matrix H , these operations give the desired one-to-one correspondence.

A significant problem of practical interest is to transmit a codeword \mathbf{c} of some code C across a noisy channel and then to compute an estimate $\hat{\mathbf{c}}$ based on the channel output. For the remainder of this paper, the codeword is assumed to be transmitted using binary antipodal modulation across the AWGN channel. The process of

computing the estimate $\hat{\mathbf{c}}$ based on the channel output and knowledge of the code is called *decoding*. Decoding can result in the following three outcomes:

- 1) $\hat{\mathbf{c}} = \mathbf{c}$, called a *decoding success*
- 2) $\hat{\mathbf{c}} = \mathbf{c}' \neq \mathbf{c}$, called a *decoding error*
- 3) $\hat{\mathbf{c}} = \mathbf{r}$, where $\mathbf{r} \notin C$, called a *decoding failure*.

When a decoding failure occurs, the decoder is said to have a *noncodeword output* and, depending on the decoder, \mathbf{r} may have binary, rational or real entries.

Wiberg [19] showed that iterative message-passing decoders such as sum-product (SP) and min-sum (MS) actually operate on finite *computation trees* associated to the Tanner graph. As *computation tree pseudocodewords* will be one major focus of this paper, we now make these notions precise.

Definition 2.4 ([19]): Let T be a Tanner graph, and assume an iterative message-passing algorithm has been run on T for a total of m iterations, where a single iteration consists of message-passing from the variable nodes to the check nodes and then back to the variable nodes. The *depth m computation tree* for T with root node v is the tree obtained by tracing the computation of the final cost function of the algorithm at the variable node v of T recursively back through time.

It should be noted that the structure of the computation tree depends upon the particular choice of scheduling used in the iterative message-passing algorithm. However, a computation tree of depth m can always be drawn as a tree with $2m + 1$ levels, labeled from 0 to $2m$, where the 0th level consists only of the root node, each even-numbered level contains only variable nodes, and each odd-numbered level contains only check nodes. Moreover, except for the variable nodes at level $2m + 1$, the computation tree locally looks like the original Tanner graph T : if (x, f) is an edge in T , then every copy of x (above level $2m + 1$) in the computation tree is adjacent to exactly one copy of f and every copy of f in the computation tree is adjacent to exactly one copy of x .

For min-sum and sum-product decoding, the decoder considers as competitors all valid configurations on $n := |X(T)|$ computation trees [19] and outputs a vector whose i^{th} entry is the value assigned to the root node by a minimal cost configuration on a computation tree rooted at variable node x_i ; the precise cost function depends on the particular iterative message-passing decoder chosen, and Wiberg gives explicit definitions for the cost functions for both MS and SP decoding. Note that, for each codeword $\mathbf{c} = (c_1, \dots, c_n)$ and for each computation tree R of T , the assignment of c_i to each copy of x_i in R determines a configuration on R . However, there are also configurations that do not correspond to codewords. This motivates the next definition.

Definition 2.5 ([19]): Let T be a Tanner graph. A *computation tree pseudocodeword* for T is any valid configuration on any computation tree for T . A *nontrivial computation tree pseudocodeword* is a computation tree pseudocodeword that does not correspond to a codeword.

Because iterative message-passing decoders operate on computation trees that, above the bottom level, are locally identical to the original Tanner graph, these decoders do not distinguish between the Tanner graph itself and any finite, unramified cover of the Tanner graph. This intuition leads one to consider *graph cover pseudocodewords*. To make this precise, we first must define what we mean by a *cover* of the Tanner graph.

Definition 2.6: An *unramified cover*, or simply a *cover*, of a finite graph G is a graph \tilde{G} along with a surjective graph homomorphism $\pi : \tilde{G} \rightarrow G$, called a *covering map*, such that for each $v \in V$ and each $\tilde{v} \in \pi^{-1}(v)$, the neighborhood of \tilde{v} is mapped bijectively to the neighborhood of v . For a positive integer M , an *M -cover* of G is cover $\pi : \tilde{G} \rightarrow G$ such that for each vertex v of G , $\pi^{-1}(v)$ contains exactly M vertices of \tilde{G} . If \tilde{G} is an M -cover of G , we say the *degree* of \tilde{G} is M .

Given a Tanner graph T with variable nodes x_1, \dots, x_n and an M -cover $\pi : \tilde{T} \rightarrow T$ of T , we label the elements of $\pi^{-1}(x_i)$ as $x_{i,1}, \dots, x_{i,M}$. The code \tilde{C} determined by \tilde{T} has length nM , and we write a codeword $\tilde{\mathbf{c}} \in \tilde{C}$ in terms of its coordinates as

$$\tilde{\mathbf{c}} = (c_{11}, \dots, c_{1M} : \dots : c_{n1}, \dots, c_{nM}).$$

Definition 2.7: Let T be a Tanner graph for a binary linear code C and let $\tilde{\mathbf{c}} = (c_{11}, \dots, c_{1M} : \dots : c_{n1}, \dots, c_{nM})$ be a codeword in some code \tilde{C} corresponding to some M -cover \tilde{T} of T . The (*unscaled*) *graph cover pseudocodeword* corresponding to $\tilde{\mathbf{c}}$ is the vector

$$\mathbf{p}(\tilde{\mathbf{c}}) = (p_1, \dots, p_n)$$

of nonnegative integers, where, for $1 \leq i \leq n$,

$$p_i = \#\{j \mid c_{ij} = 1\}.$$

The (*normalized*) *graph cover pseudocodeword* corresponding to $\tilde{\mathbf{c}}$ is the vector

$$\omega(\tilde{\mathbf{c}}) = \frac{1}{M} \mathbf{p}(\tilde{\mathbf{c}}).$$

A *nontrivial graph cover pseudocodeword* is a graph cover pseudocodeword that is not a codeword.

Intuitively, all codewords on all covers of the Tanner graph are competitors in iterative message-passing decoding algorithms. In this vein, Vontobel and Koetter [18] define *graph cover decoding*; this decoder simultaneously considers all codewords on all covers of the Tanner graph and then returns the normalized graph cover pseudocodeword corresponding to the one which, in a certain precise sense, provides the best explanation of

the channel output. They show that graph cover decoding is equivalent to *linear programming (LP) decoding*, as defined by Feldman [4]. We now turn to the formal definition of LP decoding.

Definition 2.8 ([4]): Let $H = (h_{j,i})$ be the $r \times n$ parity check matrix with corresponding Tanner graph T , and, for $1 \leq j \leq r$, set

$$N(j) = \{i \mid h_{j,i} = 1\} \subseteq \{1, \dots, n\}$$

so that $N(j)$ is the set of variable nodes adjacent to check node j in T . The *fundamental polytope* $\mathcal{P} = \mathcal{P}(H)$ is the subset of the unit hypercube $[0, 1]^n \subset \mathbb{R}^n$ consisting of all vectors $\mathbf{x} = (x_1, \dots, x_n)$ such that for $1 \leq i \leq n$, $1 \leq j \leq r$, and each subset $S \subseteq N(j)$ with $|S|$ odd, we have

$$\sum_{i \in S} x_i + \sum_{i \in N(j) \setminus S} (1 - x_i) \leq |N(j)| - 1.$$

For a given vector of log-likelihoods $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_n)$ determined by the channel output and for any $\mathbf{x} = (x_1, \dots, x_n) \in \mathbb{R}^n$, the *cost* $\gamma(\mathbf{x})$ of \mathbf{x} is given by

$$\gamma(\mathbf{x}) = \boldsymbol{\lambda} \cdot \mathbf{x} = \sum_{i=1}^n \lambda_i x_i.$$

Linear programming (LP) decoding is defined to be the task of minimizing $\gamma(\mathbf{x})$ over all $\mathbf{x} \in \mathcal{P}$.

Since the cost function is linear and the polytope is defined by linear inequalities, the output of linear programming decoding may always be taken to be a vertex of the fundamental polytope. Feldman [4] shows that a vector in $\{0, 1\}^n$ is a vertex of the fundamental polytope if and only if it is a codeword. This motivates the following definition.

Definition 2.9: A *linear programming pseudocodeword* of a code defined by the parity check matrix H is any vertex of the fundamental polytope $\mathcal{P}(H)$. A *nontrivial linear programming pseudocodeword* is a linear programming pseudocodeword that is not a codeword.

Vontobel and Koetter [18] show that the collection of rational points in the fundamental polytope is precisely the collection of graph cover pseudocodewords. Thus, with the definitions here, every linear programming pseudocodeword is a normalized graph cover pseudocodeword, but not vice versa.

III. CONNECTIONS BETWEEN LP AND ML DECODING

Feldman shows in [4] that linear programming decoding has the *ML-certificate property*, which guarantees that if LP decoding returns a codeword, this codeword is an ML codeword; however, this does not imply that LP decoding is equivalent to ML decoding in general, as is illustrated by Theorem 3.2 in the special case of

the AWGN channel. One might conjecture that these two decoders can give different outputs because ML decoding only considers codewords as outputs, whereas if the fundamental polytope contains nontrivial pseudocodewords, then the LP decoder considers these in addition to the codewords. In the case of the AWGN channel, however, we find that the difference between LP and ML decoding goes deeper than the set of configurations that each considers. To expand on this, we first describe a reasonable extension of the ML decoder over the AWGN channel that considers all vertices of the fundamental polytope.

Over the additive white Gaussian noise channel, maximum likelihood decoding is based on squared Euclidean distance between modulated points, where the modulation map is given by $\mathbf{m}(\mathbf{x}) = 2\mathbf{x} - \mathbf{1}$ with $\mathbf{1} = (1, \dots, 1)$ [12]. For a received vector $\mathbf{y} \in \mathbb{R}^n$, the output $\hat{\mathbf{x}}^{ML}$ of ML decoding is given by

$$\hat{\mathbf{x}}^{ML} = \mathbf{m}^{-1} \left(\underset{\mathbf{x} \in \mathbf{m}(C)}{\operatorname{argmin}} \sum_{i=1}^n (y_i - x_i)^2 \right),$$

where n is the length of the code. Let $\mathcal{V} = \mathcal{V}(\mathcal{P})$ denote the set of vertices of the fundamental polytope. Then a natural extension of the ML decision rule in this case is

$$\hat{\mathbf{x}}^{GML} = \mathbf{m}^{-1} \left(\underset{\mathbf{v} \in \mathbf{m}(\mathcal{V})}{\operatorname{argmin}} \sum_{i=1}^n (y_i - v_i)^2 \right) \quad (\text{III.1})$$

$$= \mathbf{m}^{-1} \left(\underset{\mathbf{v} \in \mathbf{m}(\mathcal{V})}{\operatorname{argmin}} \sum_{i=1}^n (y_i^2 - 2y_i v_i + v_i^2) \right), \quad (\text{III.2})$$

which considers all vertices of the polytope, instead of just the codewords.

We compare these notions of the maximum likelihood decoder and the generalized maximum likelihood (GML) decoder to linear programming decoding in Theorem 3.2. The proof of this proposition requires a result found in [10].

Proposition 3.1 ([10], Proposition 2.12): Let C be the code determined by the Tanner graph T with n variable nodes. Suppose that $\boldsymbol{\omega}$ is a nontrivial LP pseudocodeword of C . Then, for some vector $\boldsymbol{\lambda}$ of log-likelihood ratios, the cost $\sum_{i=1}^n \lambda_i \omega_i$ is smaller than the cost $\sum_{i=1}^n \lambda_i c_i$ of any codeword $\mathbf{c} \in C$.

Theorem 3.2: Suppose the code C defined by the parity check matrix H is used on the additive white Gaussian noise channel. Then the following are equivalent:

- 1) C has no nontrivial linear programming pseudocodewords with respect to H .
- 2) Linear programming decoding for C with respect to H is equivalent to maximum likelihood decoding for C .
- 3) Linear programming decoding for C with respect to H and generalized maximum likelihood decod-

ing for C with respect to H , as in Equation (III.2), use the same decision rule.

Proof: As before, let $\mathcal{V} = \mathcal{V}(\mathcal{P})$ be the set of vertices of the fundamental polytope. We consider the set $\mathfrak{m}(\mathcal{V})$ of all n -dimensional vertices of the modulated fundamental polytope $\mathfrak{m}(\mathcal{P})$ as possible outputs of ML decoding, GML decoding and LP decoding. This is not a natural setting for the LP decoder, which considers only vectors in some subset of $[0, 1]^n$. However, the decision rule for the LP decoder is to output

$$\hat{\mathbf{x}}^{LP} = \operatorname{argmin}_{\mathbf{w} \in \mathcal{V}} \sum_{i=1}^n \lambda_i w_i,$$

and since the modulation map is coordinate-wise linear and strictly increasing, we have

$$\operatorname{argmin}_{\mathbf{w} \in \mathcal{V}} \sum_{i=1}^n \lambda_i w_i = \mathfrak{m}^{-1} \left(\operatorname{argmin}_{\mathbf{v} \in \mathfrak{m}(\mathcal{V})} \sum_{i=1}^n \lambda_i v_i \right).$$

The log-likelihood ratios for a given a received vector \mathbf{y} on the AWGN channel with noise variance σ^2 are given by

$$\begin{aligned} \lambda_i &= \ln \left(\frac{\Pr[y_i | x_i = 0]}{\Pr[y_i | x_i = 1]} \right) \\ &= \ln \left(\frac{\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i+1)^2}{2\sigma^2}}}{\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i-1)^2}{2\sigma^2}}} \right) \\ &= \ln \left(e^{\frac{-4y_i}{2\sigma^2}} \right) \\ &= \frac{-2y_i}{\sigma^2}. \end{aligned}$$

Hence the decision rule for LP decoding can be reformulated as

$$\begin{aligned} \hat{\mathbf{x}}^{LP} &= \mathfrak{m}^{-1} \left(\operatorname{argmin}_{\mathbf{v} \in \mathfrak{m}(\mathcal{V})} \sum_{i=1}^n \frac{-2y_i}{\sigma^2} v_i \right) \\ &= \mathfrak{m}^{-1} \left(\operatorname{argmin}_{\mathbf{v} \in \mathfrak{m}(\mathcal{V})} \sum_{i=1}^n -2y_i v_i \right), \end{aligned}$$

since σ is independent of $\mathbf{v} \in \mathfrak{m}(\mathcal{V})$. Finally, notice that adding y_i^2 to the i^{th} entry inside the sum will not change the minimization, and thus

$$\hat{\mathbf{x}}^{LP} = \mathfrak{m}^{-1} \left(\operatorname{argmin}_{\mathbf{v} \in \mathfrak{m}(\mathcal{V})} \sum_{i=1}^n (y_i^2 - 2y_i v_i) \right). \quad (\text{III.3})$$

If C has no nontrivial linear programming pseudocodewords with respect to H , then $\mathfrak{m}(\mathcal{V}) = \mathfrak{m}(C)$ and $\sum_{i=1}^n v_i^2 = n$, since $v_i = \pm 1$ for all $\mathbf{v} \in \mathfrak{m}(C)$. Thus, adding v_i^2 to the i^{th} term in the sum does not change the

minimization problem, and

$$\begin{aligned} \hat{\mathbf{x}}^{LP} &= \mathfrak{m}^{-1} \left(\operatorname{argmin}_{\mathbf{v} \in \mathfrak{m}(\mathcal{V})} \sum_{i=1}^n (y_i^2 - 2y_i v_i + v_i^2) \right) \\ &= \mathfrak{m}^{-1} \left(\operatorname{argmin}_{\mathbf{v} \in \mathfrak{m}(C)} \sum_{i=1}^n (y_i^2 - 2y_i v_i + v_i^2) \right), \end{aligned}$$

which is the same decision rule used to compute $\hat{\mathbf{x}}^{GML}$ in generalized maximum likelihood decoding and $\hat{\mathbf{x}}^{ML}$ in maximum likelihood decoding. Thus, when C has no nontrivial LP pseudocodewords with respect to H , we have that LP decoding is equivalent to both ML decoding and GML decoding.

Conversely, if C has a nontrivial linear programming pseudocodeword with respect to H , then there is a vector $\boldsymbol{\lambda}$ of log-likelihood ratios such that $\sum_{i=1}^n \lambda_i \omega_i$ is smaller than the cost $\sum_{i=1}^n \lambda_i c_i$ of any codeword $\mathbf{c} \in C$, by Proposition 3.1. Over the AWGN channel the received vector may be any vector in \mathbb{R}^n , so we can construct a received vector \mathbf{y} such that the resulting log-likelihood vector is $\boldsymbol{\lambda}$. Thus, if \mathbf{y} is received, LP decoding will return a nontrivial pseudocodeword, whereas ML decoding will always return a codeword. Therefore, LP and ML are not equivalent. Furthermore, because C has a nontrivial LP pseudocodeword with respect to H , $\sum_{i=1}^n v_i^2$ is not constant over $\mathbf{v} \in \mathfrak{m}(\mathcal{V})$, and hence the LP decision rule as given in Equation III.3 differs from the GML decision rule as given in Equation III.2. Therefore, LP and GML are not equivalent either. ■

Remark 3.3: The proof of Theorem 3.2 cannot be applied to show similar results about when linear programming decoding is not equivalent to maximum likelihood decoding or generalized maximum likelihood decoding for the binary symmetric channel or the binary erasure channel because the set of possible received vectors for either of these channels is finite, and hence the set of possible log-likelihood ratios is also finite. Given a target vector of log-likelihood ratios $\boldsymbol{\lambda} \in \mathbb{R}^n$ it is impossible to guarantee that there is a received vector which yields $\boldsymbol{\lambda}$. For more discussion on this issue, see [14].

Theorem 3.2 shows that linear programming decoding differs from maximum likelihood decoding due to the presence of nontrivial pseudocodewords. Furthermore, as mentioned above, Vontobel and Koetter [18] show that LP decoding and graph cover decoding are equivalent; hence, Theorem 3.2 illustrates the difference between graph cover decoding and ML decoding as well. It is therefore essential to better understand the properties of pseudocodewords of these decoders in order to further analyze the noncodeword decoder errors of these decoders and iterative message-passing decoding.

One characterization of graph cover pseudocodewords is given by Koetter, Li, Vontobel, and Walker [11]: a

vector \mathbf{p} of nonnegative integers is an unscaled graph cover pseudocodeword if and only if it reduces modulo 2 to a codeword and it lies within the *fundamental cone* $\mathcal{K} \subseteq \mathbb{R}^n$ where

$$\mathcal{K} = \mathcal{K}(H) = \left\{ (v_1, \dots, v_n) \in \mathbb{R}^n \mid \begin{array}{l} v_i \geq 0 \text{ for all } i, \\ \sum_{i' \neq i} h_{ji'} v_{i'} \geq h_{ji} v_i \text{ for all } i, j \end{array} \right\}.$$

Vontobel and Koetter provide a different characterization of graph cover pseudocodewords in [18]: a vector $\omega = (\omega_1, \dots, \omega_n)$ of rational numbers between zero and one lies in the fundamental polytope if and only if it is a normalized graph cover pseudocodeword. It is worthwhile to note that the fundamental cone is the conic hull of the fundamental polytope [18].

IV. MINIMAL AND IRREDUCIBLE PSEUDOCODEWORDS

To further examine the impact of graph cover pseudocodewords, we will mimic the consideration in the classical coding case of *minimal codewords*, i.e., codewords whose supports do not properly contain the support of any non-zero codeword. In particular, we examine different extensions of the theory of minimal codewords to graph cover (and hence LP) pseudocodewords.

Definition 4.1 ([18]): A *minimal pseudocodeword* is a pseudocodeword \mathbf{p} such that $\{\alpha\mathbf{p} \mid \alpha \in \mathbb{R}, \alpha \geq 0\}$ is an edge of the fundamental cone.

Here, by *edges of the fundamental cone*, we mean a set of half-rays through the origin whose conic hull is the fundamental cone, with the property that no proper subset of this set has the fundamental cone as its conic hull.

A similar generalization is presented in [8]:

Definition 4.2 ([8]): An unscaled pseudocodeword is *irreducible* if it cannot be written as a (nontrivial) sum of other unscaled pseudocodewords.

Note that while a *minimal pseudocodeword* can refer to either a normalized or unscaled pseudocodeword, an *irreducible pseudocodeword* can only refer to an unscaled pseudocodeword.

Remark 4.3: If an irreducible pseudocodeword \mathbf{p} is actually a codeword, then \mathbf{p} cannot be written as a (nontrivial) sum of codewords and we will call \mathbf{p} an *irreducible codeword*. With this terminology, irreducible codewords coincide precisely with minimal codewords. Additionally, if \mathbf{p} is irreducible as a codeword, then \mathbf{p} is also irreducible as a pseudocodeword because if \mathbf{p} were the sum of pseudocodewords then each of those pseudocodewords must consist of only zeros and ones and thus must be codewords themselves [11], which contradicts the irreducibility of \mathbf{p} . Hence a vector \mathbf{p} is an irreducible codeword if and only if it is a minimal

codeword, if and only if it is a trivial irreducible pseudocodeword.

It is important to note that although the terms *irreducible pseudocodeword* and *minimal pseudocodeword* are sometimes used interchangeably, the notions do not necessarily coincide. If \mathbf{p} is a minimal pseudocodeword, then $2\mathbf{p}$ is also a minimal pseudocodeword but it is not irreducible. Conversely, irreducible pseudocodewords may not be minimal, as seen in the next example.

Example 4.4: Let C be the null space of

$$H = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

so that $C = \{(0, 0, 0, 0), (1, 1, 1, 1)\}$. The fundamental cone is given by

$$\mathcal{K}(H) = \left\{ (x_1, x_2, x_3, x_4) \in \mathbb{R}^4 \mid \begin{array}{l} x_i \geq 0 \text{ for all } i, \\ x_2 = x_3 = x_4, \text{ and } 3x_2 \geq x_1 \end{array} \right\}.$$

The cone can be seen as a two-dimensional cone embedded in \mathbb{R}^4 , and the edges are the half-rays $\{\alpha(3, 1, 1, 1) \mid \alpha \in \mathbb{R}_{\geq 0}\}$ and $\{\alpha(0, 2, 2, 2) \mid \alpha \in \mathbb{R}_{\geq 0}\}$.

If $(1, 1, 1, 1) = \mathbf{x} + \mathbf{y}$ with \mathbf{x} and \mathbf{y} being nonzero nonnegative integer vectors, then \mathbf{x} must have at least one coordinate which is 0 and one coordinate which is 1, and hence is not a pseudocodeword since it will not reduce modulo 2 to a codeword [11]. This means that $(1, 1, 1, 1)$ is an irreducible pseudocodeword, but it is not a minimal pseudocodeword, since it does not lie on an edge of the fundamental cone. Additionally, since $(1, 1, 1, 1)$ is an irreducible pseudocodeword that is also a codeword, it is actually a minimal codeword, even though it is not a minimal pseudocodeword.

Thus, we see that while the notions of minimal and irreducible pseudocodewords may prove important in different contexts, such as in determining which pseudocodewords are more likely to cause errors [8], the conflation of these terms is inaccurate and may hinder further analysis of the set of graph cover pseudocodewords.

As mentioned above, Kelley and Sridhara [8] suggest that irreducible pseudocodewords are more likely than other pseudocodewords to cause linear programming/graph cover decoding to fail to converge. Motivated by this, they examine bounds on the smallest degree cover needed to realize an irreducible pseudocodeword. With this question in mind, we make the following definition.

Definition 4.5: A normalized graph cover pseudocodeword ω for the Tanner graph T is *minimally realizable* on the cover \tilde{T} of T if there is a configuration $\tilde{\mathbf{c}}$ on \tilde{T} such that

- 1) $\omega = \omega(\tilde{\mathbf{c}})$, and
- 2) whenever ω has a realization on an N -cover of T , we have $M \leq N$, where M is the degree of \tilde{T} .

We find an exact value for the degree of a minimal realization of a normalized graph cover pseudocodeword Proposition 4.7 below, under the assumption we are given the coordinates of the graph over pseudocodeword as a point in Feldman's *extended polytope* [4], rather than simply in the fundamental polytope. We first recall the definition of the extended polytope.

Definition 4.6 ([4]): Let $H = (h_{j,i})$ be an $r \times n$ parity check matrix and let $\mathcal{P} = \mathcal{P}(H)$ be the fundamental polytope of H , as described in Definition 2.8 above. For $1 \leq j \leq r$, set

$$E_j = \{S \subseteq N(j) : |S| \text{ is even}\},$$

where, as before,

$$N(j) = \{i \mid h_{j,i} = 1\} \subseteq \{1, \dots, n\},$$

and write $e_j = |E_j|$. Label the coordinates of $\mathbb{R}^{n+e_1+\dots+e_r}$ as $\{1, \dots, n\} \cup \{w_{j,S} \mid 1 \leq j \leq r \text{ and } S \in E_j\}$, and let $E_j(i)$ be the subset of E_j consisting of those even-sized subsets of $N(j)$ that contain i . The j^{th} *local extended polytope* of H is the polytope

$$\mathcal{Q}_j(H) = \left\{ \left(\mathbf{x} \in \mathcal{P} \begin{array}{l} w_{j',S} = 0 \text{ if } j' \neq j \\ \sum_{S \in E_j} w_{j,S} = 1 \\ x_i = \sum_{S \in E_j(i)} w_{j,S} \end{array} \right) \mid (\mathbf{x} | \mathbf{w}) \in [0, 1]^{n+e_1+\dots+e_r} \right\}$$

and the *extended polytope* of H is the polytope

$$\mathcal{Q} = \mathcal{Q}(H) = \bigcap_{1 \leq j \leq r} \mathcal{Q}_j(H) \subseteq [0, 1]^{n+e_1+\dots+e_r}.$$

We define a *minimal realization* of a point in the extended polytope \mathcal{Q} in an analogous fashion to Definition 4.5 above.

Proposition 4.7: Let $\bar{\omega}$ be a rational point in the extended polytope and let M be the degree of a minimal realization of $\bar{\omega}$. Then M is the smallest positive integer such that each coordinate of $M\bar{\omega}$ is a non-negative integer.

Proof: For any positive integer t such that $t\bar{\omega}$ is a vector of integers, Feldman [4] gives a construction that yields a realization of $t\bar{\omega}$. We will show that this realization occurs on a t -cover.

Since

$$\sum_{S \in E_j} w_{j,S} = 1,$$

we have

$$\sum_{S \in E_j} a_{j,S} = t,$$

where each $a_{j,S} := tw_{j,S}$ is an integer. Feldman's construction gives that for each $S \in E_j$, there are $a_{j,S}$ copies of check node j which are satisfied via the

configuration S . This constraint implies that there are t total copies of check node j , i.e., that the realization occurs on a t -cover.

By hypothesis, $\bar{\omega}$ is realizable on an M -cover so $M\bar{\omega}$ must be a vector of integers. Furthermore, M must be the smallest number such that $M\bar{\omega}$ is a vector of integers, since if this held for some $t < M$ then, by the argument above, $\bar{\omega}$ would be realizable on a t -cover, contradicting the minimality of M . ■

Proposition 4.7 can be extended to describe the minimum degree realization of any vector ω with rational entries in the fundamental polytope whenever we can construct a corresponding $\bar{\omega}$ in the extended polytope. Feldman [4] establishes that such an $\bar{\omega}$ always exists, and Vontobel and Koetter [18] give a method for constructing $\bar{\omega}$ under particular circumstances, namely when it is already known how to express ω as a convex linear combination of vectors in \mathbb{F}_2^n that satisfy a given check node.

V. REALIZING PSEUDOCODEWORDS ON CONNECTED GRAPH COVERS

In addition to examining minimal realizations of pseudocodewords, we must also explore when connected realizations of normalized pseudocodewords exist. Connectivity of the cover is vital in order to analyze the relationship between LP/graph cover decoding and iterative message-passing decoding algorithms, because the latter operate on computation trees, which are inherently connected. The following example illustrates that not every graph cover pseudocodeword can be realized on a connected cover, and thus not all graph cover pseudocodewords may influence iterative message-passing decoders.

Example 5.1: Consider the Tanner graph T which is an 8-cycle with vertices alternating between being check nodes and variable nodes. The code determined by T is the binary $[4, 1, 4]$ repetition code, with parity check matrix

$$H = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix},$$

and the fundamental polytope is

$$\mathcal{P} = \mathcal{P}(H) = \{(\omega, \omega, \omega, \omega) \in \mathbb{R}^4 \mid 0 \leq \omega \leq 1\}.$$

The only connected covers of T are $8M$ -cycles for $M \geq 1$, so the only unscaled graph cover pseudocodewords that have connected realizations are those of the form $(0, 0, 0, 0)$ and (M, M, M, M) for $M \geq 1$. Thus the only normalized graph cover pseudocodewords with connected realizations are $(0, 0, 0, 0)$ and $(1, 1, 1, 1)$. In

particular, no rational point of \mathcal{P} that is not a vertex of \mathcal{P} has a connected graph cover realization.

Although Example 5.1 shows that there are situations in which some points in the interior of the polytope cannot be realized on a connected cover of the original Tanner graph, we know that linear programming decoding (and hence graph cover decoding) will always output a vertex of the fundamental polytope. In Example 5.1, these vertices do have connected realizations. This phenomenon happens in general, as shown by the next proposition which originally appeared in the conference paper [1].

Proposition 5.2 ([1], Proposition 3.3): Let T be a Tanner graph with corresponding fundamental polytope \mathcal{P} . Suppose ω is a vertex of \mathcal{P} , and let $(\tilde{T}, \tilde{\mathbf{c}})$ be a realization of ω . Let $\tilde{T}_1, \dots, \tilde{T}_k$ be the connected components of \tilde{T} , so that \tilde{T}_i is an M_i -cover of T for $1 \leq i \leq k$, with $M_1 + \dots + M_k = M$, and $\tilde{\mathbf{c}} = (\tilde{\mathbf{c}}_1 | \dots | \tilde{\mathbf{c}}_k)$, where $\tilde{\mathbf{c}}_i$ is a configuration on \tilde{T}_i . Then $(\tilde{T}_i, \tilde{\mathbf{c}}_i)$ is a connected realization of ω for $i = 1, \dots, k$. In other words, every graph cover realization of ω is either connected or the disjoint union of connected graph cover realizations of ω .

Proof: Set $\alpha_i = \omega(\tilde{\mathbf{c}}_i)$ for $1 \leq i \leq k$. Then, looking at the unscaled graph cover pseudocodewords, we have

$$M\omega = M_1\alpha_1 + \dots + M_k\alpha_k.$$

Dividing through by M gives

$$\omega = \frac{M_1}{M}\alpha_1 + \dots + \frac{M_k}{M}\alpha_k.$$

Since $\frac{M_i}{M} \geq 0$ for each i and

$$\frac{M_1}{M} + \dots + \frac{M_k}{M} = \frac{M_1 + \dots + M_k}{M} = \frac{M}{M} = 1,$$

we have written ω as a convex combination of $\alpha_1, \dots, \alpha_k$. But each α_i is in \mathcal{P} by [18] and so each $\frac{M_i}{M}\alpha_i$ is too since $\frac{M_i}{M} \leq 1$. Since ω is a vertex of the polytope, this forces each α_i to lie on the line segment from the origin to ω , i.e., $\alpha_i = \gamma_i\omega$ for some rational numbers $0 < \gamma_i \leq 1$. So we have

$$M\omega = (M_1\gamma_1 + \dots + M_k\gamma_k)\omega,$$

which means $M_1 + \dots + M_k = M = M_1\gamma_1 + \dots + M_k\gamma_k$. Hence $\gamma_i = 1$ for each i , i.e., $\alpha_i = \omega$ for all i . ■

Theorem 5.4 below gives another sufficient condition for connected realizations of graph cover pseudocodewords to exist. The next lemma will be used in the proof.

Lemma 5.3: Let T be a Tanner graph, let T' be a spanning tree of T , and suppose e_1, \dots, e_t are edges in T not in T' . Let $\pi: \tilde{T} \rightarrow T$ be any finite connected cover of T , and let $\tilde{e}_i \in \pi^{-1}(e_i)$ be a fixed lift of e_i to \tilde{T} for each $i = 1, \dots, t$. Then $\tilde{T} - \{\tilde{e}_1, \dots, \tilde{e}_t\}$ is connected.

Proof: Let notation be as in the statement of the lemma for $i = 1, \dots, t$. Notice that to show that $\tilde{T} - \{\tilde{e}_1, \dots, \tilde{e}_t\}$ is connected, it suffices to show that there is a path in $\tilde{T} - \{\tilde{e}_1, \dots, \tilde{e}_t\}$ from \tilde{x} to \tilde{f} for any $\tilde{e}_i = \tilde{x}\tilde{f}$. So fix i and let $\tilde{e}_i = \tilde{x}\tilde{f}$.

Since T' is a spanning tree for T , there is a path p on T' from $x = \pi(\tilde{x})$ to $f = \pi(\tilde{f})$. Then pe_i is a cycle on T containing x and f . By [7, Theorem 2.4.3], $\pi^{-1}(pe_i)$ consists of a disjoint union of cycles which project onto pe_i . Since $\tilde{e}_i \in \pi^{-1}(e_i) \subset \pi^{-1}(pe_i)$, there is a cycle Γ in $\pi^{-1}(pe_i)$ containing \tilde{e}_i . Since Γ projects onto pe_i and p is contained in T' , we see that $\pi(\Gamma)$ does not contain e_j for any $j \neq i$ and hence Γ does not contain \tilde{e}_j for any $j \neq i$. Thus, $\Gamma - \{\tilde{e}_1, \dots, \tilde{e}_t\} = \Gamma - \tilde{e}_i$ is still connected, and so $\Gamma - \tilde{e}_i$ contains a path from \tilde{x} to \tilde{f} in $\tilde{T} - \{\tilde{e}_1, \dots, \tilde{e}_t\}$. ■

Theorem 5.4: Let T be a Tanner graph with average variable node degree a_v and average check node degree a_c . Suppose that either $a_v \geq 3$ and $a_c \geq 3$, or $a_v \geq 2$ and $a_c \geq 4$. Then any rational point in the fundamental polytope of T can be minimally realized on a connected cover.

Proof: Let ω be a rational point in the fundamental polytope of T . Then ω is a normalized graph cover pseudocodeword [18] and so is minimally realizable on an M -cover for some M . Let $(\tilde{T}, \tilde{\mathbf{c}})$ be a realization on an M -cover with a minimal number of connected components. By way of contradiction, suppose \tilde{T} is not connected, and let $\pi_R: \tilde{R} \rightarrow T$ and $\pi_S: \tilde{S} \rightarrow T$ be distinct connected components of \tilde{T} . We will give an algorithm for connecting these two components \tilde{R} and \tilde{S} , demonstrating that ω is, in fact, realizable on a cover with fewer connected components.

Let T' be any spanning tree of T . We will first show that there exists a check node $f \in F(T)$ that is incident to at least two edges not on T' . Assume, for the purpose of contradiction, that every check node is incident to at most one edge that is not on T' . If the check nodes are $\{f_1, \dots, f_r\}$, then we see the number of edges on T' is at least

$$\sum_{i=1}^r (\deg(f_i) - 1) = \left(\sum_{i=1}^r \deg(f_i) \right) - r \quad (\text{V.1})$$

$$= ra_c - r \quad (\text{V.2})$$

$$= r(a_c - 1), \quad (\text{V.3})$$

since the sum of the check node degrees must be the total number of edges in T , which is equal to ra_c .

On the other hand, if there are n variable nodes and r check nodes on T , then T has $n + r$ vertices and so the number of edges on any spanning tree for T is $n + r - 1$. The number of edges in T is $na_v = ra_c$, so $n = r \frac{a_c}{a_v}$

and we have that the number of edges on T' is

$$r \frac{a_c}{a_v} + r - 1 = r \left(\frac{a_c}{a_v} + 1 \right) - 1.$$

Putting this together with the bound on the number of edges in T' given in (V.3), we see that

$$r \left(\frac{a_c}{a_v} + 1 \right) - 1 \geq r(a_c - 1).$$

Thus,

$$r \left(\frac{a_c}{a_v} + 1 \right) > r \left(\frac{a_c}{a_v} + 1 \right) - 1 \geq r(a_c - 1),$$

and so

$$\frac{a_c}{a_v} + 1 > a_c - 1.$$

Rearranging this we see

$$2 > a_c \left(1 - \frac{1}{a_v} \right).$$

In the case where $a_c, a_v \geq 3$, we have $1 - \frac{1}{a_v} \geq \frac{2}{3}$, hence

$$2 > \frac{2}{3} a_c,$$

which implies that $3 > a_c$. By assumption, $a_c \geq 3$, so we have a contradiction. A similar contradiction is reached in the case where $a_v \geq 2$ and $a_c \geq 4$.

Thus, there is some check node f of T such that at least two edges $e = (f, x)$, $e' = (f, x')$ incident to f are not on T' . Let $\tilde{e}_R = (f_R, \tilde{x}_R)$ and $\tilde{e}'_R = (f_R, \tilde{x}'_R)$ be fixed lifts of e and e' , respectively, to \tilde{R} and let $\tilde{e}_S = (f_S, \tilde{x}_S)$ and $\tilde{e}'_S = (f_S, \tilde{x}'_S)$ be fixed lifts of e and e' , respectively, to \tilde{S} . By Lemma 5.3, $\tilde{R} - \{\tilde{e}_R, \tilde{e}'_R\}$ and $\tilde{S} - \{\tilde{e}_S, \tilde{e}'_S\}$ are connected.

For any variable node \tilde{v} of \tilde{T} , let $\tilde{\mathbf{c}}(\tilde{v})$ denote the bit assignment \tilde{v} receives from the configuration $(\tilde{T}, \tilde{\mathbf{c}})$. If $\tilde{\mathbf{c}}(\tilde{x}_R) = \tilde{\mathbf{c}}(\tilde{x}_S)$, then crossing the edges \tilde{e}_R and \tilde{e}_S does not change the check sum at f_R or f_S and results in \tilde{R} and \tilde{S} becoming a single connected component. In other words, $(\tilde{T} - \{\tilde{e}_R, \tilde{e}_S\} + \{f_R \tilde{x}_S, f_S \tilde{x}_R\}, \tilde{\mathbf{c}})$ is a minimal realization of ω with fewer components than \tilde{T} , a contradiction. We get a similar contradiction if $\tilde{\mathbf{c}}(\tilde{x}'_R) = \tilde{\mathbf{c}}(\tilde{x}'_S)$.

Finally, assume $\tilde{\mathbf{c}}(\tilde{x}_R) \neq \tilde{\mathbf{c}}(\tilde{x}_S)$ and $\tilde{\mathbf{c}}(\tilde{x}'_R) \neq \tilde{\mathbf{c}}(\tilde{x}'_S)$. Then $(\tilde{\mathbf{c}}(\tilde{x}_R) + \tilde{\mathbf{c}}(\tilde{x}'_S)) \equiv (\tilde{\mathbf{c}}(\tilde{x}_S) + \tilde{\mathbf{c}}(\tilde{x}'_R)) \pmod{2}$. This means that crossing both \tilde{e}_R with \tilde{e}_S and \tilde{e}'_R with \tilde{e}'_S does not change the binary check sum at f_R or at f_S , and again results in \tilde{R} and \tilde{S} becoming a single connected component, i.e., $(\tilde{T} - \{\tilde{e}_R, \tilde{e}'_R, \tilde{e}_S, \tilde{e}'_S\} + \{f_R \tilde{x}_S, f_R \tilde{x}'_S, f_S \tilde{x}_R, f_S \tilde{x}'_R\}, \tilde{\mathbf{c}})$ is a minimal realization of ω on a cover with fewer connected components than \tilde{T} , a contradiction.

Since, in any case, assuming \tilde{R} and \tilde{S} are distinct connected components of \tilde{T} leads to a minimal realization of

ω on a cover with fewer connected components than \tilde{T} , there must be a minimal realization of ω on a connected cover of T . ■

VI. GRAPH COVERS AND COMPUTATION TREES

While intuitive links between linear programming/graph cover decoding and iterative message-passing decoding have been proposed, the only proven analysis of iterative message-passing decoding on finite-length LDPC codes hails from the fundamental work of Wiberg [19]. He establishes that iterative message-passing algorithms actually work by finding minimal cost configurations on computation trees, so it is essential to further examine computation tree pseudocodewords to gain a more precise understanding of the errors that arise in iterative message-passing decoding. To make use of the body of literature on graph cover pseudocodewords, though, we examine potential connections between graph cover pseudocodewords (including linear programming pseudocodewords) and computation tree pseudocodewords. In particular, every graph cover pseudocodeword that has a connected realization induces a computation tree pseudocodeword. It is not known, however, whether every computation tree configuration is the result of a truncated graph cover configuration. Thus there may be computation tree pseudocodewords that cause errors in iterative message-passing decoding that are unrelated to graph cover configurations. This may yield one explanation for the inconsistent behavior of min-sum decoding versus LP/graph cover decoding observed across simulations [1]. Kelley and Sridhara [8] give a characterization of computation tree pseudocodewords that arise from graph cover pseudocodewords. As we will see, this characterization is not complete. We first give a brief review the work of Kelley and Sridhara [8] in this direction, starting with the notion of a *consistent* configuration on a computation tree for the Tanner graph T .

Definition 6.1 ([8]): Let R be a computation tree of T and let \mathbf{c} be a configuration on R . For a variable node $x \in X(T)$ and for a check node $f \in N(x)$, define the *local assignment of x at f by the configuration \mathbf{c}* , denoted $L_{\mathbf{c}}(x, f)$, to be the average of the values \mathbf{c} assigns to all copies of x in R that are adjacent to a copy of f in R . The configuration \mathbf{c} is called *consistent* if, for each $x \in X(T)$, we have $L_{\mathbf{c}}(x, f_i) = L_{\mathbf{c}}(x, f_j)$ for all $f_i, f_j \in N(x)$.

Suppose that R is a computation tree of T that contains at least one copy of each check node of T , and suppose that T has variable nodes x_1, \dots, x_n . A consistent configuration \mathbf{c} on R gives rise to a vector $\omega \in [0, 1]^n$, where $\omega_i = L_{\mathbf{c}}(x_i, f)$ for arbitrary $f \in N(x_i)$. Note that ω_i is well-defined by the consistency of \mathbf{c} [8]. The consistency

of \mathbf{c} also gives us that ω satisfies each of the local check constraints in the fundamental polytope, since $L_{\mathbf{c}}(x, f)$ comes from an average of valid local configurations on $N(f)$ for any check node f that is adjacent to x . As the fundamental polytope is the intersection of all vectors satisfying all of the local check constraints, we see that ω is an element of the fundamental polytope, and hence realizable on a finite cover of T [8]. An example of a valid configuration on a computation tree that is not consistent is shown in Example 6.2 below.

Example 6.2 (See also [8]): Let T be the Tanner graph of Figure 1. Then the code determined by T is the $[4, 1, 4]$ repetition code. This realization of the repetition code allows for both nontrivial computation tree pseudocodewords as well as connected realizations of graph cover pseudocodewords that are not vertices of the fundamental polytope. Figure 2 below shows one such nontrivial computation tree pseudocodeword for T .

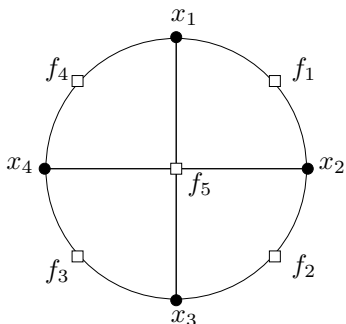


Fig. 1. The Tanner graph T_1 of Example 6.2.

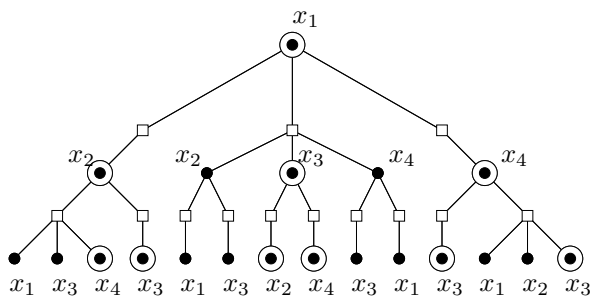


Fig. 2. A computation tree of depth 2 rooted at x_1 for the Tanner graph T in Figure 1. Labels on the check nodes are omitted for clarity. An inconsistent binary assignment \mathbf{c} is shown on the tree, where the ringed variable nodes are set to “1” and the others to “0”.

Table I gives values of $L_{\mathbf{c}}(x, f)$ for $i = 1, 2, 3, 4$ and $j = 1, 2, 3, 4, 5$. If the variable node x_i is not adjacent to the check node f_j , no value of $L_{\mathbf{c}}(x_i, f_j)$ is given. Since there is at least one column in this table that contains differing values, the configuration given in Figure 2 is not consistent.

Since the configuration in Figure 2 is not consistent, Kelley and Sridhara [8] point out that there is no mean-

	x_1	x_2	x_3	x_4
f_1	$\frac{1}{2}$	$\frac{1}{2}$	—	—
f_2	—	$\frac{2}{3}$	$\frac{2}{3}$	—
f_3	—	—	$\frac{2}{3}$	$\frac{2}{3}$
f_4	$\frac{1}{2}$	—	—	$\frac{1}{2}$
f_5	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{2}{3}$	$\frac{2}{3}$

TABLE I
VALUES OF $L_{\mathbf{c}}(x_i, f_j)$, WITH \mathbf{c} AS GIVEN IN FIGURE 2.

ingful vector of length four that we may associate to it and hence no graph cover pseudocodeword corresponds to it in this manner. In a different sense, however, the configuration in Figure 2 can be considered as being induced by a graph cover pseudocodeword. More specifically, the Tanner graph in Figure 3 is a 4-cover of the Tanner graph in Figure 1, so the configuration in Figure 3 is a realization of a graph cover pseudocodeword. By rooting a computation tree at the top-left variable node in Figure 3, one can derive the inconsistent computation tree configuration of Figure 2 from the configuration given in Figure 3. Thus, the computation tree pseudocodeword of Figure 2 is, in this sense, induced by a graph cover pseudocodeword. We see then that the criterion of consistency gives an incomplete characterization of the distinction between computation tree pseudocodewords and graph cover pseudocodewords.

It is clear that in order to study the relationship between linear programming/graph cover decoding and iterative message-passing decoders, one must better understand the relationship between graph covers and computation trees, and thus the notion of consistency or other characterizations of the distinctions between computation tree and graph cover pseudocodewords must be further explored.

VII. CONCLUSION

This paper has examined relationships between the three main types of pseudocodewords in the literature: linear programming pseudocodewords, graph cover pseudocodewords, and computation tree pseudocodewords. We have established the fundamental importance of these pseudocodewords by showing that LP pseudocodewords are the root cause of the discrepancy between LP and ML decoding. We further explored the role of particular subsets of graph cover pseudocodewords, namely minimal and irreducible pseudocodewords, and elucidated a discrepancy in the literature that resulted from the conflation of these terms. Additionally, we

