

Analysis of Connections Between Pseudocodewords

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

Abstract—The role of pseudocodewords in causing non-codeword outputs in linear programming 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 the minimal degree cover necessary to realize a 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—Graph cover decoding, iterative message-passing decoding, linear programming decoding, low-density parity-check (LDPC) codes, pseudocodewords.

I. INTRODUCTION

THE discovery of turbo codes [1] and the subsequent rediscovery of low-density parity-check (LDPC) codes [2], [3] 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 [4]. Perhaps the most important commonality between turbo and LDPC 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 non-codeword outputs that occur in computer simulations of LDPC codes with iterative message-passing algorithms.

Motivated by empirical observations of the non-codeword 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.* [6].

Manuscript received April 01, 2008; revised December 11, 2008. Current version published August 19, 2009.

N. Axvig, D. Dreher, K. Morrison, and J. L. Walker are with the Department of Mathematics, University of Nebraska, Lincoln, NE 68588-0130 USA (e-mail: s-naxvig1@math.unl.edu; s-dturk1@math.unl.edu; s-kmorri11@math.unl.edu; 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 (e-mail: epsota24@huskers.unl.edu; lperez@unl.edu).

Communicated by T. Etzion, Associate Editor for Coding Theory.

Digital Object Identifier 10.1109/TIT.2009.2025529

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 [7] 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 non-codeword outputs of iterative decoders on the BSC and AWGN channels.

In his dissertation [8], 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* [9], [10], 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 [11]. 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 non-codeword 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 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, with a focus on computation tree, graph cover, and LP pseudocodewords and characterizations of each. Section III 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 IV then focuses on the relationship between graph cover pseudocodewords and computation tree pseudocodewords, with an eye towards compactly describing the set of computation tree pseudocodewords. In particular, Section IV-A turns toward finding graphical realizations of graph cover pseudocodewords that may induce computation tree pseudocodewords. And in

Section IV-B, we explore the converse of this by examining which computation tree pseudocodewords are induced by realizations of graph cover pseudocodewords.

II. BACKGROUND AND MATHEMATICAL PRELIMINARIES

The success of LDPC 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 2.1: 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 $r := |F|$.

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; in this paper, we will assume the codeword is transmitted using binary antipodal modulation across the AWGN channel. The process of computing the estimate $\hat{\mathbf{c}}$ is called *decoding*, which can result in any of 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*.

In this paper, we will focus on decoding failures and the resulting *non-codeword outputs* of the decoder.

Wiberg [8] shows that the non-codeword outputs of iterative message-passing decoders such as sum-product (SP) and min-sum (MS) are caused by configurations 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 2.2 ([8]): 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.

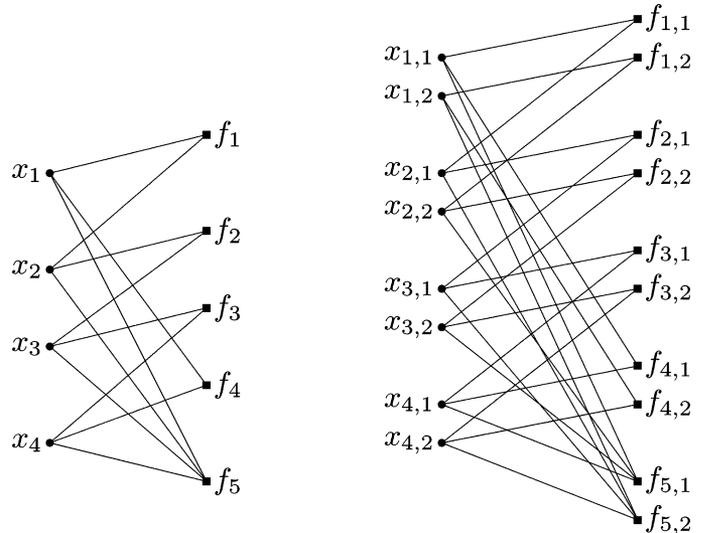


Fig. 1. A Tanner graph and a corresponding 2-cover.

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 MS and SP decoding, the decoder considers as competitors all valid configurations on $n := |X(T)|$ computation trees [8] 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 2.3 ([8]): 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; an example of a Tanner graph and one of its 2-covers is given in Fig. 1.

Definition 2.4: 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 the 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 .

We say that a graph 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 . In the remainder of this paper, we will consider only connected graphs.

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 2.5: 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 . Two kinds of *graph cover pseudocodewords* are associated to $\tilde{\mathbf{c}}$: 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

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

Since the Tanner graph is a 1-cover of itself, every codeword is both an unscaled and a normalized graph cover pseudocodeword. A *nontrivial* graph cover pseudocodeword is a graph cover pseudocodeword that is not a codeword in C .

Intuitively, all codewords on all covers of the Tanner graph are competitors in iterative message-passing decoding algorithms. In this vein, Vontobel and Koetter [11] 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 *LP decoding*, as defined by Feldman [12]. We now turn to the formal definition of LP decoding.

Definition 2.6 ([12]): 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 $[0, 1]^n \subset \mathbf{R}^n$ 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 $\mathbf{x} = (x_1, \dots, x_n) \in \mathbf{R}^n$, the *cost* $\gamma(\mathbf{x})$ of \mathbf{x} is given by

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

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 [12] 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 2.7: 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.

Feldman shows in [12] that linear programming decoding has the *ML-certificate property*, which guarantees that if LP decoding returns a codeword, this codeword is an ML codeword. Additionally, he shows that when LP decoding and ML decoding disagree, the output of LP decoding is an LP pseudocodeword, indicating that LP pseudocodewords are responsible for the difference between these two decoders. It is not immediately clear from this, however, if LP and ML decoding disagree in the presence of LP pseudocodewords simply because ML decoding is restricted to only outputting codewords while LP decoding is able to output pseudocodewords as well, or if they disagree because their decision rules are fundamentally different. It turns out that the latter explanation is true, as we will now explain.

Recall that on the AWGN channel, ML decoding is based on the squared Euclidean distance between modulated points. To determine whether the fundamental difference between LP and ML decoding is merely the collection of competitors each considers, we introduce a new decoding rule: *generalized maximum-likelihood (GML)* decoding. This rule minimizes the squared Euclidean distance between the received vector and any modulated vertex of the fundamental polytope. It is not difficult to show [13, Theorem 2.2] that whenever there exist nontrivial LP pseudocodewords, not only does LP decoding disagree with ML decoding, but it also disagrees with GML decoding as a result of fundamentally different decision rules. Thus, we see that LP pseudocodewords are essential to understanding the nonoptimality of LP decoding, and because of their ties to graph cover pseudocodewords they may aid in understanding the nonoptimality of iterative message-passing decoding algorithms as well.

Vontobel and Koetter demonstrate the relationship between LP and graph cover pseudocodewords by proving that the collection of rational points in the fundamental polytope is precisely the collection of graph cover pseudocodewords [11, Proposition 10]. Thus, with the definitions here, every linear programming pseudocodeword is a normalized graph cover pseudocodeword, but not *vice versa*. An additional characterization of the set of graph cover pseudocodewords is given by

Koetter, Li, Vontobel, and Walker [14]: 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

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

The connection between these two characterizations of graph cover pseudocodewords is clear, given the observation that the fundamental cone is the conic hull of the fundamental polytope [11].

III. 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 nonzero codeword. In particular, we examine different extensions of the theory of minimal codewords to graph cover (and hence LP) pseudocodewords.

Definition 3.1 ([11]): 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 [15].

Definition 3.2 ([15]): 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 3.3: 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 unscaled pseudocodewords then each of those pseudocodewords must consist of only zeros and ones and thus must be codewords themselves [14], 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 3.4: 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

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

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 that is 0 and one coordinate that is 1, and hence is not a pseudocodeword since it will not reduce modulo 2 to a codeword [14]. 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 [15], 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 [15] 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 3.5: A normalized graph cover pseudocodeword $\boldsymbol{\omega}$ 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) $\boldsymbol{\omega} = \boldsymbol{\omega}(\tilde{\mathbf{c}})$, and
- 2) whenever $\boldsymbol{\omega}$ 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 in Proposition 3.7 below, under the assumption we are given the coordinates of the graph cover pseudocodeword as a point in Feldman's *extended polytope* [12], rather than simply in the fundamental polytope. We first recall the definition of the extended polytope.

Definition 3.6 ([12]): 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 2.6 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

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

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 3.5 above.

Proposition 3.7: Let $\bar{\boldsymbol{\omega}}$ be a rational point in the extended polytope and let M be the degree of a minimal realization of $\bar{\boldsymbol{\omega}}$. Then M is the smallest positive integer such that each coordinate of $M\bar{\boldsymbol{\omega}}$ is a nonnegative integer.

Proof: For any positive integer t such that $t\bar{\boldsymbol{\omega}}$ is a vector of integers, Feldman [12] gives a construction that yields a realization of $t\bar{\boldsymbol{\omega}}$. 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, $\bar{\boldsymbol{\omega}}$ is realizable on an M -cover so $M\bar{\boldsymbol{\omega}}$ must be a vector of integers. Furthermore, M must be the smallest number such that $M\bar{\boldsymbol{\omega}}$ is a vector of integers, since if this held for some $t < M$ then, by the argument above, $\bar{\boldsymbol{\omega}}$ would be realizable on a t -cover, contradicting the minimality of M . \square

Proposition 3.7 can be extended to describe the minimum degree realization of any vector $\boldsymbol{\omega}$ with rational entries in the fundamental polytope whenever we can construct a corresponding $\bar{\boldsymbol{\omega}}$ in the extended polytope. Feldman [12] establishes that such an $\bar{\boldsymbol{\omega}}$ always exists, and Vontobel and Koetter [11] give a method for constructing $\bar{\boldsymbol{\omega}}$ under particular circumstances, namely, when it is already known how to express $\boldsymbol{\omega}$ as a convex linear combination of vectors in \mathbb{F}_2^n that satisfy a given check node.

IV. CONNECTIONS BETWEEN GRAPH COVER AND COMPUTATION TREE PSEUDOCODEWORDS

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 [8]. He establishes that iterative message-passing algorithms actually work by finding minimal cost configurations on computation trees, and 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. However, further analysis of computation tree pseudocodewords has proven difficult, largely because one generally cannot enumerate or easily describe the set of these pseudocodewords. On the other hand, this has been one of the major advantages of graph cover pseudocodewords: as mentioned above, multiple compact characterizations of graph cover pseudocodewords have been found, such as the fundamental cone and fundamental polytope. This has led us to investigate the relationship between graph cover pseudocodewords and computation tree pseudocodewords with the goal of using the various characterizations of graph cover pseudocodewords to yield a compact description of the set of computation tree pseudocodewords. In Section IV-A, we investigate which graph cover pseudocodewords can give rise to computation tree pseudocodewords. In particular, since computation trees are necessarily connected, we focus on the existence of connected realizations of graph cover pseudocodewords. In Section IV-B, we explore which computation tree pseudocodewords are induced by these connected realizations of graph cover pseudocodewords.

A. Connected Realizations of Pseudocodewords

In this subsection, we provide sets of conditions under which graph cover pseudocodewords are minimally realizable on connected covers. 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 4.1: 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

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

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 $M \geq 1$, and 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 4.1 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 4.1, these vertices do have connected realizations. This phenomenon happens in general, as shown in the next proposition which originally appeared in the conference paper [16].

Proposition 4.2 ([16], 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{e}) 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 $\tilde{e} = (\tilde{e}_1 | \dots | \tilde{e}_k)$, where \tilde{e}_i is a configuration on \tilde{T}_i . Then $(\tilde{T}_i, \tilde{e}_i)$ is a connected realization of ω for $i = 1, \dots, k$. In other words, every graph cover realization of a vertex of the fundamental polytope is either connected or the disjoint union of connected graph cover realizations of the vertex. In particular, every minimal realization of every vertex of the fundamental polytope is connected.

Proof: Set $\alpha_i = \omega(\tilde{e}_i)$ 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 a normalized graph cover pseudocodeword and hence is in \mathcal{P} [11, Proposition 10], 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 . \square

Theorem 4.4 below gives another sufficient condition for connected realizations of graph cover pseudocodewords to exist. The next lemma will be used in the proof.

Lemma 4.3: 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 [17, Theorem 2.4.3], $\pi^{-1}(pe_i)$ consists of a disjoint union of cycles that 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 $\tilde{T} - \{\tilde{e}_1, \dots, \tilde{e}_t\}$. \square

Theorem 4.4: 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 [11] and so is minimally realizable on an M -cover for some M . Let (\tilde{T}, \tilde{e}) 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

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

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

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

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 $na_v = ra_c$, and 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 (IV.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, $a_c \geq 3$, and 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 $\tilde{e}_R = (\tilde{f}_R, \tilde{x}_R)$ and $\tilde{e}'_R = (\tilde{f}_R, \tilde{x}'_R)$ be fixed lifts of e and e' , respectively, to \tilde{R} and let $\tilde{e}_S = (\tilde{f}_S, \tilde{x}_S)$ and $\tilde{e}'_S = (\tilde{f}_S, \tilde{x}'_S)$ be fixed lifts of e and e' , respectively, to \tilde{S} . By Lemma 4.3, $\tilde{R} - \{\tilde{e}_R, \tilde{e}'_R\}$ and $\tilde{S} - \{\tilde{e}_S, \tilde{e}'_S\}$ are connected.

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

Finally, assume $\tilde{c}(\tilde{x}_R) \neq \tilde{c}(\tilde{x}_S)$ and $\tilde{c}(\tilde{x}'_R) \neq \tilde{c}(\tilde{x}'_S)$. Then $(\tilde{c}(\tilde{x}_R) + \tilde{c}(\tilde{x}'_S)) \equiv (\tilde{c}(\tilde{x}_S) + \tilde{c}(\tilde{x}'_R)) \pmod{2}$. 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 \tilde{f}_R or at \tilde{f}_S , and again results in \tilde{R} and \tilde{S} becoming a single connected component, i.e., $(\tilde{T} - \{\tilde{e}_R, \tilde{e}'_R, \tilde{e}_S, \tilde{e}'_S\} + \{\tilde{f}_R\tilde{x}_S, \tilde{f}_R\tilde{x}'_S, \tilde{f}_S\tilde{x}_R, \tilde{f}_S\tilde{x}'_R\}, \tilde{c})$ is a minimal realization of ω on a cover with fewer connected components than \tilde{T} , a contradiction.

Since, in each 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 . \square

Remark 4.5: It should be noted that while the conditions given in Theorem 4.4 may sound rather restrictive, most practical codes actually satisfy these conditions. Tanner graphs with many check nodes of degree two or less are not generally practical since these check nodes unnecessarily decrease the rate of the code. Furthermore, closer examination of the proof of Theorem 4.4 reveals that the restrictions given in the statement of that theorem can even be loosened to include any graphs whose average variable node degree a_v and average check node degree a_c satisfy

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

Thus, the theorem applies to the majority of practical codes as well as a significant portion of cycle codes.

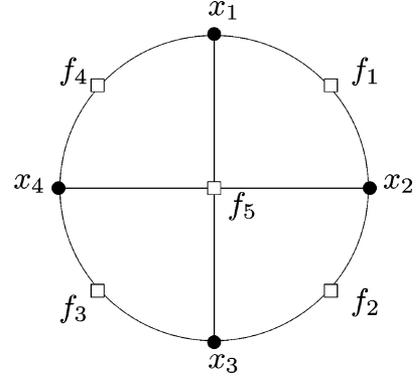


Fig. 2. The Tanner graph T of Example 4.7.

B. Computation Tree Pseudocodewords Induced by Graph Cover Pseudocodewords

As outlined above, 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 MS decoding versus LP/graph cover decoding observed across simulations [16]. Kelley and Sridhara [15] 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 of the work of Kelley and Sridhara [15] in this direction, starting with the notion of a *consistent* configuration on a computation tree for the Tanner graph T .

Definition 4.6 ([15]): 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} [15]. 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 . Since the fundamental polytope is the intersection of all vectors satisfying all of the local check constraints for every check node f , we see that ω is an element of the fundamental polytope, and hence realizable on a finite cover of T [15]. An example of a valid configuration on a computation tree that is not consistent is shown in Example 4.7.

Example 4.7 (See Also [15]): Let T be the Tanner graph of Fig. 2. Then the code determined by T is the $[4, 1, 4]$ repetition code. This realization of the repetition code allows for both non-trivial computation tree pseudocodewords as well as connected

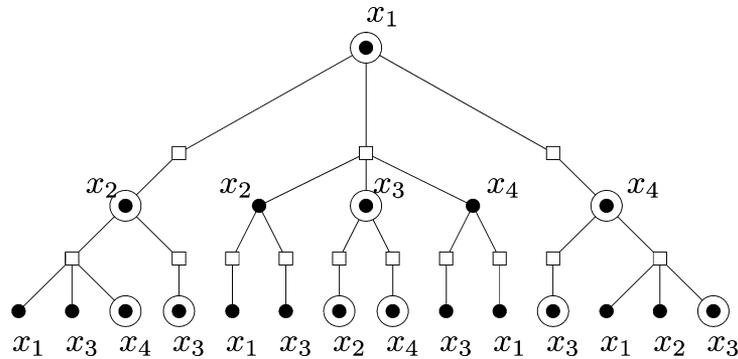


Fig. 3. A computation tree of depth 2 rooted at x_1 for the Tanner graph T in Fig. 2. Labels on the check nodes are omitted for clarity. An inconsistent binary assignment \mathbf{c} is shown on the tree, where the circled variable nodes are set to “1” and the others to “0.”

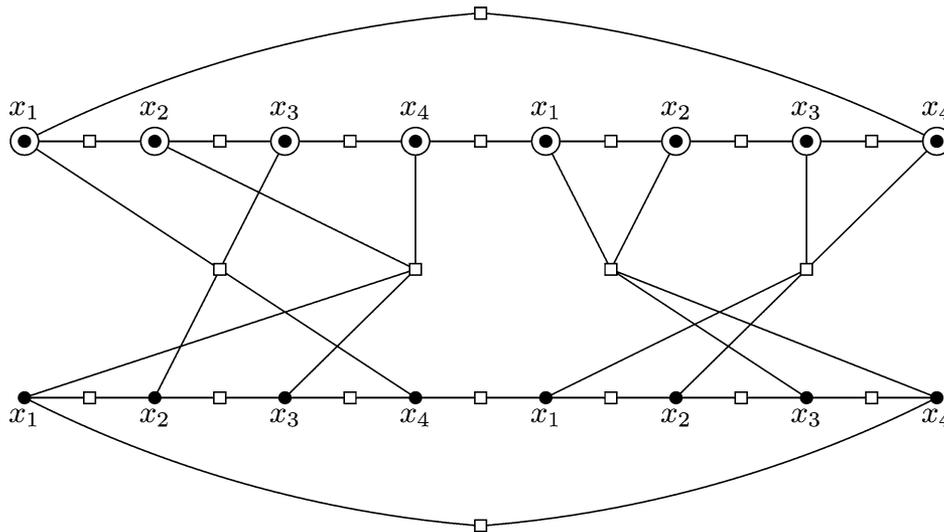


Fig. 4. A configuration on a 4-cover of the Tanner graph T given in Fig. 2. Circled nodes have a binary value of “1,” and other nodes have a binary value of “0.”

TABLE I
VALUES OF $L_{\mathbf{c}}(x_i, f_j)$, WITH \mathbf{c} AS GIVEN IN FIG. 3

	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	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{2}{3}$	$\frac{2}{3}$

realizations of graph cover pseudocodewords that are not vertices of the fundamental polytope. Fig. 3 shows one such non-trivial computation tree pseudocodeword for T .

Table I gives values of $L_{\mathbf{c}}(x_i, f_j)$ 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 Fig. 3 is not consistent.

Since the configuration in Fig. 3 is not consistent, Kelley and Sridhara [15] point out that there is no meaningful 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 Fig. 3 can be considered as being induced by a graph cover pseudocodeword. More specifically, the Tanner graph in Fig. 4 is a 4-cover of the Tanner graph in Fig. 2, and so the configuration in Fig. 4 is a realization of a graph cover pseudocodeword. By rooting a computation tree at the top-left variable node in Fig. 4, one can derive the inconsistent computation tree configuration of Fig. 3 from the configuration given in Fig. 4. Thus, the computation tree pseudocodeword of Fig. 3 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. Furthermore, we conjecture that every computation tree pseudocodeword is induced by a graph cover pseudocodeword, thus strengthening the notion that characterizations of graph cover pseudocodewords may be utilized to compactly describe the set of computation tree 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.

V. 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 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 presented results describing conditions under which connected realizations of graph cover pseudocodewords exist, and we examined when graph cover pseudocodewords correspond to computation tree pseudocodewords and *vice versa*. A number of open questions still remain as highlighted by this analysis. Further investigation is necessary to determine the roles of minimal and irreducible pseudocodewords, keeping in mind that these notions are distinct. The relationship between graph cover pseudocodewords and computation tree pseudocodewords must also be further examined in order to reach our final goal of fully understanding the non-codeword decoder errors of iterative message-passing decoders.

REFERENCES

- [1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding," in *Proc. 1993 IEEE Int. Conf. Communications*, Geneva, Switzerland, 1993, pp. 1064–1070.
 - [2] R. G. Gallager, *Low-Density Parity Check Codes*. Cambridge, MA: MIT Press, 1963.
 - [3] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low-density parity check codes," *IEE Electron. Lett.*, vol. 32, no. 18, pp. 1645–1646, Aug. 1996.
 - [4] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, pp. 379–423, 623–656, Jul., Oct. 1948.
 - [5] G. D. Forney, Jr., R. Koetter, F. R. Kschischang, and A. Reznik, "On the effective weights of pseudocodewords for codes defined on graphs with cycles," in *Codes, Systems, and Graphical Models*, ser. IMA Vol. Math. Appl. New York/Minneapolis, NY/MN: Springer, 1999, 2001, vol. 123, pp. 101–112.
 - [6] D. Changyan, D. Proetti, I. E. Telatar, T. J. Richardson, and R. L. Urbanke, "Finite length analysis of low-density parity-check codes on the binary erasure channel," *IEEE Trans. Inf. Theory*, vol. 48, no. 6, pp. 1570–1579, Jun. 2002.
 - [7] C. Kelley, D. Sridhara, J. Xu, and J. Rosenthal, "Pseudocodeword weights and stopping sets," in *Proc. 2004 IEEE Int. Symp. Information Theory*, Chicago, IL, Jun./Jul. 2004, p. 150.
 - [8] N. Wiberg, "Codes and Decoding on General Graphs," Ph.D. dissertation, Linköping Univ., Linköping, Sweden, 1996.
 - [9] T. Richardson, A. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity check codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 619–637, Feb. 2001.
 - [10] T. Richardson and R. Urbanke, "The capacity of low-density parity check codes under message-passing decoding," *IEEE Trans. Inf. Theory*, 47, no. 2, pp. 599–618, Feb. 2001.
 - [11] P. Vontobel and R. Koetter, "Graph-cover decoding and finite-length analysis of message-passing iterative decoding of LDPC codes," *IEEE Trans. Inf. Theory*, to be published.
 - [12] J. Feldman, "Decoding Error-Correcting Codes via Linear Programming," Ph.D. dissertation, MIT, Cambridge, MA, 2003.
 - [13] N. Axvig, K. Morrison, E. Psota, D. Dreher, L. C. Pérez, and J. L. Walker, "Towards a Universal Theory of Decoding and Pseudocodewords University of Nebraska-Lincoln, SGER Tech. Rep. 0801, Mar. 2008 [Online]. Available: http://www.math.unl.edu/~jwalker7/papers/SGER_tech_report.pdf
 - [14] R. Koetter, W.-C. W. Li, P. O. Vontobel, and J. L. Walker, "Characterizations of pseudo-codewords of LDPC codes," *Adv. Math.*, vol. 213, pp. 205–229, 2007.
 - [15] C. Kelley and D. Sridhara, "Pseudocodewords of Tanner graphs," *IEEE Trans. Inf. Theory*, vol. 53, no. 11, pp. 4013–4038, Nov. 2007.
 - [16] N. Axvig, E. Price, E. Psota, D. Turk, L. Pérez, and J. Walker, "A universal theory of pseudocodewords," in *Proc. 45th Ann. Allerton Conf. Communication, Control, and Computing*, Monticello, IL, Sept. 2007, pp. 336–343.
 - [17] J. L. Gross and T. W. Tucker, *Topological Graph Theory*. Mineola, NY: Dover, 2001, reprint of the 1987 original [Wiley, New York; MR0898434 (88h:05034)] with a new preface and supplementary bibliography.
- Nathan Axvig** received the B.S. degree in mathematics from the University of North Dakota, Grand Forks, ND, in 2005 and the M.S. degree in mathematics from the University of Nebraska, Lincoln, NE, in 2007. He is currently working towards the Ph.D. degree in mathematics at the University of Nebraska, Lincoln. His research interests lie in error control coding and graph theory.
- Deanna Dreher** received the B.S. degree in mathematics from the University of Northern Colorado, Greeley, CO, in 2004. She received the M.S. degree in mathematics from the University of Nebraska, Lincoln, NE, in 2005. She participated in the Graduate Mathematics Program at the National Security Agency in 2007 and is currently working towards the Ph.D. degree in mathematics with a minor in electrical engineering at the University of Nebraska, Lincoln. Her research interests are in the area of error control coding and graph theory, particularly in the study of graph-based codes and iterative message-passing decoding schemes.
- Katherine Morrison** received the B.A. degrees in mathematics and psychology from Swarthmore College, Swarthmore, PA, in 2005. She received the M.S. degree in mathematics from the University of Nebraska, Lincoln, NE, in 2008. She is currently working towards the Ph.D. degree in mathematics with a minor in electrical engineering at the University of Nebraska, Lincoln. Her research interests include error control coding, network coding, and graph theory.
- Eric Psota (S'05)** received the B.S. and M.S. degrees in electrical engineering from the University of Nebraska, Lincoln, NE, in 2004 and 2006, respectively. He is currently working towards the Ph.D. degree in electrical engineering from the University of Nebraska, Lincoln. His research interests include digital communication systems, signal processing, error control coding, network coding, image processing, multiview geometric imaging systems, and computer vision.
- Lance C. Pérez (M'87–SM'01)** received the B.S. degree in electrical engineering from the University of Virginia, Charlottesville, VA, in 1987 and the M.S. and Ph.D. degrees in electrical engineering from the University of Notre Dame, Notre Dame, IN, in 1989 and 1995, respectively. Since 1996, he has been with the Department of Electrical Engineering at the University of Nebraska, Lincoln, NE, where he is currently an Associate Professor. He is the coauthor, with Christian B. Schlegel, of the book *Trellis and Turbo Coding* published by IEEE Press/Wiley. His research interests are in error control coding, communication and information theory, and applications of wireless communications to assistive technology. Dr. Pérez is a past recipient of a National Science Foundation CAREER Award. He is a member of the IEEE Information Theory, Communication, and Education societies. He is also a member of Eta Kappa Nu and Tau Beta Pi.
- Judy L. Walker (M'96)** received the B.S. degree from the University of Michigan, Ann Arbor, MI, in 1990, and the M.S. and the Ph.D. degrees from the University of Illinois at Urbana-Champaign, Urbana, IL, in 1992 and 1996, all in mathematics. She has been with the University of Nebraska, Lincoln, NE, since 1996 and currently serves as Professor and Graduate Chair of the Department of Mathematics. Her research interests are in algebraic coding theory, including algebraic aspects of codes on graphs and algebraic geometry codes. Dr. Walker is a member of the American Mathematical Society (AMS), the Association for Women in Mathematics (AWM), and the Mathematical Association of America (MAA). She has served as an elected member of both the AMS Council and the AWM Executive Committee. She was awarded the Deborah and Franklin Tepper Haimo Award for Distinguished Teaching by the MAA in 2006 and will serve as the MAA's Polya Lecturer for 2009–2011.