10-2007

A Universal Theory of Pseudocodewords

Nathan Axvig
University of Nebraska - Lincoln, s-naxvig1@math.unl.edu

Emily Price
University of Nebraska - Lincoln

Eric T. Psota
University of Nebraska-Lincoln, epsota@unl.edu

Deanna Turk
University of Nebraska - Lincoln

Lance C. Pérez
University of Nebraska-Lincoln, lperez@unl.edu

See next page for additional authors

Axvig, Nathan; Price, Emily; Psota, Eric T.; Turk, Deanna; Pérez, Lance C.; and Walker, Judy L., "A Universal Theory of Pseudocodewords" (2007). *Faculty Publications, Department of Mathematics*. 176.
https://digitalcommons.unl.edu/mathfacpub/176

Published in Proceedings of the 45th Annual Allerton Conference on Communication, Control and Computing, Monticello, IL; October 2007.

A Universal Theory of Pseudocodewords

Nathan Axvig, Emily Price, Eric Psota, Deanna Turk, Lance C. Pérez, and Judy L. Walker

Abstract— **Three types of pseudocodewords for LDPC codes are found in the literature: graph cover pseudocodewords, linear programming pseudocodewords, and computation tree pseudocodewords. In this paper we first review these three notions and known connections between them. We then propose a new decoding rule — universal cover decoding — for LDPC codes. This new decoding rule also has a notion of pseudocodeword attached, and this fourth notion provides a framework in which we can better understand the other three.**

I. Introduction

The discovery of turbo codes [2] and the subsequent rediscovery of low-density parity-check (LDPC) codes [4], [8] represent a major milestone in the field of coding theory. These two classes of codes can achieve realistic bit error rates, between 10^{-5} and 10^{-12}, 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 [11].

Perhaps the most important commonality between turbo and LDPC codes is that they both utilize iterative message-passing decoding algorithms. The focus of this paper is the determination of the behavior of iterative message-passing decoding and the relationships between the various decoding algorithms, with a particular aim toward an understanding of the noncodeword decoder errors that occur in computer simulations of LDPC codes with iterative message-passing decoders.

In the remainder of this section, we give some necessary background and terminology from graph theory. In Section II, we discuss the intuitively appealing view of iterative message-passing algorithms as acting locally on the Tanner graph, a view which leads to *graph cover pseudocodewords*. Section III considers linear programming decoding, which was introduced by Feldman [3], and its relationship to the ideas of Section II. In Section IV, we give a simulation result which provides a contradiction to the graph cover intuition. This leads us to return to the foundational work of Wiberg [13] in Section V. Finally, in Section VI we propose a decoding algorithm which provides insight as to how the newer work of Sections II and III fits in with Wiberg's theory.

Definition 1.1: 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 simple. In this case, we can uniquely identify any edge e with its endpoints, and we write $e = (u, v)$.

Definition 1.2: Let $G = (V, E)$ be a simple graph. For $v \in V$, the *neighborhood* of v is the set of vertices $u \in V$ such that $(u, v) \in E$. Elements of the neighborhood of 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 $v_0, \ldots, v_k$ of G such that v_{i-1} and v_i are neighbors for $1 \le i \le k$. A *cycle* in G is a path $v_0, \ldots, v_k$ 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 $u = v_0, v_1, \ldots, v_k = v$ 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 (c, d)*-regular* if the degree of every vertex in X is c and the degree of every vertex in F is d. We say G is a *tree* if G is connected and has no cycles.

Definition 1.3: 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 *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 determined by T is then 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 $r := |F|$.

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

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

Note that the code determined by $T(H)$ is precisely the code with parity check matrix H.

Conversely, if $T = (\{x_1, \ldots, x_n\} \cup \{f_1, \ldots, f_r\}, E)$ is a

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

N. Axvig, E. Price, D. Turk and J. L. Walker are with the Department of Mathematics, University of Nebraska, Lincoln, NE 68588-0130, USA {s-naxvig1,s-dturk1,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

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 code with parity check matrix $H(T)$ is precisely the code determined by T.

Since $T = T(H(T))$ for any Tanner graph T and $H = H(T(H))$ for any binary matrix H, we have, for any binary linear code C, a one-to-one correspondence between parity check matrices and Tanner graphs for C.

The proof of the next proposition is clear.

Proposition 1.4: Suppose T is a Tanner graph which is not connected, say $T_1, \ldots, T_k$ are the connected components of T. Let $C \subseteq \mathbb{F}_2^n$ be the code determined by T and $C_i \subseteq \mathbb{F}_2^{n_i}$ the code determined by T_i for $i = 1, \ldots, k$, where $n_1 + \cdots + n_k = n$. Then C is the direct sum of the C_i, i.e.,

$$\begin{aligned} C &= C_1 \oplus \cdots \oplus C_k \\ &:= \{(\mathbf{c}_1 | \ldots | \mathbf{c}_k) \in \mathbb{F}_2^n \,|\, \mathbf{c}_1 \in C_1, \ldots, \mathbf{c}_k \in C_k\}. \end{aligned}$$

In light of Proposition 1.4, we will assume for the remainder of the paper that the Tanner graph T is connected.

The power of LDPC codes lies in the existence of iterative message-passing decoding algorithms which act on the associated Tanner graph. Two such algorithms are the min-sum and the sum-product algorithms, discussed by Wiberg in [13]. Loosely speaking, an iterative message-passing algorithm is a method of decoding in which each variable node and each check node of T is initialized with some data set provided by the output of the channel. The variable and check nodes take turns passing data to their neighbors in T, performing some calculation at each step. This process of 'message passing' is allowed to continue for some predetermined number of iterations, with the expectation being that if the number of iterations is sufficiently large, then the messages will converge.

II. Graph Cover Pseudocodewords

In any iterative message-passing algorithm, the computation at each particular vertex uses only information from its immediate neighbors. This local nature of the algorithm on the Tanner graph prompts us to consider *covers* of graphs.

Definition 2.1: An *unramified cover*, or simply a *cover*, of a finite graph G is a graph $\widetilde{G}$ along with a surjective graph homomorphism $\pi : \widetilde{G} \to G$, 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 $\pi : \widetilde{G} \to G$ such that for each vertex v of G, $\pi^{-1}(v)$ contains exactly M vertices of $\widetilde{G}$.

Notice that, with this definition, a cover of a connected graph need not be connected.

Example 2.2: If G is an r-cycle, then for $M \geq 1$ the only connected cover of G is the rM-cycle. Other graphs, however, admit several connected covers of each degree. For example, the graph on the left in Figure 1 has two connected 2-covers, as shown on the right of that figure.

Let $T = (X \cup F, E)$ be a Tanner graph for the code $C \subseteq \mathbb{F}_2^n$ and let $\pi : \widetilde{T} \to T$ be an M-cover of T. Then $\widetilde{T}$ is a Tanner graph, with variable nodes in the set

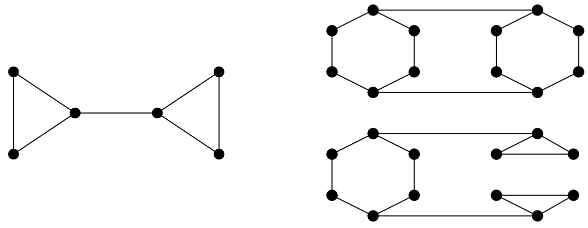

Fig. 1. The graph of Example 2.2, along with its two connected 2-covers.

$$\widetilde{X} := \bigcup_{x \in X} \pi^{-1}(x)$$

and check nodes in the set

$$\widetilde{F} := \bigcup_{f \in F} \pi^{-1}(f).$$

Writing $X = \{x_1, \ldots, x_n\}$, we can write

$$\widetilde{X} = \{x_{ik} \,|\, 1 \leq i \leq n, 1 \leq k \leq M\}$$

with $\pi(x_{ik}) = x_i$. We use this indexing to write a vector $\widetilde{\mathbf{a}} \in \mathbb{F}_2^{nM}$ as

$$\widetilde{\mathbf{a}} = (a_{11} : \cdots : a_{1M}, \ldots, a_{n1} : \cdots : a_{nM}).$$

Definition 2.3: For any vector $\mathbf{a} = (a_1, \ldots, a_n) \in \mathbb{F}_2^n$, the *$M$-lift* $\mathbf{a}^{\uparrow M} \in \mathbb{F}_2^{nM}$ is the vector

$$\mathbf{a}^{\uparrow M} = (a_{11} : \cdots : a_{1M}, \ldots, a_{n1} : \cdots : a_{nM})$$

with $a_{ik} = a_i$ for $1 \leq i \leq n$, $1 \leq k \leq M$.

Let $\widetilde{C} \subseteq \mathbb{F}_2^{nM}$ be the code determined by $\widetilde{T}$. It is easy to see that if $\widetilde{\mathbf{a}} \in \mathbb{F}_2^{nM}$ satisfies $\widetilde{\mathbf{a}} = \mathbf{a}^{\uparrow M}$ for some $\mathbf{a} \in \mathbb{F}_2^n$, then $\mathbf{a}^{\uparrow M} \in \widetilde{C}$ if and only if $\mathbf{a} \in C$. In particular, $C^{\uparrow M} := \{\mathbf{c}^{\uparrow M} \,|\, \mathbf{c} \in C\}$ is a subcode of $\widetilde{C}$. However, in many situations we have $C^{\uparrow M} \neq \widetilde{C}$, i.e., $\widetilde{C}$ contains codewords which are not constant on all preimages of variable nodes of T. This leads us to the next definition.

Definition 2.4: Let $C \subseteq \mathbb{F}_2^n$ be a binary linear code with Tanner graph T. Let $\widetilde{T}$ be an M-cover of T and let $\widetilde{C} \subseteq \mathbb{F}_2^{nM}$ be the code determined by $\widetilde{T}$. For any codeword

$$\widetilde{\mathbf{c}} = (c_{11} : \cdots : c_{1M}, \ldots, c_{n1} : \cdots : c_{nM}) \in \widetilde{C},$$

the *unscaled graph cover pseudocodeword* associated to $\widetilde{\mathbf{c}}$ is the vector

$$\mathbf{p}(\widetilde{\mathbf{c}}) := (p_1, \ldots, p_n)$$

of nonnegative integers, where $p_i = \#\{k \,|\, c_{ik} = 1\}$. The *normalized graph cover pseudocodeword* associated to $\widetilde{\mathbf{c}}$ is the vector

$$\boldsymbol{\omega}(\widetilde{\mathbf{c}}) := \frac{1}{M}\mathbf{p}(\widetilde{\mathbf{c}})$$

of rational numbers between 0 and 1. If $\mathbf{p}$ is an unscaled graph cover pseudocodeword for T, then $(\widetilde{T}, \widetilde{\mathbf{c}})$ is a *realization* for $\mathbf{p}$ if $\widetilde{T}$ is a finite cover of T and $\widetilde{\mathbf{c}}$ is a configuration on $\widetilde{T}$ (i.e., a codeword in the code determined by $\widetilde{T}$) such that $\mathbf{p}(\widetilde{\mathbf{c}}) = \mathbf{p}$; a *realization* of a normalized graph cover pseudocodeword is defined similarly. A realization $(\widetilde{T}, \widetilde{\mathbf{c}})$

of a graph cover pseudocodeword is called a *connected realization* if $\widetilde{T}$ is connected.

If the appropriate adjective is clear from context, the term *pseudocodeword* will often be used to refer to either an unscaled graph cover pseudocodeword or a normalized graph cover pseudocodeword. We note that every codeword is both an unscaled graph cover pseudocodeword and a normalized graph cover pseudocodeword, since the Tanner graph T is a 1-cover of itself.

Example 2.5 (See also [6].): The Tanner graph T on top in Figure 2 determines the code C spanned by $\mathbf{a} := (1, 1, 1, 0, 0, 0, 0)$ and $\mathbf{b} := (0, 0, 0, 0, 1, 1, 1)$. The graph $\widetilde{T}$ on the bottom in Figure 2 is a 2-cover of T and determines the code $\widetilde{C}$ spanned by $\mathbf{a}^{\uparrow 2}$, $\mathbf{b}^{\uparrow 2}$ and $\widetilde{\mathbf{c}} := (1{:}0, 1{:}0, 1{:}0, 1{:}1, 1{:}0, 1{:}0, 1{:}0)$. The unscaled graph cover pseudocodeword corresponding to $\widetilde{\mathbf{c}}$ is $\mathbf{p}(\widetilde{\mathbf{c}}) = (1, 1, 1, 2, 1, 1, 1)$ and the normalized graph cover pseudocodeword corresponding to $\widetilde{\mathbf{c}}$ is $\boldsymbol{\omega}(\widetilde{\mathbf{c}}) = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, 1, \frac{1}{2}, \frac{1}{2}, \frac{1}{2})$.

The relevance of graph covers to iterative message-passing decoding is very intuitive: since every finite cover $\widetilde{T}$ of T is locally isomorphic to T, a local algorithm on T cannot distinguish between T and any finite cover of T. Thus, it seems reasonable that all codewords in all covers of the Tanner graph are considered by an iterative decoder. This intuition was formalized by Vontobel and Koetter [12] with their definition of *graph cover decoding*.

Definition 2.6: [12] Assume the code C with Tanner graph T is used on a binary-input, memoryless channel with channel law described by the conditional probability density function $P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^{n} P_{Y_i|X_i}(y_i|x_i)$. Let $\mathcal{M}$ be the collection of all triples $(M, \widetilde{T}, \widetilde{\mathbf{x}})$ where $\widetilde{T}$ is an M-cover of T and $\widetilde{\mathbf{x}}$ is a codeword in the code determined by $\widetilde{T}$. For a given channel output $\mathbf{y}$ and for any $\widetilde{\mathbf{x}}$ with $(M, \widetilde{T}, \widetilde{\mathbf{x}}) \in \mathcal{M}$ for some M and $\widetilde{T}$, set

$$P_{\widetilde{\mathbf{Y}}|\widetilde{\mathbf{X}}}(\mathbf{y}^{\uparrow M}\widetilde{\mathbf{x}}) := \prod_{i=1}^{n}\prod_{k=1}^{M} P_{Y_i|X_i}(y_i|x_{ik}),$$

where $\mathbf{y}^{\uparrow M}$ is the vector obtained by repeating each entry M times. *Graph cover decoding* is the decoding algorithm given by the following decision rule: For a received vector $\mathbf{y}$, find the triple $(M, \widetilde{T}, \widetilde{\mathbf{x}}) \in \mathcal{M}$ that maximizes the quantity $\frac{1}{M}\log P_{\widetilde{\mathbf{Y}}|\widetilde{\mathbf{X}}}(\mathbf{y}^{\uparrow M}|\widetilde{\mathbf{x}})$, and return $\boldsymbol{\omega}(\widetilde{\mathbf{x}})$.

In other words, graph cover decoding simultaneously lifts the received vector to all finite covers of the Tanner graph, compares these lifts to all the codewords in the corresponding codes, and returns the normalized graph cover pseudocodeword corresponding to the covering-code codeword which has the highest likelihood of having been sent.

In [6], Koetter, Li, Vontobel and Walker study characterizations of graph cover pseudocodewords. In particular, they show that 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 lives in the *fundamental cone* $\mathcal{K} \subseteq \mathbb{R}^n$ given by

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

In the case of *cycle codes*, they show that a vector $\mathbf{p} = (p_1, \ldots, p_n)$ of nonnegative integers is an unscaled graph cover pseudocodeword if and only if $\mathbf{u}^{\mathbf{p}} := u_1^{p_1} \cdots u_n^{p_n}$ appears with nonzero coefficient in the *edge zeta function* of the *normal graph* of the Tanner graph, and they give a generalization of this characterization to arbitrary LDPC codes.

III. Linear Programming Pseudocodewords

In this section, we discuss linear programming decoding and the notion of linear programming pseudocodewords. We also review the connections found by Vontobel and Koetter [12] between linear programming decoding and graph cover decoding.

Much of the setup is the same as in the previous section. In particular, we assume a binary-input, memoryless channel with channel law described by $P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^{n} P_{Y_i|X_i}(y_i|x_i)$. For a given output $\mathbf{y} = (y_1, \ldots, y_n)$ of the channel, the *log-likelihood vector* $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_n)$ is given by

$$\lambda_i = \log\left(\frac{P_{Y_i|X_i}(y_i|0)}{P_{Y_i|X_i}(y_i|1)}\right),$$

and, for $\mathbf{x} \in \mathbb{R}^n$, the *cost* of $\mathbf{x}$ is

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

Definition 3.1: [3] Let $H = (h_{ji})$ be a fixed $r \times n$ parity-check matrix with corresponding Tanner graph T. For each $j = 1, \ldots, r$, let $N(j)$ be the set of variable nodes which are adjacent to check node j in T, i.e.,

$$N(j) := \{i \,|\, h_{ji} = 1\}.$$

We call a vector $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_n) \in \mathbb{R}^n$ *good* if $0 \leq \omega_i \leq 1$ for each i, and

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

for each j and each subset $S \subseteq N(j)$ with $|S|$ odd. The *fundamental polytope* $\mathcal{P} = \mathcal{P}(H)$ of H is the set of all good vectors and *linear programming decoding* is the decision rule which returns a vector $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_n) \in \mathcal{P}$ which has minimal cost.

Because the output of linear programming decoding may always be taken to be a vertex of the fundamental polytope, we define a *linear programming pseudocodeword* to be any vertex of the fundamental polytope. Feldman [3] showed that every codeword is a linear programming pseudocodeword and that a vector of 0's and 1's is a linear programming pseudocodeword if and only if it is a codeword. However, as is the case with graph cover pseudocodewords, there are often linear programming pseudocodewords which are not codewords.

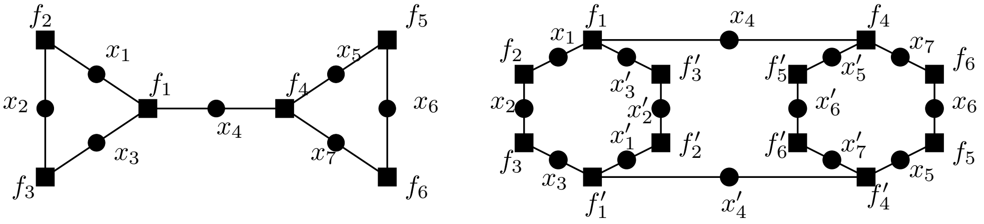

Fig. 2. The graphs of Example 2.5.

In [12], Vontobel and Koetter showed that linear programming decoding and graph cover decoding are essentially the same: for a given channel output, graph cover decoding and linear programming decoding return the same vector of rational numbers between 0 and 1. Moreover, the fundamental cone mentioned in Section II above is precisely the conic hull of Feldman's fundamental polytope $\mathcal{P}$, and a vector $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_n)$ of rational numbers between 0 and 1 is in $\mathcal{P}$ if and only if it an unscaled graph cover pseudocodeword.

Notice that disconnected covers are needed for this last statement to be true, as the next example shows.

Example 3.2: 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

$$\mathcal{P} = \mathcal{P}(H) = \left\{ (\omega, \omega, \omega, \omega) \in \mathbb{R}^4 \,|\, 0 \le \omega \le 1 \right\}.$$

As discussed in Example 2.2, the only connected covers of T are $8M$-cycles for $M \ge 1$. Hence the only unscaled graph cover pseudocodewords which have connected realizations are those of the form $(0, 0, 0, 0)$ and (M, M, M, M) for $M \ge 1$, and so 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 $\mathcal{P}$ which is not a vertex of $\mathcal{P}$ has a connected graph cover realization.

On the other hand, we know that linear programming decoding (and hence graph cover decoding) will always output a vertex of the fundamental polytope and, we observe that in Example 3.2, these vertices do have connected realizations. This phenomenon happens in general, as shown by the next proposition.

Proposition 3.3: Let T be a Tanner graph with corresponding fundamental polytope $\mathcal{P}$. Suppose $\boldsymbol{\omega}$ is a vertex of $\mathcal{P}$, and let $(\widetilde{T}, \widetilde{\mathbf{c}})$ be a realization of $\boldsymbol{\omega}$. Let $\widetilde{T}_1$, …, $\widetilde{T}_k$ be the connected components of $\widetilde{T}$, so that $\widetilde{T}_i$ an M_i-cover of T for $1 \le i \le k$, with $M_1 + \cdots + M_k = M$, and $\widetilde{\mathbf{c}} = (\widetilde{\mathbf{c}}_1 | \ldots | \widetilde{\mathbf{c}}_k)$, where $\widetilde{\mathbf{c}}_i$ is a configuration on $\widetilde{T}_i$. Then $(\widetilde{T}_i, \widetilde{\mathbf{c}}_i)$ is a connected realization of $\boldsymbol{\omega}$ for $i = 1, \ldots, k$. 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 $\boldsymbol{\alpha}_i = \boldsymbol{\omega}(\widetilde{\mathbf{c}}_i)$ for $1 \le i \le k$. Then, looking at the unscaled graph cover pseudocodewords, we have

$$M\boldsymbol{\omega} = M_1\boldsymbol{\alpha}_1 + \cdots + M_k\boldsymbol{\alpha}_k.$$

Dividing through by M gives

$$\boldsymbol{\omega} = \frac{M_1}{M}\boldsymbol{\alpha}_1 + \cdots + \frac{M_k}{M}\boldsymbol{\alpha}_k.$$

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

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

we have written $\boldsymbol{\omega}$ as a convex combination of $\boldsymbol{\alpha}_1$, …, $\boldsymbol{\alpha}_k$. But each $\boldsymbol{\alpha}_i$ is in $\mathcal{P}$ by [12] and so each $\frac{M_i}{M}\boldsymbol{\alpha}_i$ is too since $\frac{M_i}{M} \le 1$. Since $\boldsymbol{\omega}$ is a vertex of the polytope, this forces each $\boldsymbol{\alpha}_i$ to lie on the line segment from the origin to $\boldsymbol{\omega}$, i.e., $\boldsymbol{\alpha}_i = \gamma_i\boldsymbol{\omega}$ for some rational numbers $0 < \gamma_i \le 1$. So we have

$$M\boldsymbol{\omega} = (M_1\gamma_1 + \cdots + M_k\gamma_k)\boldsymbol{\omega},$$

which means $M_1 + \cdots + M_k = M = M_1\gamma_1 + \cdots + M_k\gamma_k$. Hence $\gamma_i = 1$ for each i, i.e., $\boldsymbol{\alpha}_i = \boldsymbol{\omega}$ for all i. ■

IV. A Simulation Result

As discussed above, intuition tells us that iterative message-passing decoders are approximations to graph cover decoding. As graph cover decoding requires a comparison involving all codewords in the codes corresponding to all finite covers of the Tanner graph, it must be viewed as a theoretical tool rather than as an implementable (or simulatable) algorithm. However, one can implement linear programming decoding and, since (as discussed above) linear programming decoding and graph cover decoding are equivalent [12], this yields a way of testing the intuition. In particular, if the intuition is correct, then graph cover decoding (i.e., linear programming decoding) should always out-perform iterative message-passing-decoding. However, simulations show that this is not the case, as is shown in the next example.

Example 4.1: Figure 3 shows the simulation results for a turbo-based LDPC code [7] with linear programming decoding, sum-product decoding and min-sum decoding. This figure clearly shows that the iterative message-passing decoders are superior to linear programming decoding (i.e., graph cover decoding) for this particular code, with respect to both word error rate and bit error rate.

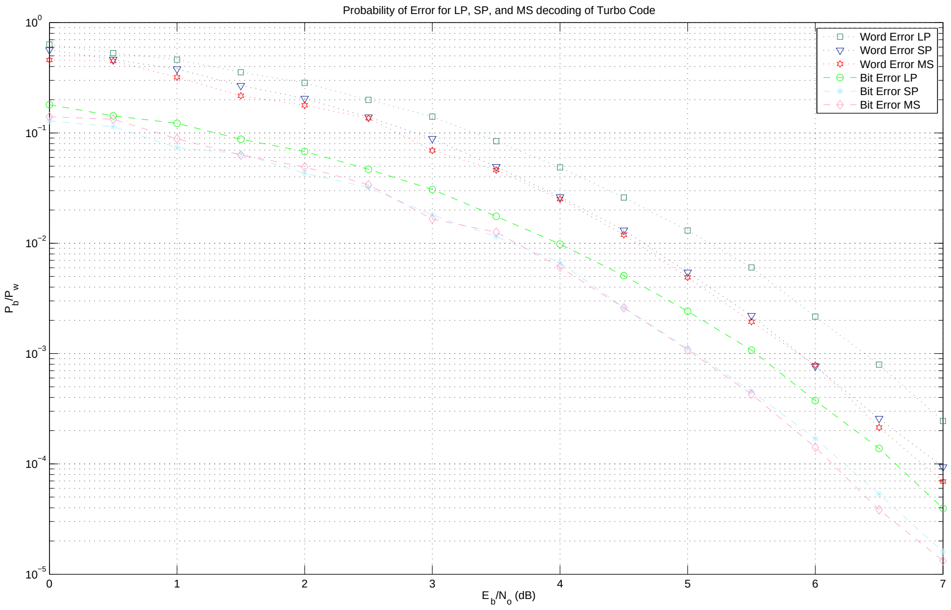

Fig. 3. Performance of a turbo code based LDPC code with linear programming (LP), sum-product (SP) and min-sum (MS) decoding.

V. Computation Tree Pseudocodewords

The conflict observed above — that iterative message-passing decoding does not appear to be an approximation of graph cover decoding — is resolved by returning to the fundamental work of Wiberg [13]. Recall that every iterative message-passing decoding algorithm works by recursively computing a cost function at each variable node and then making a decision based on those cost functions.

Definition 5.1: [13] 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 0^{th} 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, the computation tree locally looks like the original Tanner graph T: if (x, f) is an edge in T, then every copy of x 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.

Definition 5.2: Let R be a computation tree for the Tanner graph T. The *variable nodes* of R are the vertices of R which are copies of the variable nodes of T, and the *check nodes* of R are the vertices of R which are copies of the check nodes of T. A *configuration* on R is an assignment of 0's and 1's to the variable nodes of R in such a way that the binary sum of the neighbors of each check node in R is 0.

Wiberg [13] proves that iterative message-passing algorithms actually work by finding the minimal cost configuration on the computation tree. To make this more precise, we focus on the min-sum algorithm. In this case, we have:

Definition 5.3 (Wiberg, [13]): Let R be a computation tree for the Tanner graph T and let $X(R)$ be the variable nodes of R. Let $\mathbf{c} = (c_x)_{x \in X(R)}$ be a configuration on a computation tree. For each $x \in X(R)$ which is a copy of the i^{th} variable node of T, define the *local cost function* λ_x by

$$\lambda_x(\alpha) := \lambda_i \alpha,$$

where $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_n)$ is the log-likelihood vector and $\alpha \in \{0, 1\}$. The *(global) cost* of $\mathbf{c}$ is

$$G(\mathbf{c}) := \sum_{x \in X(R)} \lambda_x(c_x).$$

Theorem 5.4 (Wiberg, [13]): For each variable node on the Tanner graph, the min-sum algorithm computes, after m iterations, the lowest cost configuration on the computation tree of depth m rooted at that variable node. The output of the algorithm is the vector $(c_1, \ldots, c_n) \in \{0, 1\}^n$ such that c_i is the value at the root node of the lowest cost configuration on the depth m tree rooted at the i^{th} variable node of T.

In analogy with the graph cover and linear programming situations, we define a *computation tree pseudocodeword* to be any configuration on any computation tree. Note that if $(c_1, \ldots, c_n)$ is a codeword, then, for any computation tree R, the assignment $\mathbf{c} = (c_x)_{x \in X(R)}$, given by $c_x = c_i$ if x

is a copy of the i^{th} variable node of T, is a configuration on R. Therefore, every codeword is a computation tree pseudocodeword. In most cases, there are computation tree pseudocodewords which are not codewords. We call such computation tree pseudocodewords *nontrivial*.

Example 5.5: [See also [5].] It can be shown that there are no nontrivial computation tree pseudocodewords for the Tanner graph T of Example 3.2. However, the addition of a new, redundant check allows for nontrivial computation tree pseudocodewords. Let T_1 be the Tanner graph of Figure 4. Then the code determined by T_1 is again the $[4, 1, 4]$ repetition code, but Figure 5 shows a nontrivial computation tree pseudocodeword for T_1.

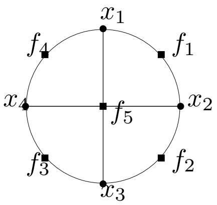

Fig. 4. The Tanner graph T_1 of Example 5.5.

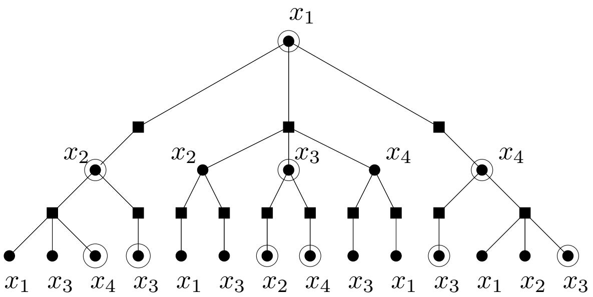
x_1
x_2
x_2
x_3
x_4
x_4
x_1 x_3 x_4 x_3 x_1 x_3 x_2 x_4 x_3 x_1 x_3 x_1 x_2 x_3

Fig. 5. A computation tree of depth 2 rooted at x_1 for the Tanner graph T_1 of Example 5.5. Labels on the check nodes are omitted for clarity. A nontrivial computation tree pseudocodeword is shown on the tree, where the ringed variable nodes are assumed to be set to "1" and the others to "0".

In light of Wiberg's theorem, we see that it is the nontrivial computation tree pseudocodewords which are impediments to correct decoding with iterative message-passing algorithms. This simple observation was certainly known to Wiberg at the time of his thesis in 1996. However, perhaps because of stark contrast between the complicated nature of computation trees and the elegance of finite covers, the pseudocodeword literature has focused almost exclusively on graph cover pseudocodewords. In the next section, we propose a new decoding rule, the study of which should shed light on computation tree pseudocodewords, including their connection to graph cover pseudocodewords.

VI. Universal Cover Decoding

The connection between computation trees and finite covers can be found in the *universal cover*, a notion from topology which we now define in the context of graph theory. For more information, see [10] or [9].

Definition 6.1: Let G be a finite connected graph and suppose the cover $\widehat{\pi} : \widehat{G} \to G$ enjoys the following universal property: For any finite connected cover $\pi : \widetilde{G} \to G$ of G, there is a covering map $\widetilde{\pi} : \widehat{G} \to \widetilde{G}$ such that $\pi \circ \widetilde{\pi} = \widehat{\pi}$. Then $\widehat{\pi} : \widehat{G} \to G$ is called a *universal cover* of G.

It can be shown (see [10] or [9]) that every finite connected graph G has a unique (up to graph isomorphism) universal cover. The universal cover $\widehat{G}$ of G is always a tree, and it is an infinite tree (i.e., it has infinitely many vertices) if and only if G is not a tree. Moreover, $\widehat{G}$ can be constructed from G by following the computation tree construction for infinite depth with any vertex of G as the root node. The importance of the universal cover in terms of decoding is that every finite connected cover of a Tanner graph T is a surjective image of the universal cover of T, and every computation tree for T is a subgraph of the universal cover of T.

Definition 6.2: Let $T = (X \cup F, E)$ be a Tanner graph and $\widehat{\pi} : \widehat{T} \to T$ the universal cover of T. Set

$$\widehat{X} := \bigcup_{x \in X} \widehat{\pi}^{-1}(x) \qquad \text{and} \qquad \widehat{F} := \bigcup_{f \in F} \widehat{\pi}^{-1}(f).$$

We call $\widehat{X}$ the set of *variable nodes* of $\widehat{T}$ and $\widehat{F}$ the set of *check nodes* of $\widehat{T}$. A *configuration* on $\widehat{T}$ is an assignment $\widehat{\mathbf{c}} = (\widehat{c}_{\widehat{x}})_{\widehat{x} \in \widehat{X}}$ of 0's and 1's to the variable nodes of $\widehat{T}$ in such a way that the binary sum of the neighbors of each check node in $\widehat{T}$ is 0. A *universal cover pseudocodeword* for T is a configuration on $\widehat{T}$.

The next proposition shows how graph cover pseudocodewords, computation tree pseudocodewords, and universal cover pseudocodewords are related. Recall that linear programming pseudocodewords and graph cover pseudocodewords were shown to coincide by Vontobel and Koetter [12].

Proposition 6.3: Let T be a Tanner graph. Then

1) Every computation tree pseudocodeword for T extends to a universal cover pseudocodeword.
2) Every graph cover pseudocodeword for T which has a connected graph cover realization induces a universal cover pseudocodeword for T.
3) Every universal cover pseudocodeword induces a computation tree pseudocodeword on every computation tree for T.

Proof: Let $\widehat{\pi} : \widehat{T} \to T$ be the universal cover of T.

1) Suppose $\mathbf{c} = (c_x)_{x \in X(R)}$ is a computation tree pseudocodeword on some computation tree R of T, rooted at the variable node v. Thinking of $\widehat{T}$ as a computation tree of infinite depth rooted at v, $\mathbf{c}$ can be superimposed onto the top portion of $\widehat{T}$ and, since there are no cycles in $\widehat{T}$, it can be extended (possibly in several ways) to a configuration on all of $\widehat{T}$.
2) Suppose $\mathbf{p}$ is an unscaled graph cover pseudocodeword with a connected graph cover realization $(\widetilde{T}, \widetilde{\mathbf{c}})$, where $\widetilde{T}$ is a connected finite cover of T and $\widetilde{\mathbf{c}}$ is a configuration on $\widetilde{T}$. Since $\widetilde{T}$ is connected, there is a covering map $\widetilde{\pi} : \widehat{T} \to \widetilde{T}$ and $\mathbf{p}$ induces a configuration $\widehat{\mathbf{c}} = (\widehat{c}_{\widehat{x}})_{\widehat{x} \in \widehat{X}}$ on $\widehat{T}$ by setting $\widehat{c}_{\widehat{x}} = c_{\widetilde{\pi}(\widehat{x})}$ for each $\widehat{x} \in \widehat{X}$.
3) Suppose $\widehat{\mathbf{c}}$ is a universal cover pseudocodeword and R is a computation tree for T, rooted at the variable node

v of T. Then $\widehat{T}$ can be drawn as an infinite computation tree for T, rooted at v, and the truncation of $\widehat{\mathbf{c}}$ to R yields a computation tree pseudocodeword on R.

$\blacksquare$

Note that the restriction to graph cover pseudocodewords with connected realizations in part (2) of Proposition 6.3 does not pose a problem since the output of the graph cover decoding algorithm is always a graph cover pseudocodeword with a connected realization by Proposition 3.3.

In order to propose a definition for universal cover decoding, we need some notion of cost for a configuration on the universal cover of a Tanner graph. As the universal cover is the infinite computation tree, the idea is that the cost of a configuration on the universal cover ought to be the limit of the costs of the truncated versions of the configuration on finite computation trees. Since the cost of a configuration on a computation tree is defined as a sum over all variable nodes in the computation tree and the number of variable nodes grows exponentially with the depth of the tree, we must first define a normalized computation tree cost function.

Definition 6.4: Let T be a Tanner graph, let R be a (finite) computation tree for T, and let $X(R)$ be the set of variable nodes of R. For any configuration $\mathbf{c} = (c_x)_{x \in X(R)}$ on R, let $G(\mathbf{c})$ be the cost of $\mathbf{c}$, as defined in Definition 5.3. The *normalized cost* of $\mathbf{c}$ on R is

$$\overline{G}(\mathbf{c}) := \frac{1}{|X(R)|} G(\mathbf{c}).$$

Definition 6.5: Let T be a Tanner graph and let v be a variable node of T. Let $\widehat{T}_v$ be the universal cover of T, realized as an infinite computation tree rooted at v. For any positive integer m, let $R_v^{(m)}$ be the computation tree of depth m rooted at v, so that $R_v^{(m)}$ is formed after truncating $\widehat{T}_v$ after level $2m$. For any configuration $\mathbf{c}_v$ on $\widehat{T}_v$, let $\mathbf{c}_v^{(m)}$, $m \geq 1$, be the truncation of $\mathbf{c}_v$ to $R_v^{(m)}$, and let $\overline{G}_v^{(m)}(\mathbf{c}_v^{(m)})$ be the normalized cost of $\mathbf{c}_v^{(m)}$ on $R_v^{(m)}$. The *rooted v-cost* of the configuration $\mathbf{c}_v$ on the infinite computation tree $\widehat{T}_v$ is defined as

$$G_v(\mathbf{c}) := \limsup_{m \to \infty} \overline{G}_v^{(m)}(\mathbf{c}^{(m)}).$$

For any variable node v of T and any variable node $\hat{v}$ of the universal cover $\widehat{T}$ lying over v, we can root $\widehat{T}$ at $\hat{v}$ to obtain the infinite computation tree $\widehat{T}_v$ rooted at v. Thus, for any configuration $\mathbf{c}$ on $\widehat{T}$ and any variable node $\hat{v}$ of $\widehat{T}$ lying over v, we get a configuration $\mathbf{c}_{\hat{v}}$ on $\widehat{T}_v$. We conjecture that, for many universal cover configurations $\mathbf{c}$, the rooted v-cost of $\mathbf{c}_{\hat{v}}$ is independent of choice of $\hat{v}$ lying over v, so that the v-cost of $\mathbf{c}$ is defined in a non-rooted manner. Moreover, we conjecture that, for many universal cover configurations, this v-cost is actually independent of v, so that one can discuss the *cost* of a universal cover configuration. In particular, we have:

Theorem 6.6 ([1]): Let T be the (finite, connected) Tanner graph of a (d_v, d_c)-regular LDPC code, with $d_v \geq 3$ being the degree of each variable node and $d_c \geq 3$ being the degree of each check node. Let $\widehat{T}$ be the universal cover of T and let $\mathbf{c}$ be a configuration on $\widehat{T}$ which is induced by a connected graph cover pseudocodeword. Then for any $v \in X(T)$ and any $\hat{v} \in X(\widehat{T})$ lying over v, the rooted v-cost of $\mathbf{c}_{\hat{v}}$ is independent of choice of $\hat{v}$ and is equal to

$$\lim_{m \to \infty} \overline{G}_v^{(m)}(\widehat{c}^{(m)}).$$

Moreover, the value of this limit is independent of the choice of v.

Note that requiring the minimum degree of T to be at least three is not a serious restriction. If $d_c \leq 2$, the code defined by T is either the entire vector space, the zero code or the repetition code. Furthermore, if $d_v \leq 2$, we either get the entire vector space, a code consisting of all even weight vectors in $\mathbb{F}_2^n$ or a cycle code.

We can now define *universal cover decoding*.

Definition 6.7: Assume the code C with Tanner graph T having variable nodes $\{x_1, \ldots, x_n\}$ is used on a binary-input, memoryless channel. *Universal cover decoding* is the decoding algorithm given by the following decision rule: For a given channel output and each $i = 1, ..., n$, find a configuration $\mathbf{c}_i$ on $\widehat{T}_{x_i}$ of minimal rooted x_i-cost. Return $\omega = (\omega_1, \ldots, \omega_n)$, where ω_i is the probability that a random copy $\hat{x}_i$ of x_i in $\widehat{T}_{x_i}$ is assigned a value of one by $\mathbf{c}_i$.

Because the universal cover is infinite for any graph that is not a tree, universal cover decoding cannot be simulated; rather, it should be seen as a theoretical tool that formally ties together graph cover decoding and iterative message-passing decoding. Indeed, graph cover decoding is a sub-decoder of universal cover decoding in the sense that both seek to minimize the same cost but universal cover decoding sees all configurations on the universal cover whereas graph cover decoding sees only those configurations induced by graph cover pseudocodewords. On the other hand, universal cover decoding is the limit of the min-sum decoder, in the sense that both seek to minimize the same cost but universal cover decoding operates on the infinite computation tree whereas min-sum operates on a finite computation tree, i.e., universal cover decoding allows for infinitely many iterations whereas min-sum must run for only finitely many iterations. See [1] for details.

References

[1] N. Axvig, K. Morrison, E. Psota, D. Turk, L.C. Pérez, and J.L. Walker. “Universal cover decoding”. In preparation, 2007.

[2] C. Berrou, A. Glavieux, and P. Thitimajshima. “Near Shannon limit error-correcting coding and decoding”. In *Proceedings of the 1993 IEEE International Conference on Communications*, pages 1064–1070, Geneva, Switzerland, 1993.

[3] J. Feldman. *Decoding Error-Correcting Codes via Linear Programming*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 2000.

[4] R. G. Gallager. *Low-Density Parity Check Codes*. MIT Press, Cambridge, MA, 1963.

[5] C. Kelley and D. Sridhara. “Pseudocodewords of Tanner graphs”. To appear in *IEEE Transactions on Information Theory*.

[6] R Koetter, W.-C. W. Li, P. O. Vontobel, and J. L. Walker. “Characterizations of pseudo-codewords of LDPC codes”. *Advances in Mathematics*, 213:205–229, 2007.

[7] Y. Li, B. Vucetic, F. Jiang, and L. C. Pérez. “Recent Advances in Turbo Code Design and Theory”. *Proceedings of the IEEE*, 95:1323–1344, June 2007.

[8] D. J. C. MacKay and R. M. Neal. “Near Shannon limit performance of low-density parity check codes”. *IEE Electronic Letters*, 32(18):1645–1646, August 1996.

[9] V. V. Prasolov. *Elements of combinatorial and differential topology*, volume 74 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 2006. Translated from the 2004 Russian original by Olga Sipacheva.

[10] J.-P. Serre. *Trees*. Springer Monographs in Mathematics. Springer-Verlag, Berlin, 2003. Translated from the French original by John Stillwell, Corrected 2nd printing of the 1980 English translation.

[11] C. E. Shannon. “A mathematical theory of communication”. *Bell System Technical Journal*, 27:379–423 and 623–656, July and October 1948.

[12] P. Vontobel and R. Koetter. “Graph-cover decoding and finite-length analysis of message-passing iterative decoding of LDPC codes”. To appear in *IEEE Transactions on Information Theory*.

[13] N. Wiberg. *Codes and Decoding on General Graphs*. PhD thesis, Linköping University, Linköping, Sweden, 1996.