7-7-2008

A Universal Theory of Decoding and Pseudocodewords

Nathan Axvig
s-naxvig1@math.unl.edu

Deanna Dreher
University of Nebraska - Lincoln

Katherine Morrison
University of Nebraska-Lincoln

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

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

See next page for additional authors

Axvig, Nathan; Dreher, Deanna; Morrison, Katherine; Psota, Eric T.; Pérez, Lance C.; and Walker, Judy L., "A Universal Theory of Decoding and Pseudocodewords" (2008). *Faculty Publications, Department of Mathematics*. 165.
https://digitalcommons.unl.edu/mathfacpub/165

Authors
Nathan Axvig, Deanna Dreher, Katherine Morrison, Eric T. Psota, Lance C. Pérez, and Judy L. Walker

A Universal Theory of Decoding and Pseudocodewords

SGER Technical Report 0801*

Nathan Axvig[†], Deanna Dreher[†], Katherine Morrison[†], Eric Psota[‡],

Lance C. Pérez[‡], and Judy L. Walker[†]

July 7, 2008

*This work was supported in part by NSF grants DMS-0735099 and DMS-0602332 and AFOSR Grant FA9550-06-1-0375.

[†] N. Axvig, D. Dreher, K. Morrison, and J. L. Walker are with the Department of Mathematics, University of Nebraska, Lincoln, NE 68588-0130, USA, {s-naxvig1,s-dturk1,s-kmorri11,jwalker}@math.unl.edu

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

1 Introduction

The discovery of turbo codes [5] and the subsequent rediscovery of low-density parity-check (LDPC) codes [9, 18] 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. In this sense, these codes are said to be *near-capacity-achieving* codes and are sometimes considered to have solved (in the engineering sense, at least) the coding problem for the additive white Gaussian noise (AWGN) channel and its derivative channels.

Perhaps the most important commonality between turbo and low-density parity-check codes is that they both utilize iterative message-passing decoding algorithms. For turbo codes, one uses the so-called turbo decoding algorithm, and for LDPC codes, both the sum-product (SP) and the min-sum (MS) algorithms are used. The success of the various iterative message-passing algorithms is sometimes said to have ushered in a new era of "modern" coding theory in which the design emphasis has shifted from optimizing some code property, such as minimum distance, to optimizing the corresponding decoding structure of the code, such as the degree profile [24, 25], with respect to the behavior of a message-passing decoder. As successful as these codes and decoders have been in terms of application, there are several major questions that must be answered before a complete understanding of them can be achieved.

The theoretical research in the area of capacity-achieving codes is focused on two main themes. The first theme is whether different types of capacity-achieving codes have common encoder and structural properties. In [17], it was claimed that turbo codes could be viewed as LDPC codes, but the relationship was not made explicit. More recently, Pérez, his student Jiang, and others [11, 16] developed a construction for the parity-check matrices of arbitrary turbo codes that clearly connects the components of the turbo encoder to the resulting structure of the parity-check matrix. From a more abstract perspective, turbo codes and low-density parity-check codes are examples of codes with long block lengths that exhibit the random structure inherent in Shannon's original theorems.

The second and more active research theme is the determination of the behavior of iterative message-passing decoding and the relationships between the various decoding algorithms. The dominant problem in this area is to understand the non-codeword decoder errors that occur in computer simulations of LDPC codes with iterative message-passing decoders. Motivated by empirical observations of the non-codeword outputs of LDPC decoders, the notion of *stopping sets* was first introduced by Forney, *et al.* [8] in 2001. Two years later, a formal definition of stopping sets was given by Changyan, *et al.* [6]. 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. (Here, a stopping set S is a subset of the set of variable nodes such that all neighboring check nodes of S are connected to S at least twice.)

The intuition behind stopping sets begins with an understanding of message-passing algorithms. Information given to a specific variable node from a neighboring check node is derived from all other variable nodes connected to that check node. If two variable nodes with erasures are connected to a common check node, then the check node is not able to determine the value of either of them. For this reason, the check nodes connected to a stopping set are incapable of resolving erasures if every variable node in the stopping set begins with an erasure.

Decoding errors for iterative message-passing algorithms are also often attributed to *pseudocodewords*, the various notions of which are central to this paper and will be defined and studied in detail later. Work on relating pseudocodewords to stopping sets for the BEC [8], the binary symmetric channel (BSC) and the AWGN channel [13] 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 [31], 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* [24], [25], 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 tractable [29]. 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. This report is concerned with the analysis, performance and design of iterative message-passing decoding algorithms for LDPC codes with an emphasis on the role of the various notions of pseudocodewords.

The formal study of pseudocodewords and their role in iterative message-passing decoders necessarily begins with several definitions. For additional background in graph theory, we refer the reader to [30].

Definition 1.1. A *(simple) graph* G is a pair (V, E), where V is a nonempty set and E is a (possibly empty) subset of the collection of unordered pairs of distinct elements of V. Elements of V are called *vertices* and elements of E are called *edges*. If an edge e represents the unordered pair (u, v) of vertices, we sometimes write $e = uv$ and call u and v the *endpoints* of e. The graph G is *finite* if V is a finite set. For $v \in V$, the *neighborhood* of v is the set $N(v)$ of vertices $u \in V$ such that $(u, v) \in E$. Elements of $N(v)$ are called *neighbors* of v, and the *degree* of v is the number of neighbors v has. We say G is *d-regular* if every vertex in G has degree d. A *path* in G is a finite sequence of distinct vertices $v_0, \ldots, v_k$ of G such that v_{i-1} and v_i are neighbors for $1 \leq i \leq k$. A *cycle* in G is a finite sequence of vertices $v_0, \ldots, v_k$ of G such that the sequence $v_0, \ldots, v_{k-1}$ is a path in G and $v_0 = v_k$ is a neighbor of v_{k-1}. 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 (d_X, d_F)*-regular* if the degree of every vertex in X is d_X and the degree of every vertex in F is d_F. We say G is a *tree* if G is connected and has no nontrivial cycles.

The success of low-density parity-check codes stems from the fact that each such code comes 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 1.2 (Tanner [28])**.** A *Tanner graph* is a finite, connected, bipartite graph $T = (X \cup F, E)$ such that $\deg(f) \geq 2$ for all $f \in F$. We call X the set of *variable nodes* of T and F the set of *check nodes* of T. A *(valid) configuration* on a Tanner graph T is an assignment $\mathbf{c} = (c_x)_{x \in X}$ of 0's and 1's to the variable nodes of T such that, at each check node f of T, the binary sum of the values at the neighbors of f is 0. The collection of configurations on a Tanner graph T is called the *(LDPC) code determined by T*.

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

Given a binary linear code, *i.e.*, a subspace $C \subseteq \mathbb{F}_2^n$, there is a one-to-one correspondence between Tanner graphs for C and parity-check matrices for C. Indeed, if $H = (h_{ji})$ is an

$r \times n$ binary matrix, then we associate a Tanner graph $T = T(H) = (X(H) \cup F(H), E(H))$ to H by setting

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

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

A significant problem of practical interest is to transmit a codeword $\mathbf{c}$ of some code C across a noisy channel and then to compute an estimate $\widehat{\mathbf{c}}$ based on the channel output. For the remainder of this report, codewords are assumed to be transmitted using binary antipodal modulation across the AWGN channel. The process of computing the estimate $\widehat{\mathbf{c}}$ based on the channel output and knowledge of the code is called *decoding*. Decoding can result in the following three outcomes:

1. $\widehat{\mathbf{c}} = \mathbf{c}$, called a *decoding success*
2. $\widehat{\mathbf{c}} = \mathbf{c}'$, where $\mathbf{c}' \in C \setminus \{\mathbf{c}\}$, called a *decoding error*
3. $\widehat{\mathbf{c}} = \mathbf{r}$, where $\mathbf{r} \notin C$, called a *decoding failure*.

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

Wiberg [31] showed that iterative message-passing decoders such as sum-product and min-sum actually operate on finite *computation trees* associated to the Tanner graph. As *computation tree pseudocodewords* will be one major focus of this report, we now make these notions precise.

Definition 1.3 (Wiberg [31], p.27)**.** 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 $R = R_v^{(m)}$ 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 [23] used in the iterative message-passing algorithm. However, a computation tree corresponding to m iterations 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, 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 $2m$, *i.e.*, the level consisting of the leaf nodes of the tree) in the computation tree is adjacent to exactly one copy of f and every copy of f in the computation tree is adjacent to exactly one copy of x.

For min-sum and sum-product decoding, the decoder considers as competitors all valid configurations on $n := |X(T)|$ computation trees [31] 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 $\mathbf{c} = (c_1, \ldots, c_n)$ 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 1.4. Let T be a Tanner graph. A *computation tree pseudocodeword* for T is any valid configuration on any computation tree for T. A *nontrivial computation tree pseudocodeword* is a computation tree pseudocodeword that does not correspond to a codeword.

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

Definition 1.5 (See, for example, [27], p.130)**.** 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 $\widetilde{v} \in \pi^{-1}(v)$, the neighborhood of $\widetilde{v}$ is mapped bijectively to the neighborhood of v via π. For a positive integer M, an *M-cover* of G is a cover $\pi : \widetilde{G} \to G$ such that $\pi^{-1}(v)$ contains exactly M vertices of $\widetilde{G}$ for each vertex v of G. If $\widetilde{G}$ is an M-cover of G, we say the *degree* of $\widetilde{G}$ is M.

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

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

Definition 1.6 (See, for example, [15])**.** Let T be a Tanner graph for a binary linear code

C and let $\widetilde{\mathbf{c}} = (c_{11}, \ldots, c_{1M} : \cdots : c_{n1}, \ldots, c_{nM})$ be a codeword in some code $\widetilde{C}$ corresponding to some M-cover $\widetilde{T}$ of T. The *(unscaled) graph cover pseudocodeword* corresponding to $\widetilde{\mathbf{c}}$ is the vector

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

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

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

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

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

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

Often, given an unscaled or normalized graph cover pseudocodeword, we will want to consider configurations on covers of the Tanner graph that yield that pseudocodeword. This motivates the next definition.

Definition 1.7. Let T be a Tanner graph, let $\mathbf{p}$ be an unscaled graph cover pseudocodeword for T and let $\boldsymbol{\omega}$ be a corresponding normalized graph cover pseudocodeword. A *realization* for $\mathbf{p}$, or for $\boldsymbol{\omega}$, is a pair $(\widetilde{T}, \widetilde{\mathbf{c}})$, where $\widetilde{T}$ is a cover of T and $\widetilde{\mathbf{c}}$ is a configuration on $\widetilde{T}$ such that $\mathbf{p} = \mathbf{p}(\widetilde{\mathbf{c}})$ and $\boldsymbol{\omega} = \boldsymbol{\omega}(\widetilde{\mathbf{c}})$. The realization $(\widetilde{T}, \widetilde{\mathbf{c}})$ is called a *connected realization* if $\widetilde{T}$ is a connected graph. If a connected realization exists for $\mathbf{p}$, or for $\boldsymbol{\omega}$, then $\mathbf{p}$, or $\boldsymbol{\omega}$, is called a *connected graph cover pseudocodeword*.

Intuitively, all codewords on all covers of the Tanner graph are competitors in iterative message-passing decoding algorithms. In this vein, Vontobel and Koetter [29] 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 that, in a certain precise sense, provides the best explanation of the channel output. They show that graph cover decoding is equivalent to *linear programming (LP) decoding*, as defined by Feldman [7]. We now turn to the formal definition of LP decoding.

Definition 1.8 (Feldman [7])**.** Let $H = (h_{ji})$ be the $r \times n$ parity-check matrix with corresponding Tanner graph T, and, for $1 \leq j \leq r$, set

$$N(j) = \{i \,|\, h_{ji} = 1\} \subseteq \{1, \ldots, n\}$$

so that $N(j)$ is the set of variable nodes adjacent to check node j in T. The *fundamental polytope* $\mathcal{P} = \mathcal{P}(H)$ is the subset of the unit hypercube $[0,1]^n \subset \mathbb{R}^n$ consisting of all vectors $\mathbf{x} = (x_1, \ldots, x_n)$ such that for $1 \leq j \leq r$ 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 $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_n)$ determined by the channel output and for any $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$, the *cost* $\gamma(\mathbf{x})$ of $\mathbf{x}$ is given by

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

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

Since the cost function is linear and the polytope is defined by linear inequalities, the output of linear programming decoding may always be taken to be a vertex of the fundamental polytope. Moreover, since the linear inequalities have integer coefficients, the vertices have rational coordinates. Feldman [7] 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 1.9. A *linear programming pseudocodeword* of a code defined by the parity-check matrix H is any vertex of the fundamental polytope $\mathcal{P}(H)$. A *nontrivial linear programming pseudocodeword* is a linear programming pseudocodeword that is not a codeword.

Remark 1.10. *This definition of linear programming pseudocodewords is different from that given by Feldman on page 64 of [7]; the definition there coincides precisely with the unscaled graph cover pseudocodewords.*

Vontobel and Koetter [29] show that the collection of rational points in the fundamental polytope is precisely the collection of normalized graph cover pseudocodewords. Thus, with the definitions here, every linear programming pseudocodeword is a normalized graph cover pseudocodeword, but not vice versa.

2 Linear Programming and Graph Cover Decoding

In this section we review the known connections between linear programming decoding and graph cover decoding and further examine the relationship between both of these and maximum likelihood decoding. We also analyze properties of graph cover and LP pseudocodewords, in particular with respect to their realizations on covers of the original Tanner graph.

In [29], Vontobel and Koetter show that linear programming decoding and graph cover decoding are equivalent in the sense that, for a given channel output, graph cover and linear programming decoding return the same vector of rational numbers between zero and one (assuming ties are resolved in the same way for both decoders). Thus, in order to understand the behavior of graph cover decoding it is sufficient to understand that of LP decoding and vice versa. In particular, it is possible to infer relationships between graph cover decoding and maximum likelihood (ML) decoding from known relationships between LP and ML decoding. Feldman [7] shows that LP decoding has the *ML certificate property*, which guarantees that if LP decoding returns a codeword, then this codeword is an ML codeword. This fact does not necessarily imply that LP decoding is equivalent to ML decoding, as will be seen in Theorem 2.2 below for the special case where the AWGN channel is used. The proof of Theorem 2.2 requires the following result from [14].

Proposition 2.1 (Koetter, *et al.* [14], Proposition 2.12)**.** *Let C be the parity-check code determined by the Tanner graph T with n variable nodes. Suppose that $\boldsymbol{\omega}$ is a nontrivial LP pseudocodeword of C. Then, for some vector $\boldsymbol{\lambda}$ of log-likelihood ratios, the cost $\sum_{i=1}^{n} \lambda_i \omega_i$ is smaller than the cost $\sum_{i=1}^{n} \lambda_i c_i$ of any codeword $\mathbf{c} \in C$.*

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

Over the additive white Gaussian noise channel, maximum likelihood decoding is based on the squared Euclidean distance between modulated points, where the modulation map $\mathfrak{m} : \{0,1\}^n \to \{\pm 1\}^n$ is given by $\mathfrak{m}(\mathbf{x}) = 2\mathbf{x} - \mathbf{1}$, with $\mathbf{1} = (1, \ldots, 1)$. For a received vector $\mathbf{y}$, the output $\widehat{\mathbf{x}}^{\mathrm{ML}}$ of ML decoding is given by

$$\widehat{\mathbf{x}}^{\mathrm{ML}} = \mathfrak{m}^{-1}\left(\operatorname*{argmin}_{\mathbf{x} \in \mathfrak{m}(C)} \sum_{i=1}^{n} (y_i - x_i)^2 \right),$$

where n is the length of the code. Let $\mathcal{V} = \mathcal{V}(\mathcal{P})$ denote the set of vertices of the fundamental polytope. The modulation map $\mathfrak{m}$ extends naturally to $\mathfrak{m} : \mathbb{R}^n \to \mathbb{R}^n$ via the same formula, and the natural extension of the ML decision rule in this case is the *generalized maximum likelihood (GML)* decision rule given by

$$\widehat{\mathbf{x}}^{\mathrm{GML}} = \mathfrak{m}^{-1}\left(\operatorname*{argmin}_{\mathbf{v}\in\mathfrak{m}(\mathcal{V})} \sum_{i=1}^{n} (y_i - v_i)^2\right) \tag{2.1}$$

$$= \mathfrak{m}^{-1}\left(\operatorname*{argmin}_{\mathbf{v}\in\mathfrak{m}(\mathcal{V})} \sum_{i=1}^{n} y_i^2 - 2y_i v_i + v_i^2\right), \tag{2.2}$$

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

Theorem 2.2 ([1], Theorem 3.2)**.** *Suppose the code C defined by the parity-check matrix H is used on the additive white Gaussian noise channel. Then the following are equivalent:*

1. *C has no nontrivial linear programming pseudocodewords with respect to H.*

2. *Linear programming decoding for C with respect to H is equivalent to maximum likelihood decoding for C.*

3. *Linear programming decoding for C with respect to H and generalized maximum likelihood decoding for C with respect to H, as in Equation* (2.2)*, use the same decision rule.*

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

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

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

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

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

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

Hence the decision rule for linear programming decoding can be reformulated as

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

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

$$\widehat{\mathbf{x}}^{\mathrm{LP}} = \mathfrak{m}^{-1}\left(\operatorname*{argmin}_{\mathbf{v}\in\mathfrak{m}(\mathcal{V})} \sum_{i=1}^{n} (y_i^2 - 2y_i v_i) \right). \tag{2.3}$$

If C has no nontrivial linear programming pseudocodewords with respect to H, then $\mathfrak{m}(\mathcal{V}) = \mathfrak{m}(C)$ and $\sum_{i=1}^{n} v_i^2 = n$, since $v_i = \pm 1$ for all $\mathbf{v}\in\mathfrak{m}(C)$. Thus, adding v_i^2 to the i^{th}

term in the sum does not change the minimization problem, and

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

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

Conversely, if C has a nontrivial linear programming pseudocodeword with respect to H, then there is a vector $\boldsymbol{\lambda}$ of log-likelihood ratios such that $\sum_{i=1}^{n}\lambda_i\omega_i$ is smaller than the cost $\sum_{i=1}^{n}\lambda_ic_i$ of any codeword $\mathbf{c}\in C$, by Proposition 2.1. Over the AWGN channel the received vector may be any vector in $\mathbb{R}^n$, so we can construct a received vector $\mathbf{y}$ such that the resulting log-likelihood vector is $\boldsymbol{\lambda}$. Thus, if $\mathbf{y}$ is received, LP decoding will return a nontrivial pseudocodeword, whereas ML decoding will always return a codeword. Therefore, LP and ML are not equivalent. Furthermore, because C has a nontrivial LP pseudocodeword with respect to H, $\sum_{i=1}^{n}v_i^2$ is not constant over $\mathbf{v}\in\mathfrak{m}(\mathcal{V})$, and hence the LP decision rule as given in Equation 2.3 differs from the GML decision rule as given in Equation 2.2. Therefore, LP and GML are not equivalent either. ■

Remark 2.3. *The proof of Theorem 2.2 cannot be applied to show similar results about when linear programming decoding is not equivalent to maximum likelihood decoding or generalized maximum likelihood decoding for the binary symmetric channel or the binary erasure channel because the set of possible received vectors for each of these channels is finite, and hence the set of possible log-likelihood ratios is also finite. Given a target vector of log-likelihood ratios* $\boldsymbol{\lambda}\in\mathbb{R}^n$*, it is impossible to guarantee that there is a received vector that yields* $\boldsymbol{\lambda}$*. For more* *discussion on this issue, see [26].*

Theorem 2.2 shows that linear programming decoding and graph cover decoding differ from even the generalized version of maximum likelihood decoding when nontrivial pseudocodewords are present. In order to better understand these decoders, we must further study properties of their pseudocodewords. One characterization is given by Koetter, *et al.* [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

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

Vontobel and Koetter point out an analogous characterization of normalized graph cover pseudocodewords in [29]: a vector $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_n)$ of rational numbers between zero and one lies in the fundamental polytope if and only if it is a normalized graph cover pseudocodeword. These characterizations coincide since the fundamental cone is the conic hull of the fundamental polytope [29]. While these results elegantly describe graph cover pseudocodewords, they alone do not provide insight into the realizations of these pseudocodewords as coming from codewords on covers of the Tanner graph. In particular, given a normalized graph cover pseudocodeword $\boldsymbol{\omega}$, we are interested in understanding the minimum degree of a cover $\widetilde{T}$ of T on which $\boldsymbol{\omega}$ can be realized. With this question in mind, we make the following definition.

Definition 2.4. A normalized graph cover pseudocodeword $\boldsymbol{\omega}$ for the Tanner graph T is *minimally realizable* on the cover $\widetilde{T}$ of T if there is a configuration $\widetilde{\mathbf{c}}$ on $\widetilde{T}$ such that

1. $\boldsymbol{\omega} = \boldsymbol{\omega}(\widetilde{\mathbf{c}})$, 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 $\widetilde{T}$.

We find an exact value for the degree of a minimal realization of a normalized graph cover pseudocodeword in Proposition 2.6 below, under the assumption we are given the coordinates of the graph cover pseudocodeword as a point in Feldman's *extended polytope* [7], rather than simply in the fundamental polytope. We first recall the definition of the extended polytope.

Definition 2.5 (Feldman [7])**.** Let $H = (h_{ji})$ 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 1.8. For $1 \leq j \leq r$, set

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

where, as before,

$$N(j) = \{i \,|\, h_{ji} = 1\} \subseteq \{1, \ldots, n\},$$

and write $e_j = |E_j|$. Label the coordinates of $\mathbb{R}^{n+e_1+\cdots+e_r}$ as $\{1, \ldots, n\} \cup \{w_{j,S} \,|\, 1 \leq j \leq r \text{ and } S \in E_j\}$, 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

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

and the *extended polytope* of H is the polytope

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

A *minimal realization* of a point in the extended polytope $\mathcal{Q}$ is defined in an analogous fashion to Definition 2.4 above.

Proposition 2.6 ([1], Proposition 4.7)**.** *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 [7] 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 that 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. ■

Proposition 2.6 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 [7] establishes that such an $\bar{\boldsymbol{\omega}}$ always exists, and Vontobel and Koetter [29] 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.

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

Example 2.7 (See also [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

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

and the fundamental polytope is

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

The only connected covers of T are $8M$-cycles for $M \geq 1$, so the only unscaled graph cover pseudocodewords that have connected realizations are those of the form $(0,0,0,0)$ and (M,M,M,M) for $M \geq 1$. Thus the only normalized graph cover pseudocodewords with connected realizations are $(0,0,0,0)$ and $(1,1,1,1)$. In particular, no rational point of $\mathcal{P}$ that is not a vertex of $\mathcal{P}$ has a connected graph cover realization.

Although Example 2.7 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 2.7, these vertices do have connected realizations. This phenomenon happens in general, as shown by the next proposition.

Proposition 2.8 ([4], 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 $\widetilde{T}_i$ is an M_i-cover of T for $1 \leq i \leq k$, 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 $\boldsymbol{\omega}$ is either connected or the disjoint union of connected graph cover realizations of $\boldsymbol{\omega}$.*

Proof. Set $\boldsymbol{\alpha}_i = \boldsymbol{\omega}(\widetilde{\mathbf{c}}_i)$ for $1 \leq i \leq k$. 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 $\frac{M_i}{M} \geq 0$ 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 $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_k$. But each $\boldsymbol{\alpha}_i$ is in $\mathcal{P}$ by [29] and so each $\frac{M_i}{M}\boldsymbol{\alpha}_i$ is too since $\frac{M_i}{M} \leq 1$. 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 $0 < \gamma_i \leq 1$. 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. ■

Theorem 2.10 below gives another sufficient condition for connected realizations of graph cover pseudocodewords to exist. The next lemma will be used in the proof.

Lemma 2.9 ([1], Lemma 5.3)**.** *Let T be a Tanner graph, let T' be a spanning tree of T, and suppose $e_1, \ldots, e_t$ are edges in T not in T'. Let $\pi : \widetilde{T} \to T$ be any finite connected cover of T, and let $\widetilde{e}_i \in \pi^{-1}(e_i)$ be a fixed lift of e_i to $\widetilde{T}$ for each $i = 1, \ldots, t$. Then $\widetilde{T} - \{\widetilde{e}_1, \ldots, \widetilde{e}_t\}$ is connected.*

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

Since T' is a spanning tree for T, there is a path p on T' from $x = \pi(\widetilde{x})$ to $f = \pi(\widetilde{f})$. Then pe_i is a cycle on T containing x and f. By [10, Theorem 2.4.3], $\pi^{-1}(pe_i)$ consists of a disjoint union of cycles that project onto pe_i. Since $\widetilde{e}_i \in \pi^{-1}(e_i) \subset \pi^{-1}(pe_i)$, there is a cycle Γ in $\pi^{-1}(pe_i)$ containing $\widetilde{e}_i$. 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 $\widetilde{e}_j$ for any $j \neq i$. Thus, $\Gamma - \{\widetilde{e}_1, \ldots, \widetilde{e}_t\} = \Gamma - \widetilde{e}_i$ is still connected, and so $\Gamma - \widetilde{e}_i$ contains a path from $\widetilde{x}$ to $\widetilde{f}$ in $\widetilde{T} - \{\widetilde{e}_1, \ldots, \widetilde{e}_t\}$. ■

Theorem 2.10 ([1], Theorem 5.4)**.** *Let T be a Tanner graph with average variable node degree a_X and average check node degree a_F. Suppose that either $a_X \geq 3$ and $a_F \geq 3$, or $a_X \geq 2$ and $a_F \geq 4$. Then any rational point in the fundamental polytope of T can be minimally realized on a connected cover.*

Proof. Let $\boldsymbol{\omega}$ be a rational point in the fundamental polytope of T. Then $\boldsymbol{\omega}$ is a normalized graph cover pseudocodeword [29] and so is minimally realizable on an M-cover for some M. Let $(\widetilde{T}, \widetilde{\mathbf{c}})$ be a realization on an M-cover with a minimal number of connected components. By way of contradiction, suppose $\widetilde{T}$ is not connected, and let $\pi_R : \widetilde{R} \to T$ and $\pi_S : \widetilde{S} \to T$ be distinct connected components of $\widetilde{T}$. We will give an algorithm for connecting these two components $\widetilde{R}$ and $\widetilde{S}$, demonstrating that $\boldsymbol{\omega}$ 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 $\{f_1, \ldots, f_r\}$, then we see the number of edges on T' is at least

$$\sum_{i=1}^{r}(\deg(f_i) - 1) = \left(\sum_{i=1}^{r}\deg(f_i)\right) - r \tag{2.4}$$

$$= ra_F - r \tag{2.5}$$

$$= r(a_F - 1), \tag{2.6}$$

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

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 $na_X = ra_F$, so $n = r\frac{a_F}{a_X}$ and we have that the number of edges on T' is

$$r\frac{a_F}{a_X} + r - 1 = r\left(\frac{a_F}{a_X} + 1\right) - 1.$$

Putting this together with the bound on the number of edges in T' given in (2.6), we see that

$$r\left(\frac{a_F}{a_X} + 1\right) - 1 \geq r(a_F - 1).$$

Thus,

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

and so

$$\frac{a_F}{a_X} + 1 > a_F - 1.$$

Rearranging this we see

$$2 > a_F\left(1 - \frac{1}{a_X}\right).$$

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

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

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

Thus, there is some check node f of T such that at least two edges incident to f are not on T'; let these edges be $e = (f, x)$ and $e' = (f, x')$. Let $\widetilde{e}_R = (\widetilde{f}_R, \widetilde{x}_R)$ and $\widetilde{e}'_R = (\widetilde{f}_R, \widetilde{x}'_R)$ be fixed lifts of e and e', respectively, to $\widetilde{R}$ and let $\widetilde{e}_S = (\widetilde{f}_S, \widetilde{x}_S)$ and $\widetilde{e}'_S = (\widetilde{f}_S, \widetilde{x}'_S)$ be fixed lifts of e and e', respectively, to $\widetilde{S}$. By Lemma 2.9, $\widetilde{R} - \{\widetilde{e}_R, \widetilde{e}'_R\}$ and $\widetilde{S} - \{\widetilde{e}_S, \widetilde{e}'_S\}$ are connected.

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

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

Since, in any case, assuming $\widetilde{R}$ and $\widetilde{S}$ are distinct connected components of $\widetilde{T}$ leads to a minimal realization of $\boldsymbol{\omega}$ on a cover with fewer connected components than $\widetilde{T}$, there must be a minimal realization of $\boldsymbol{\omega}$ on a connected cover of T. ■

3 Weights of Pseudocodewords and C-Symmetry

In order to further understand the impact of linear programming pseudocodewords and graph cover pseudocodewords, we must be able to analyze the probability of decoding error that will arise from a given set of these pseudocodewords. It is well established that the distance between codewords significantly impacts the probability of decoding errors, and thus it is important to further explore the effect of the distance between pseudocodewords and codewords on LP/graph cover decoding performance. For binary linear codes, the classical problem of finding distances between codewords is significantly simplified by looking instead at the weights of codewords, which is made possible by the algebraic structure of the code. To parallel the classical case, we consider the *(effective Hamming) weight* [8] of a pseudocodeword. It should be noted that this notion of weight was originally motivated by the definition of the *generalized weight* of a computation tree configuration, as given by

Wiberg in [31].

Definition 3.1 (See Forney, *et al.* [8], Corollary 3.1)**.** On the additive white Gaussian noise channel, the *(effective Hamming) weight* of a nonzero vector $\mathbf{x} = (x_1, \ldots, x_n)$ of nonnegative rational numbers is given by

$$w(\mathbf{x}) = \frac{\left(\displaystyle\sum_{i=1}^{n} x_i\right)^2}{\displaystyle\sum_{i=1}^{n} x_i^2}.$$

Using the weight measure of Definition 3.1, Forney, *et al.* [8] show that the minimum weight of a vertex of the fundamental polytope determines bounds on linear programming decoding performance. It is important to note that these results deal only with the overall probability of word error when decoding; they say nothing about the probability of word error caused by a given pseudocodeword. Example 3.2 provides an illustration of two pseudocodewords that have the same weight but appear as errors in decoding with significantly different probabilities.

Example 3.2. Consider the code C_1 defined by the parity-check matrix

$$H_1 = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Then C_1 has precisely two codewords: the all-zeros word $\mathbf{c}_0$ and the all-ones word $\mathbf{c}_1$. The fundamental polytope is

$$\mathcal{P}_1 = \left\{ (x_1, \ldots, x_{10}) \in [0,1]^{10} \;\middle|\; \begin{array}{c} x_1 = \cdots = x_5, x_6 = \cdots = x_{10}, \\ x_1 - \frac{2}{3} \le x_6 \le x_1 + \frac{2}{3},\ x_6 \le 5x_1 \le x_6 + 4 \end{array} \right\}.$$

The non-codeword vertices of $\mathcal{P}_1$ are

$$\begin{aligned} \mathbf{p}_1 &= \left(\frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{5}{6}, \frac{5}{6}, \frac{5}{6}, \frac{5}{6}, \frac{5}{6}\right), \\ \mathbf{p}_2 &= \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, 1, 1, 1, 1, 1\right), \\ \mathbf{p}_3 &= \left(\frac{5}{6}, \frac{5}{6}, \frac{5}{6}, \frac{5}{6}, \frac{5}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}\right), \end{aligned}$$

and

$$\mathbf{p}_4 = \left(\frac{2}{3}, \frac{2}{3}, \frac{2}{3}, \frac{2}{3}, \frac{2}{3}, 0, 0, 0, 0, 0\right).$$

One can check that that $w(\mathbf{p}_1) = w(\mathbf{p}_3)$; however, $\mathbf{p}_3$ shows up much less frequently than $\mathbf{p}_1$ in linear programming decoding when the all-zeros word is sent; see Appendix A for details of the simulation. In order to visualize the fundamental polytope, it is convenient to project $\mathcal{P}_1$ into $\mathbb{R}^2$ since $x_1 = x_2 = x_3 = x_4 = x_5$ and $x_6 = x_7 = x_8 = x_9 = x_{10}$. We may then think of the codewords as $\mathbf{c}_0 = (0, 0)$ and $\mathbf{c}_1 = (1, 1)$, and the pseudocodewords as $\mathbf{p}_1 = (\frac{1}{6}, \frac{5}{6})$, $\mathbf{p}_2 = (\frac{1}{3}, 1)$, $\mathbf{p}_3 = (\frac{5}{6}, \frac{1}{6})$, and $\mathbf{p}_4 = (\frac{2}{3}, 0)$. This projected version of the fundamental polytope $\mathcal{P}_1$ is given in Figure 1. Notice that the ray from the origin to $\mathbf{p}_3$ is not an edge of the fundamental polytope, but the ray from the origin to $\mathbf{p}_1$ is. Thus, there are no vertices of the polytope with smaller weight than $\mathbf{p}_1$ that lie between $\mathbf{p}_1$ and the origin, but there is a vertex of smaller weight lying between $\mathbf{p}_3$ and the origin, namely $\mathbf{p}_4$. This might explain the observed difference in the number of errors caused by $\mathbf{p}_1$ and $\mathbf{p}_3$.

Although considering projections of polytopes into lower dimensional spaces can help to visualize these structures, care must be taken when using such projections. The weight of a vector frequently changes under a projection, and pseudocodewords of the same weight may even be mapped to vectors of different weights, as shown in the following example.

Example 3.3. Consider the code $C_2 = \{(0, 0, 0, 0, 0, 0, 0, 0), (1, 1, 1, 1, 1, 1, 1, 1)\}$ defined by

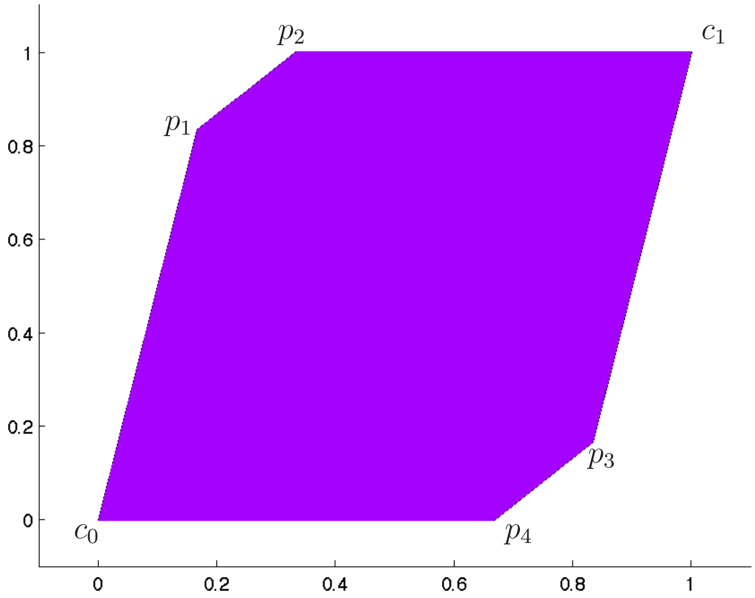

Figure 1: The projected polytope of $\mathcal{P}_1$ into $\mathbb{R}^2$

the parity-check matrix

$$H_2 = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}.$$

The fundamental polytope is

$$\mathcal{P}_2 = \left\{ (x_1, \ldots, x_8) \in [0,1]^8 \;\middle|\; \begin{array}{c} x_1 = \cdots = x_5, x_6 = \cdots = x_8, \\ x_1 - \frac{2}{3} \leq x_6 \leq x_1 + \frac{2}{3},\ x_6 \leq 5x_1 \leq x_6 + 4 \end{array} \right\}$$

and the non-codeword vertices of this polytope are

$$\begin{aligned} \mathbf{q}_1 &= \left(\frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{5}{6}, \frac{5}{6}, \frac{5}{6} \right), \\ \mathbf{q}_2 &= \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, 1, 1, 1 \right), \\ \mathbf{q}_3 &= \left(\frac{5}{6}, \frac{5}{6}, \frac{5}{6}, \frac{5}{6}, \frac{5}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6} \right), \end{aligned}$$

and

$$\mathbf{q}_4 = \left(\frac{2}{3}, \frac{2}{3}, \frac{2}{3}, \frac{2}{3}, \frac{2}{3}, 0, 0, 0 \right).$$

Notice that if we again make the projection into $\mathbb{R}^2$ by identifying $x_1 = x_2 = x_3 = x_4 = x_5$ and $x_6 = x_7 = x_8$, we get the same polytope as in Figure 1. In $\mathcal{P}_2$, we have $w(\mathbf{q}_1) = w(\mathbf{q}_4)$, while the projections of $\mathbf{q}_1$ and $\mathbf{q}_4$ to the vectors $(\frac{1}{6}, \frac{5}{6})$ and $(\frac{2}{3}, 0)$ have weights $\frac{18}{13}$ and 1, respectively.

To analyze the performance of a classical decoder (e.g., a maximum likelihood decoder) on a binary linear code, it is sufficient to consider only the case where the all-zeros codeword is sent. The reason for this is that the linearity of the code implies that for two codewords $\mathbf{c}_1$ and $\mathbf{c}_2$, the codeword $\mathbf{c}_1 - \mathbf{c}_2$ has the same relation to the all-zeros word as the vector $\mathbf{c}_1$ has to $\mathbf{c}_2$. The fundamental polytope of a code displays a similar type of symmetry.

Proposition 3.4 (See Feldman [7])**.** *Any fundamental polytope $\mathcal{P}$ for the binary linear code*

C of length n has the following property: for every $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_n) \in \mathcal{P}$ and $\mathbf{c} \in C$, the point $\boldsymbol{\omega}^{[\mathbf{c}]} = (\omega_1^{[\mathbf{c}]}, \ldots, \omega_n^{[\mathbf{c}]})$ is also in $\mathcal{P}$, where $\omega_i^{[\mathbf{c}]} = |\omega_i - c_i|$ for $i = 1, \ldots, n$.

Because of the symmetry described in Proposition 3.4 above, the fundamental polytope of a binary linear code C is said to have *C-symmetry.* As a consequence of C-symmetry, the probability of word error is independent of the codeword sent, implying that the all-zeros assumption may be used when analyzing the performance of the LP decoder [7]. As was the case with the bounds on performance given by Forney, *et al.* in [8], this result only deals with the probability of error; it says nothing about errors caused by a particular LP pseudocodeword. Additionally, if we allow LP pseudocodewords to be sent across a channel using the generalized modulation map described before Theorem 2.2, we are not guaranteed that the probability of decoding error will be independent of the LP pseudocodeword sent. The next example illustrates this.

Example 3.5. Let

$$C_3 = \begin{array}{l} \{(0,0,0,0,0,0,0,0,0,0,0,0,0,0),(1,1,1,0,0,0,0,0,0,0,0,0,0,0), \\ (0,0,0,0,1,1,1,1,1,0,0,0,0,0),(1,1,1,0,1,1,1,1,1,0,0,0,0,0)\} \end{array}$$

be defined by the parity-check matrix

$$H_3 = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}.$$

Two of the non-codeword vertices of the polytope are

$$\mathbf{p}_1 = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1\right)$$

and

$$\mathbf{p}_2 = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, 1, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, 0, 0, 0, 0, 0\right).$$

Notice that $w(\mathbf{p}_1) = w(\mathbf{p}_2)$.

A simulation was performed in which the linear programming pseudocodewords $\mathbf{p}_1$ and $\mathbf{p}_2$, in addition to all of the codewords, were transmitted across the additive white Gaussian noise channel using the generalized modulation map $\mathfrak{m} : \mathbb{R}^n \to \mathbb{R}^n$ described in the discussion preceding Theorem 2.2. For example, the vector $\mathbf{p}_1 = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1\right)$ was modulated to $\mathfrak{m}(\mathbf{p}_1) = (0, 0, 0, 1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1)$.

For a given signal-to-noise ratio, the four binary codewords and the linear programming pseudocodewords $\mathbf{p}_1$ and $\mathbf{p}_2$ were each transmitted across the additive white Gaussian noise channel 20,000 times. Each received vector was input to the LP decoder and the output of the decoder was recorded. This experiment was performed over a signal-to-noise ratio (SNR) range from 0.0 dB to 6.5 dB in 0.5 dB increments. Table 1 shows the results of this experiment for the SNR value of 3.0 dB; see Appendix B for additional results.

word sent	total errors	errors due to $\mathbf{p}_1$	errors due to $\mathbf{p}_2$
00000000000000	3057	200	100
11100000000000	3033	216	88
00001111100000	3084	4	108
11101111100000	3035	11	110
$\mathbf{p}_1$	3498	—	57
$\mathbf{p}_2$	15113	518	—

Table 1: Simulation results for the code of Example 3.5 at 3.0 dB.

4 Minimal and Irreducible Pseudocodewords

We will again mimic the classical coding case in looking at *minimal codewords*, *i.e.*, codewords whose supports do not properly contain the support of any other nonzero codeword. In particular, we examine different extensions of the theory of minimal codewords to graph cover (and hence LP) pseudocodewords.

Definition 4.1 (Vontobel and Koetter [29], p.27)**.** A *minimal pseudocodeword* is a pseudocodeword $\mathbf{p}$ such that $\{\alpha\mathbf{p} \,|\, \alpha \in \mathbb{R}, \alpha \geq 0\}$ 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 [12]:

Definition 4.2 (Kelley and Sridhara [12], Definition 2.7)**.** An unscaled pseudocodeword is *irreducible* if it cannot be written as a (nontrivial) sum of other unscaled pseudocodewords.

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

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

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

Example 4.4. Let C be the null space of

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

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

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

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

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.

Since the fundamental cone consists of only nonnegative vectors and we consider only linear codes, it always has a vertex at the origin. This means that we can find a set of nonnegative integer vectors that generates the fundamental cone under addition [19]. We had originally conjectured that the set of unscaled minimal pseudocodewords would generate all pseudocodewords in the fundamental cone. However, Example 4.4 above shows that unscaled minimal pseudocodewords certainly do not generate the pseudocodewords in the fundamental cone, since $(1,1,1,1)$ is a pseudocodeword and is not generated by the unscaled minimal pseudocodewords.

Remark 4.5. *It is stated in passing in [12] that the irreducible pseudocodewords correspond to the vertices of the fundamental polytope. A formal proof of this fact is not known to us but, if true, it yields a finite generating set for the set of pseudocodewords. However, enumerating the vertices of the fundamental polytope is still computationally difficult.*

5 Connections Between Min-Sum and Linear Programming/Graph Cover Decoding

Iterative message-passing decoders such as min-sum are computationally efficient suboptimal decoders for low-density parity-check codes. Their efficiency makes them ideal for implementation, but in order to use them effectively it is important to be able to characterize their non-optimality; in other words, it is necessary to understand the errors that arise in iterative message-passing decoding. Theoretical analyses of these errors have been scarce thus far. On the other hand, graph cover decoding has proven much more attractive from a theoretical standpoint. There are multiple characterizations of graph cover pseudocodewords, as mentioned in Section 2, and significant progress has been made in understanding when these pseudocodewords result in decoding errors, e.g., analysis of weights of pseudocodewords; see, for example, [12] and [29]. Graph cover decoding, however, is only a theoretical tool and is not implementable as defined. Furthermore, its equivalent practical decoder, LP, becomes computationally unfeasible at block lengths larger than about 100, and hence is much less desirable than iterative message-passing decoders in practice. Thus, we are motivated to explore possible relationships between iterative message-passing decoders and LP/graph cover decoding to determine if the insights gained for these latter decoders can shed light on the decoding performance of the former. This section focuses primarily on the relationship between LP/graph cover decoding and MS decoding, with MS decoding taken as a specific example of an iterative message-passing decoder.

Iterative message-passing decoders perform operations locally, in that they only take information from immediate neighbors on the Tanner graph T when performing computations. Since every finite cover $\widetilde{T}$ of a Tanner graph T is locally isomorphic to T, a local algorithm cannot distinguish between T and any finite cover of T. Given this, it seems reasonable to assume that iterative message-passing decoders actually operate on all possible covers of the

Tanner graph, as is the case with graph cover decoding. Thus, it is conjectured [29] that iterative message-passing decoders and graph cover decoding approximate each other.

Figure 2 illustrates a problem with this view of graph cover decoding as the correct model for iterative message-passing decoders. The common perception, based on past simulation results, is that LP/graph cover decoding typically performs worse than SP decoding, and better than MS decoding. However, in Figure 2, we get very different behavior: MS and SP both outperform LP decoding, in terms of both word error rate and bit error rate, at all SNRs simulated.

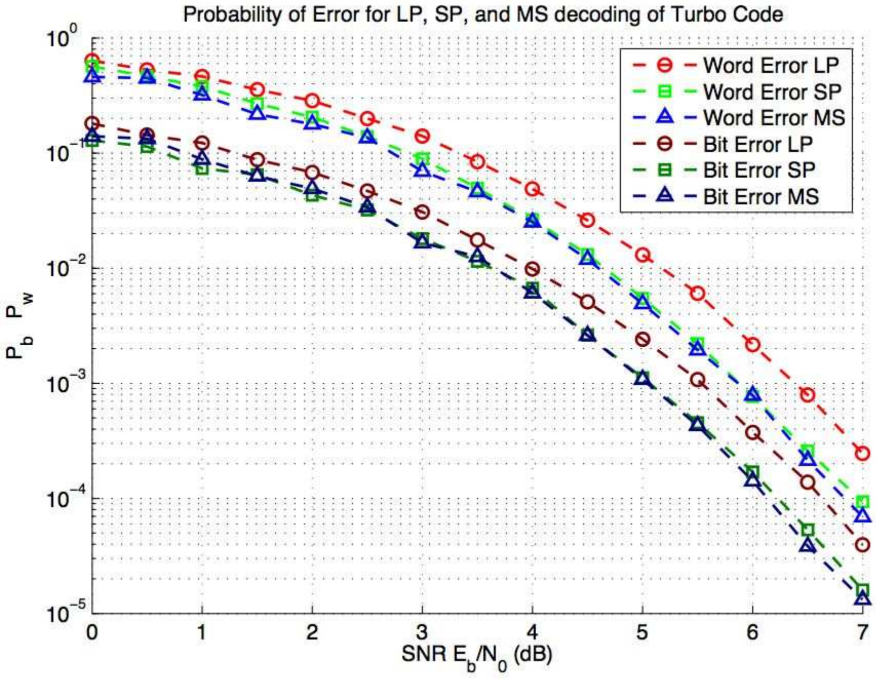

Figure 2: Performance of a turbo-code-based LDPC code with linear programming (LP), sum-product (SP) and min-sum (MS) decoding.

If iterative message-passing decoding and linear programming/graph cover decoding did, in fact, approximate each other, we would expect the behavior of these decoders to be inextricably linked; in particular, we would expect their relative performances to be consistent across simulations and LP/graph cover decoding would always outperform MS (since it does in many previous simulations). Since this is not the case in the simulation represented in Figure 2, we are led to believe that LP/graph cover decoding is not the correct model for iterative message-passing decoders.

5.1 Computation Trees and Graph Covers

The conflict observed above — that iterative message-passing decoding and graph cover decoding do not appear to approximate each other — is resolved by returning to the fundamental work of Wiberg [31]. Wiberg proves that iterative message-passing algorithms actually work by finding minimal cost configurations on computation trees. To make this more precise, we focus on the min-sum algorithm. In this case, we have:

Definition 5.1 (Wiberg [31], pp.16–17)**.** Let R be a computation tree for a Tanner graph T with variable nodes $x_1, \ldots, x_n$. Let $X(R)$ be the variable nodes of R, and let $\mathbf{c} = (c_x)_{x \in X(R)}$ be a configuration on R. For each $x \in X(R)$ that is a copy of x_i, 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 5.2 (Wiberg [31], Corollary 4.1)**.** *Let T be a Tanner graph with variable nodes $x_1, \ldots, x_n$. For each $i = 1, \ldots, n$, the min-sum algorithm finds, after m iterations, the value c_i at the root node of a lowest cost configuration on the computation tree of depth m rooted at x_i. The output of the algorithm is the vector $(c_1, \ldots, c_n) \in \{0, 1\}^n$.*

To better understand errors that arise in min-sum decoding, it is necessary to further examine computation tree pseudocodewords. One important observation to begin this analysis is that any configuration on a connected graph cover induces a configuration on every computation tree. This is shown in Section 6 below by examining the notion of truncated universal covers. Thus, every graph cover pseudocodeword that has a connected realization gives rise to at least one computation tree pseudocodeword, which may impact MS decoding.

It is not known, however, whether every computation tree configuration is induced by a graph cover configuration. Thus there may be computation tree pseudocodewords that cause errors in MS decoding and 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. Kelley and Sridhara [12] give a characterization of computation tree pseudocodewords that arise from graph cover pseudocodewords. They first define the notion of a *consistent* valid binary configuration on a computation tree for the Tanner graph T.

Definition 5.3 (Kelley and Sridhara [12], p. 4014)**.** Let T be a Tanner graph and let R be a computation tree of T that contains at least one copy of each check node of T and let $\mathbf{c}$ be a configuration on R. For each variable node x of T and for each check node f adjacent to x, the *local assignment of x at f by the configuration* $\mathbf{c}$, written $L_{\mathbf{c}}(x, f)$, is 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 variable node x of T, the value of $L_{\mathbf{c}}(x, f)$ is independent of the choice of check node f adjacent to x.

Remark 5.4. *Kelley and Sridhara [12] do not specify that the computation tree R must contain at least one copy of each check node of the Tanner graph T. However, as we will see below, the point of defining consistency is that a consistent configuration yields a vector of length n that is a point in the fundamental polytope, i.e., a normalized graph cover pseudocodeword. If a check is not represented in the computation tree, we can find otherwise consistent configurations whose corresponding vectors of length n are not elements of the* *fundamental polytope. An example of this is given as follows.*

Let T be the Tanner graph shown in Figure 3 and consider the configuration shown in Figure 4 on the computation tree for T rooted at x_1 of depth 1.

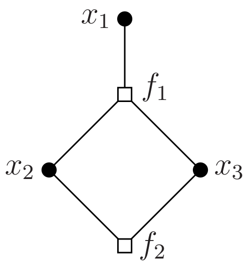

Figure 3: The Tanner graph T.

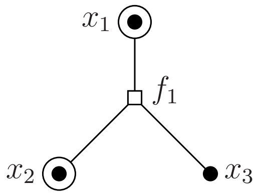

Figure 4: A configuration on a computation tree for T. The circled nodes have a binary value of 1, and the other node has a binary value of 0.

It is clear that, modulo the requirement that all check nodes of T are represented, that the configuration in Figure 4 is consistent. The vector of length 3 associated to this vector is $(1, 1, 0)$*. This vector is not in the fundamental polytope associated to T since it does not satisfy the constraint given by f_2 that states that $x_2 = x_3$.*

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 $x_1, \ldots, x_n$. A consistent configuration $\mathbf{c}$

on R gives rise to a vector $\boldsymbol{\omega} \in [0,1]^n$, 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 $\mathbf{c}$ [12]. The consistency of $\mathbf{c}$ also gives us that $\boldsymbol{\omega}$ 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 adjacent to x. As the fundamental polytope is the intersection of the collections of vectors satisfying each of the local check constraints, we see that $\boldsymbol{\omega}$ is a point in the fundamental polytope, and hence is realizable on a finite cover of T [12]. An example of a valid configuration on a computation tree that is not consistent is shown in Example 5.5 below.

Example 5.5 (Compare to Kelley and Sridhara [12], pp. 4017-4018.)**.** It can be shown that there are no nontrivial computation tree pseudocodewords for the Tanner graph T of Example 2.7. However, the addition of a new, redundant check allows for both nontrivial computation tree pseudocodewords as well as connected realizations of graph cover pseudocodewords that are not vertices of the fundamental polytope. Let T_1 be the Tanner graph of Figure 5. Then the code determined by T_1 is again the $[4,1,4]$ repetition code, but Figure 6 shows a nontrivial computation tree pseudocodeword for T_1.

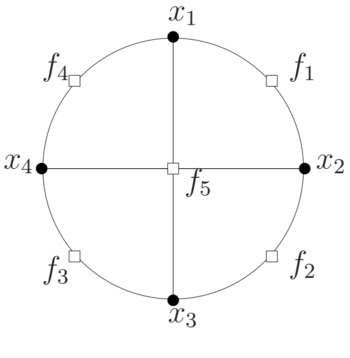

Figure 5: The Tanner graph T_1 of Example 5.5.

Table 2 gives values of $L_{\mathbf{c}}(x_i, f_j)$ for $1 \leq i \leq 4$ and $1 \leq j \leq 5$. If the variable node x_i is not adjacent to the check node f_j, no value of $L_{\mathbf{c}}(x_i, f_j)$ is given. Since there is at least one column in this table that contains differing values, the configuration given in Figure 6 is not

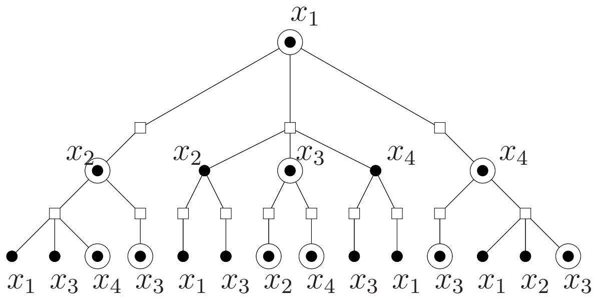
x_1
x_2
x_2
x_3
x_4
x_4
x_1 x_3 x_4 x_3 x_1 x_3 x_2 x_4 x_3 x_1 x_3 x_1 x_2 x_3

Figure 6: A computation tree of depth 2 rooted at x_1 for the Tanner graph T_1 in Figure 5. 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.

consistent.

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

Table 2: Values of $L_{\mathbf{c}}(x_i, f_j)$, with $\mathbf{c}$ as given in Figure 6.

Since the configuration in Figure 6 is not consistent, Kelley and Sridhara [12] point out that there is no meaningful vector of length four we may associate to it and, hence, it cannot correspond directly to a graph cover pseudocodeword. In a different sense, however, the configuration in Figure 6 can be considered as being induced by a graph cover pseudocodeword. More specifically, the Tanner graph in Figure 7 is a 4-cover of the Tanner graph of Figure 5, so the configuration in Figure 7 is a realization of a graph cover pseudocodeword. By rooting a computation tree at the top-left variable node in Figure 7, one can derive the inconsistent computation tree configuration of Figure 6 from the configuration given in Figure 7. Thus, the computation tree pseudocodeword of Figure 6 is, in this sense, induced by a graph cover pseudocodeword.

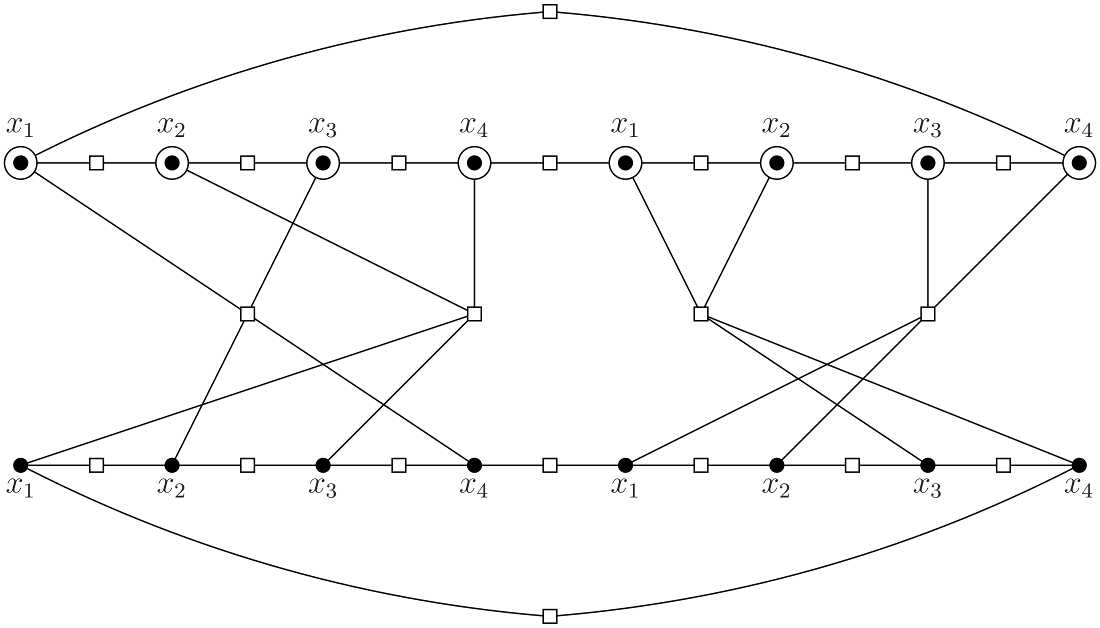

Figure 7: A configuration on a 4-cover of the Tanner graph T given in Figure 5. Circled nodes have a binary value of 1, and other nodes have a binary value of 0.

We see from the preceding example that the criterion of consistency does not lead to a complete characterization of the distinction between computation tree pseudocodewords and graph cover pseudocodewords. It is clear that in order to study the relationship between LP/graph cover decoding and MS decoding, 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. Many questions remain unanswered about the behavior of iterative message-passing decoders in general; hence, in order to simplify the analysis, it is natural to study the behavior in a restricted environment. For example, if only computation tree configurations induced by graph cover pseudocodewords were allowed, how would the performance of MS compare to that of LP/graph cover decoding? In Section 6, questions such as these are addressed in the context of universal covers.

5.2 Average Min-Sum Decoding

One characteristic of min-sum decoding that makes it particularly difficult to analyze is its perpetually changing output across iterations. The observed behavior of MS in computer simulations can be characterized in two ways: either the output stabilizes at a codeword or the output oscillates between a set of non-codeword/codeword outputs. By stabilizing at a codeword, we mean that for a sufficiently large number of iterations the output vector of min-sum is consistently the same codeword. The other behavior mentioned, oscillation between a set of codeword or non-codeword outputs, is best illustrated by an example.[1]

Example 5.6. Decoding of a single received vector was performed using 800 iterations of min-sum on the Tanner graph given in Figure 8. When the all-zeros codeword was sent (modulated as the vector $(-1,-1,-1,-1,-1,-1,-1)$) over an additive white Gaussian noise (AWGN) channel with SNR 0.0 dB, the channel output was

$$(-1.1504, -1.8782, -1.1789, 2.6625, -2.3933, 1.2978, 0.4123).$$

For sufficiently large iterations, the output of the min-sum decoder cycled through six vectors in $\mathbb{F}_2^7$, of which only one was a codeword. For example, the outputs after iterations 783–800 were as follows:

[1] Much of the material in this section has been previously published in [2].

$$\begin{array}{ccc}
(1,1,1,1,1,1,0) & (1,1,1,1,1,1,0) & (1,1,1,1,1,1,0) \\
(1,1,1,1,1,0,0) & (1,1,1,1,1,0,0) & (1,1,1,1,1,0,0) \\
(0,0,0,1,0,0,0) & (0,0,0,1,0,0,0) & (0,0,0,1,0,0,0) \\
(0,0,0,1,0,0,1) & (0,0,0,1,0,0,1) & (0,0,0,1,0,0,1) \\
(0,0,0,1,0,1,1) & (0,0,0,1,0,1,1) & (0,0,0,1,0,1,1) \\
(1,1,1,0,1,1,1) & (1,1,1,0,1,1,1) & (1,1,1,0,1,1,1)
\end{array}$$

Here, the first column gives the outputs after iterations 783–788, the second column corresponds to iterations 789–794, and the third column represents iterations 795–800. Notice the pronounced pattern that these output vectors follow.

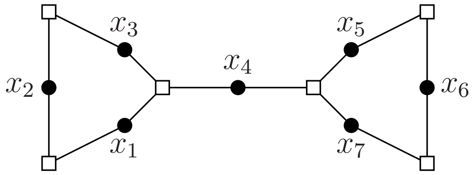

Figure 8: The Tanner graph of Example 5.6.

Example 5.6 presents a situation where the min-sum algorithm repeatedly cycles through a set of outputs that includes both codewords and non-codewords, even after more than 700 iterations have been performed. If the min-sum algorithm were stopped in this example at iterations 788, 794, or 800, the decoder would output the codeword $(1, 1, 1, 0, 1, 1, 1)$. Within the set of iterations shown above, however, the codeword $(1, 1, 1, 0, 1, 1, 1)$ only represents one sixth of all possible outputs of MS. In particular, for all non-codeword outputs, the binary value assigned to the fourth coordinate is 1. In an oscillatory case such as this, we see that the output of MS can vary drastically even between consecutive iterations. We propose the following decoding algorithm with the aim of capturing the overall behavior of MS, rather than simply a snapshot of a particular iteration.

Definition 5.7. The *average min-sum (AMS) decoder* is given by the following rule: After m iterations, the decoder outputs

$$\widehat{\mathbf{x}}^{\mathrm{AMS}} := \frac{1}{m}\sum_{i=1}^{m}\widehat{\mathbf{x}}^{(i)},$$

where $\widehat{\mathbf{x}}^{(i)}$ is the output of the min-sum decoder after i iterations.

Example 5.8. Again, consider the $[7, 2, 3]$ code of Example 5.6, defined by the Tanner graph of Figure 8. A simulation of 800 iterations of MS decoding on the AWGN channel with SNR of 0.0 dB was performed to obtain the AMS output. It was observed that over these iterations, MS always reached a steady oscillatory pattern, which resulted in an output of AMS that was a vector of "nice" rational numbers. In particular, four common non-codeword outputs were observed. These four outputs were extremely close to the following rational vectors:

$$\begin{aligned}\{&\left(\tfrac{1}{2}, \tfrac{1}{2}, \tfrac{1}{2}, 1, \tfrac{1}{2}, \tfrac{1}{2}, \tfrac{1}{2}\right), \left(\tfrac{1}{2}, \tfrac{1}{2}, \tfrac{1}{2}, \tfrac{5}{6}, \tfrac{1}{2}, \tfrac{1}{2}, \tfrac{1}{2}\right),\\ &\left(\tfrac{1}{2}, \tfrac{1}{2}, \tfrac{1}{2}, \tfrac{2}{3}, \tfrac{1}{2}, \tfrac{1}{2}, \tfrac{1}{2}\right), \left(\tfrac{1}{2}, \tfrac{1}{2}, \tfrac{1}{2}, \tfrac{1}{2}, \tfrac{1}{2}, \tfrac{1}{2}, \tfrac{1}{2}\right)\}\end{aligned}$$

Notice that the vector $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{5}{6}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2})$ is obtained, for example, in the situation of Example 5.6. Meanwhile, there is only one nontrivial LP pseudocodeword for this code, namely, $\left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, 1, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)$.

Because the observed behaviors of the min-sum decoder imply that the outputs always either stabilize at a codeword or eventually repeatedly cycle through some finite set of vectors, it is reasonable to believe that the output of the average min-sum decoder always approaches some limit, *i.e.*, that for any channel input, the limit

$$\lim_{m\to\infty}\widehat{\mathbf{x}}^{\mathrm{AMS}} = \lim_{m\to\infty}\frac{1}{m}\sum_{i=1}^{m}\widehat{\mathbf{x}}^{(i)} \tag{5.1}$$

always exists. This motivates the next definition.

Definition 5.9. An *average min-sum pseudocodeword* is a limiting value of the output vectors of the average min-sum decoding algorithm. A *nontrivial average min-sum pseudocodeword* is an average min-sum pseudocodeword that is not a codeword.

If the limit of (5.1) exists, then we have

$$\begin{aligned}\lim_{m\to\infty} \widehat{\mathbf{x}}^{\mathrm{AMS}} &= \lim_{m\to\infty} \frac{1}{m}\sum_{i=1}^{m} \widehat{\mathbf{x}}^{(i)} \\ &= \lim_{m\to\infty} \frac{1}{(m-(\ell-1))}\sum_{i=\ell}^{m} \widehat{\mathbf{x}}^{(i)}\end{aligned}$$

for any $\ell \in \mathbb{N}$. Since the output of MS for the first several iterations typically jumps around before the behavior described above manifests itself, using a larger value of ℓ may actually improve the rate of convergence in the above limits. Because of this, the implementation of AMS used below computes

$$\frac{1}{(m-599)}\sum_{i=600}^{m} \widehat{\mathbf{x}}^{(i)},$$

where m is chosen uniformly at random from the integers $\{800, 801, \ldots, 900\}$. We note that applying this implementation to Example 5.8 results in the same performance and set of observed outputs.

5.2.1 Average Min-Sum Simulation Results

Simulation results are presented here on a variety of codes in order to explore the performance of the average min-sum decoder. The focus is on the relationship between MS and AMS performance as well as a comparison of these two decoders with LP/graph cover decoding. In particular, when practical, the set of nontrivial AMS pseudocodewords is examined in hopes of elucidating the oscillatory nature of MS across iterations. Additionally, examination of AMS pseudocodewords may further explain a link, or perhaps a disparity, between MS and LP/graph cover decoding.

Let H_1 be the parity-check matrix

$$H_1 = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

and let C_1 be the code determined by H_1. Then

$$C_1 = \begin{array}{c} \{(0,0,0,0,0,0),(0,0,0,0,1,1),(0,1,1,1,1,0),(1,1,0,0,0,0), \\ (1,1,0,0,1,1),(1,0,1,1,0,1),(1,0,1,1,1,0),(0,1,1,1,0,1)\} \end{array}$$

has length 6 and dimension 3. In simulations, we observed very few distinct AMS pseudocodewords, and all of them were approximations of simple rational vectors in the fundamental polytope. While the only nontrivial LP pseudocodewords determined by H_1 are $\left(\frac{1}{2},\frac{1}{2},1,0,\frac{1}{2},\frac{1}{2}\right)$ and $\left(\frac{1}{2},\frac{1}{2},0,1,\frac{1}{2},\frac{1}{2}\right)$, the following AMS pseudocodewords were the ones commonly seen in simulations:

$$\begin{array}{c} \left(\frac{1}{2},\frac{1}{2},1,0,\frac{1}{2},\frac{1}{2}\right), \left(\frac{1}{2},\frac{1}{2},0,1,\frac{1}{2},\frac{1}{2}\right), \left(\frac{1}{2},\frac{1}{2},\frac{1}{4},0,\frac{1}{2},\frac{1}{2}\right), \\ \left(\frac{1}{2},\frac{1}{2},0,\frac{1}{4},\frac{1}{2},\frac{1}{2}\right), \left(\frac{1}{2},\frac{1}{2},\frac{3}{4},0,\frac{1}{2},\frac{1}{2}\right), \left(\frac{1}{2},\frac{1}{2},0,\frac{3}{4},\frac{1}{2},\frac{1}{2}\right). \end{array}$$

Moreover, AMS, MS and LP perform identically in almost all simulations on this code with respect to both word and bit error rate, and are very close to ML. When AMS output a codeword, it always equaled the LP output and, as a result of the ML certificate property of LP decoding, was the ML codeword.

Now consider H_2 the semi-regular parity-check matrix

$$H_2 = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

and let

$$C_2 = \{(0,0,0,0,0,0,0), (0,0,0,0,1,1,1), (1,1,1,0,0,0,0), (1,1,1,0,1,1,1)\}$$

be the code of length 7 and dimension 2 determined by H_2; this is the same code considered in Examples 5.6 and 5.8 above. Simulations of this code result in behavior similar to that of the code C_1 above, in that AMS, MS and LP perform identically and are close to ML, and there was a small set of AMS pseudocodewords. The sets of LP and common AMS pseudocodewords are discussed in Example 5.8. The performance results from the previous two codes agree with Feldman's comment [7] that the performance of LP and MS agree when the parity-check matrix has a constant column weight of two.

Consider next the $(3,3)$-regular low-density parity-check code C_3 of length 6 and dimension 3 determined by the parity-check matrix

$$H_3 = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}.$$

Note that, as subspaces of $\mathbb{F}_2^6$, we have $C_3 = C_1$. Min-sum and AMS perform similarly in simulations, as do LP and ML, though LP and ML perform much better than either MS or AMS. The following set of commonly witnessed AMS pseudocodewords is still simple, in that it is small and consists of vectors in the fundamental polytope that appear to be approximations of fractions with small denominators:

$$\left\{ \left(\tfrac{1}{2}, \tfrac{1}{2}, \tfrac{1}{2}, \tfrac{1}{2}, \tfrac{1}{2}, \tfrac{1}{2}\right), \left(\tfrac{1}{2}, \tfrac{1}{2}, 0, 0, 0, 0\right), \left(0, 0, 0, 0, \tfrac{1}{2}, \tfrac{1}{2}\right), \right.$$
$$\left. \left(\tfrac{1}{2}, \tfrac{1}{2}, 0, 0, 1, 1\right), \left(1, 1, 0, 0, \tfrac{1}{2}, \tfrac{1}{2}\right) \right\}.$$

In contrast to the case of the code determined by H_1, where two of the six nontrivial AMS pseudocodewords appearing in simulations were nontrivial LP pseudocodewords, in this case the nontrivial LP pseudocodewords $\left(\tfrac{1}{2}, \tfrac{1}{2}, 1, 0, \tfrac{1}{2}, \tfrac{1}{2}\right)$ and $\left(\tfrac{1}{2}, \tfrac{1}{2}, 0, 1, \tfrac{1}{2}, \tfrac{1}{2}\right)$ are not present in the set of AMS pseudocodewords for H_3.

As with the parity-check matrices of column weight two above, the regularity condition may be loosened to semi-regularity so as to further examine the impact of regularity on the behavior of the average min-sum decoder. The matrix

$$H_4 = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

defines a semi-regular LDPC code C_4 of length 6 and dimension 3 with constant column weight 3. Notice that, as subsets of $\mathbb{F}_2^6$, we have $C_4 = C_3 = C_1$. Min-sum and AMS again performed similarly in simulations. The performances of LP and ML are still close, but now they are only slightly better than MS and AMS. With this description of the code, however, the set of AMS pseudocodewords is now large and unwieldy; even at high SNRs the non-codeword outputs of AMS no longer appear to be simple rational vectors. Four such AMS outputs obtained at 5.0 dB are

$$\begin{aligned} &(0.50, 0.50, 0.27, 0.47, 0.28, 0.33)\,, \\ &(0.14, 0.14, 0.14, 0.17, 0.81, 0.80)\,, \\ &(0.45, 0.33, 0.41, 0.49, 0.50, 0.50)\,, \qquad \text{and} \\ &(0.11, 0.11, 0.04, 0.07, 0.02, 0.08)\,. \end{aligned}$$

Additionally, enforcing regularity at the check nodes, *i.e.*, maintaining constant row weight, does not appear to simplify the set of AMS pseudocodewords any more than enforcing regularity at the variable nodes, as seen in the next example. Let H_5 be the parity-check

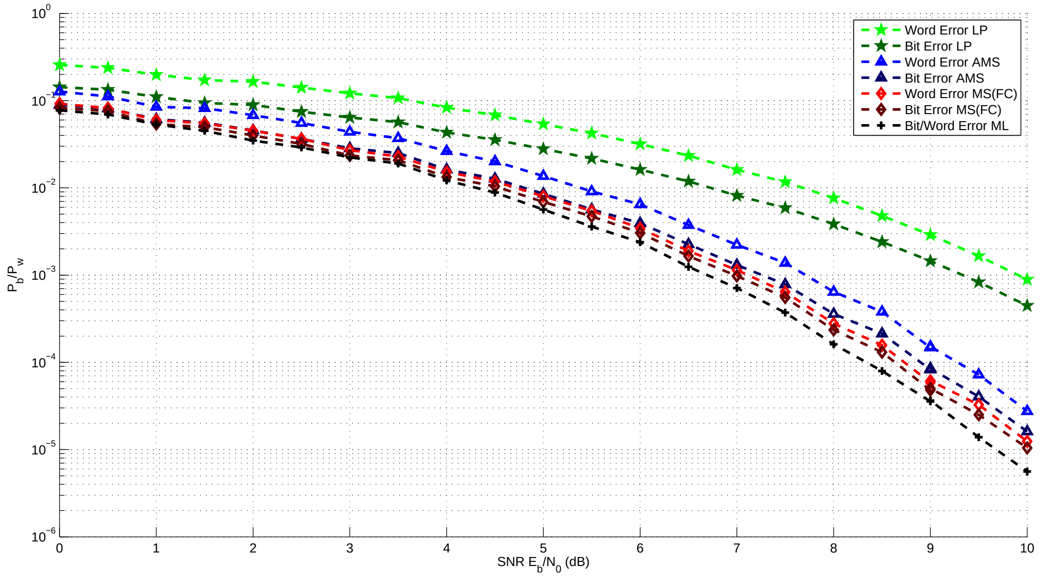

Figure 9: Performance of the irregular LDPC code C_6 with maximum likelihood (ML), linear programming (LP), min-sum (MS) and average min-sum (AMS) decoding.

matrix

$$H_5 = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

and let C_5 be the code determined by H_5. Then C_5 is a code of length 6 and dimension 3 and H_5 has a constant row weight of three. As subspaces of $\mathbb{F}_2^6$, we have $C_5 = C_4 = C_3 = C_1$. The relative performance is similar to that of C_4: MS and AMS are close, with LP and ML performing similarly and only slightly better than MS and AMS. As was the case with C_4, however, we also observe very little discernible structure in the large set of nontrivial AMS pseudocodewords.

The final example that we consider for which it is practical to examine the set of average min-sum pseudocodewords is an irregular low-density parity-check code C_6 of length 10 and dimension 1 defined by the parity-check matrix

$$H_6 = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

The performance of this code under various decoding algorithms is shown in Figure 9. In this simulation, MS is implemented such that it terminates when it reaches its first codeword. The performance of both AMS and MS are very close to ML and far better than that of LP. The nontrivial AMS pseudocodewords are extremely irregular and again it is hard to discern any structure.

Finally, we offer in Figures 10 and 11 results of two simulations of the average min-sum decoder on larger regular codes that are more similar to codes actually used in practice. In these simulations, it can be seen that the performance of AMS is the same as that of MS with respect to word error rate, but better than that of MS with respect to bit error rate. Unfortunately, it is impractical to examine the set of AMS pseudocodewords or to compare this performance to that of LP or ML for codes with such large block lengths.

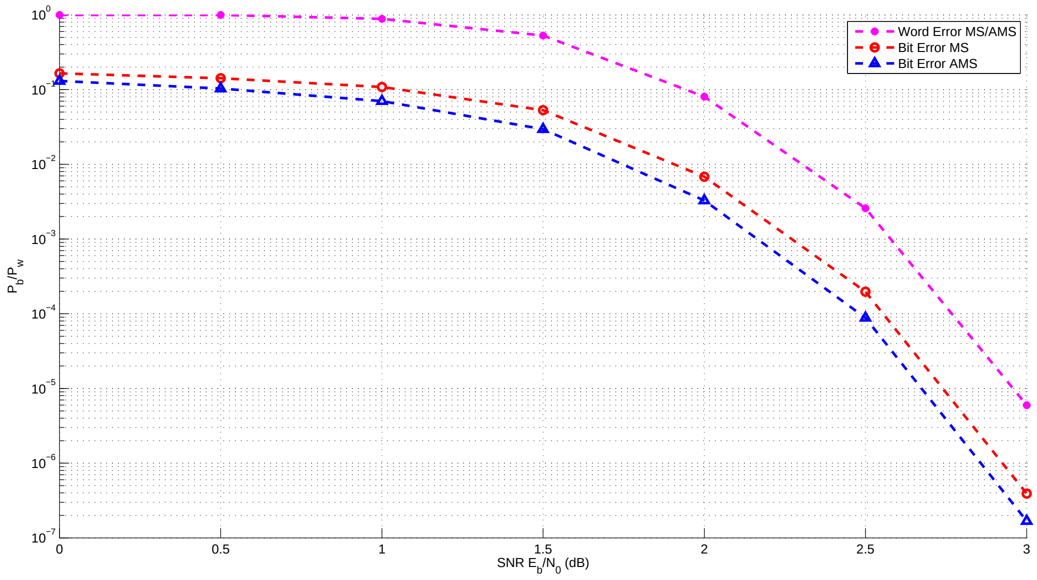

Figure 10: Performance of a rate 1/2, (6,3)-regular LDPC code of length 1080 with min-sum (MS) and average min-sum (AMS) decoding.

5.2.2 Discussion of Average Min-Sum Results

We have observed through simulation of the small parity-check matrices H_1, H_2, H_3, H_4 and H_5 that min-sum and average min-sum have similar performance with respect to both bit and word error rate. Furthermore, on the large codes of lengths 1,080 and 131,072, MS and AMS again have comparable word error rates, but AMS has a significantly better bit error rate than MS; see Figures 10 and 11. The question of whether AMS typically outperforms MS with respect to bit error rate for codes with reasonable parameters is an object of future investigation.

Also of interest is the set of nontrivial average min-sum pseudocodewords. It was observed that for codes defined by parity-check matrices of column weight of two or of uniform row and column weight, the set of nontrivial AMS pseudocodewords is a small set of vectors resembling "nice" rational vectors that lie within the fundamental polytope. In the cases

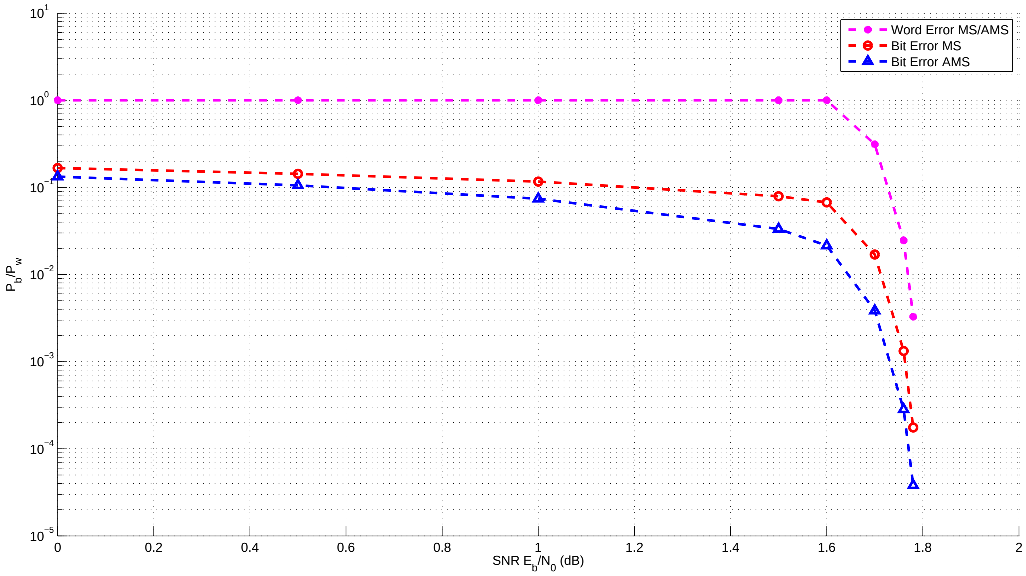

Figure 11: Performance of a rate 1/2, (6,3)-regular LDPC code of length $131,072$ with min-sum (MS) and average min-sum (AMS) decoding.

where the parity-check matrix is irregular with at least one column having weight different from two, the set of nontrivial AMS pseudocodewords was extremely large and it was difficult to find any apparent structure in the vectors. For parity-check matrices with a uniform column weight two, perhaps the phenomenon of "nice" rational vectors can be explained by Feldman's comment that MS and LP have identical performance on *cycle codes* [7], *i.e.*, codes determined by parity-check matrices of constant column weight of 2. As for the regular LDPC code defined by H_3, possible connections between the distribution of copies of variable nodes in computation trees and the outputs of MS remain under investigation.

In summary, the average min-sum decoder performs analogously in most cases with min-sum, and in the simulations on codes most similar to those used in practice, AMS displayed a significant improvement in bit error rate over MS. Thus, AMS is interesting in its own right as well as being an instrument with which to study the long-term behavior of MS.

Additionally, in the simulations performed with regular LPDC codes and cycle codes, the set of nontrivial AMS pseudocodewords approximated rational points in the fundamental polytope. Moreover, some of these nontrivial AMS pseudocodewords were not vertices of the fundamental polytope, and hence are not possible outputs of (the standard implementations of) LP/graph cover decoding. These results suggest that the study of AMS decoding may shed light on relationships that exist between MS and LP/graph cover decoding.

6 Limiting Behavior of Min-Sum

As discussed previously, a primary goal of this work is to gain a more complete understanding of the behavior of the min-sum decoding algorithm. In this section[2], we examine how the algorithm behaves as the number of iterations goes to infinity. Since, as Wiberg [31] shows, the min-sum algorithm for m iterations operates on the computation tree of depth m derived from the Tanner graph, we are led to consider computation trees of infinite depth. As we will see, such infinite trees are actually *universal covers* of the Tanner graph and, as such, are well-known topological objects. Since the Tanner graphs we consider are finite and connected, we deal exclusively with universal covers of finite connected graphs. For universal covers of more general topological spaces, see [20].

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 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.

If G is a finite connected graph, then a universal cover of G exists, is a tree, and is unique up to graph isomorphism [22], so we will refer to *the* universal cover of G. Note that if G is a tree, then G is its own universal cover. When G is not a tree, a practical way of constructing

[2] Many of the results of this section and the next were previously published in [4]; see also [3].

the universal cover of G is to build the computation tree of infinite depth rooted at a vertex of G. It is clear that this infinite tree is a cover; the covering map $\widehat{\pi}$ is found by using the labels typically included on the computation tree. That it is the universal cover follows from Theorem 1.24 in [22], noting that trees have trivial fundamental group.

6.1 Configurations and Cost on Universal Covers

Given a Tanner graph T, the universal cover of T can be thought of as an infinite Tanner graph in the following sense.

Definition 6.2 ([4], Definition 6.2)**.** Let $T = (X \cup F, E)$ be a Tanner graph, and let $\widehat{\pi} : \widehat{T} \to T$ be the universal cover of T. Set

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

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 $\widehat{T}$ such 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 $\widehat{T}$.

Proposition 6.3 outlines some relationships between universal cover pseudocodewords, graph cover pseudocodewords and computation tree pseudocodewords.

Proposition 6.3 ([4], 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 that has a connected graph cover realization induces a universal cover pseudocodeword.*

3. *Every universal cover pseudocodeword truncates to a computation tree pseudocodeword on every computation tree for T.*

Proof. Let $\widehat{\pi} : \widehat{T} \to T$ be the universal cover of T and let $\widehat{X}$ be the set of variable nodes of $\widehat{T}$.

1. Let $\mathbf{c} = (c_x)_{x \in X(R)}$ be a computation tree pseudocodeword on some computation tree R of T, rooted at the variable node v of T. 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. Let $\mathbf{p}$ be 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 $\widehat{c}_{\widehat{x}} = \widetilde{c}_{\widetilde{\pi}(\widehat{x})}$ for each $\widehat{x} \in \widehat{X}$.

3. Let $\widehat{\mathbf{c}}$ be a universal cover pseudocodeword and let R be 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, so that R is a subtree of T. The truncation of $\widehat{\mathbf{c}}$ to R yields a computation tree pseudocodeword on R.

■

The next step is to apply the notion of the universal cover towards the analysis of other decoders. We first define a cost function on the set of universal cover pseudocodewords. We would like the cost function to resemble the limit of the cost given in Definition 5.1, which is Wiberg's cost function for finite computation trees [31]. Since the number of variable nodes on a computation tree increases exponentially with the number of iterations, we cannot simply take the limit of Wiberg's cost function. These costs are therefore normalized by dividing through by the average number of copies of a given variable node in the computation tree before a limiting process is applied.

Definition 6.4 (See also [4], Definition 6.4)**.** Let $T = (X \cup F, E)$ 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 $\mathbf{c}$ on R, let $G(\mathbf{c})$ be the cost of $\mathbf{c}$, as given in Definition 5.1. The *normalized cost* of $\mathbf{c}$ on R is

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

Definition 6.5 (See also [4], Definition 6.5)**.** Let $T = (X \cup F, E)$ be a Tanner graph and let $\widehat{T}_v$ be the universal cover of T, realized as an infinite computation tree rooted at the variable node v of T. 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 by truncating $\widehat{T}_v$ after the $2m^{\text{th}}$ level. For any configuration $\widehat{\mathbf{c}}_v$ on $\widehat{T}_v$, let $\widehat{\mathbf{c}}_v^{(m)}$, $m \geq 1$, be the truncation of $\widehat{\mathbf{c}}_v$ to $R_v^{(m)}$, and let $\overline{G}_v^{(m)}(\widehat{\mathbf{c}}_v^{(m)})$ be the normalized cost of $\widehat{\mathbf{c}}_v^{(m)}$ on $R_v^{(m)}$. The *rooted cost* of the universal cover configuration $\widehat{\mathbf{c}}_v$ on the infinite computation tree $\widehat{T}_v$ is defined to be

$$G_v(\widehat{\mathbf{c}}_v) := \limsup_{m\to\infty} \overline{G}_v^{(m)}(\widehat{\mathbf{c}}_v^{(m)}) = \limsup_{m\to\infty} \frac{|X|}{\left|X_v^{(m)}\right|} G(\widehat{\mathbf{c}}_v^{(m)}),$$

where $X_v^{(m)}$ is the set of variable nodes of $R_v^{(m)}$.

Notice that the normalized cost of the configuration $\widehat{\mathbf{c}}_v$ always satisfies $|\overline{G}_v^{(m)}(\widehat{\mathbf{c}}_v^{(m)})| \leq |X| \sum_{i=1}^{n} |\lambda_i|$, where $(\lambda_1, \ldots, \lambda_n)$ is the cost vector associated with the received vector, and, moreover, that this bound is independent of $\widehat{\mathbf{c}}_v$, m and v. As long as each λ_i is finite, we therefore have that $G_v(\widehat{\mathbf{c}}_v)$ exists and is finite for any configuration $\widehat{\mathbf{c}}_v$ on the universal cover $\widehat{T}_v$ of T.

We further investigate the notion of rooted cost to provide the missing link between graph cover decoding and min-sum decoding. Toward this goal, we must first introduce the notion of a *non-backtracking random walk*, which is described for an arbitrary graph G as follows: select a vertex v_0 of G from which to begin the walk, and select uniformly at random an edge e_1 incident to v_0. Let v_1 be the other endpoint of e_1 and select uniformly at random an edge $e_2 \neq e_1$ incident to v_1. Repeat this process some predetermined finite number of times.

More formally (and restricting to walks of even length for reasons that will become clear shortly), let $W_v^{(m)}$ be the set of all non-backtracking walks of length $2m$ in G whose initial vertex is v, and define the probability of the walk $w \in W_v^{(m)}$ with vertices $v = v_0, v_1, \ldots, v_{2m}$ to be

$$P(w) := \frac{1}{\deg(v)(\deg(v_1) - 1)\ldots(\deg(v_{2m-1}) - 1)}.$$

A *non-backtracking random walk* of length $2m$ with initial vertex v is then a pair $(w, P(w))$ where $w \in W_v^{(m)}$ and $P(w)$ is its probability. One can see that this is a probability measure on $W_v^{(m)}$ by using induction on m, and it is clear that this measure agrees with the intuitive description in the previous paragraph.

In our situation, the graph G is a Tanner graph T, and the walks of interest to us must start and end at variable nodes. Since any such walk must have even length because T is bipartite, we focus exclusively on walks of even length.

Let $q_v^{(m)}(x) = \sum_{w \in W} P(w)$, where $W = W_v^{(m)}(x)$ is the set of non-backtracking walks in T of length $2m$ that start at variable node v and end at variable node x. With this definition, we see that $q_v^{(m)}(x)$ is the probability that a non-backtracking random walk in T of length $2m$ that starts at vertex v will have terminal vertex x [21]. Theorem 6.6 and Proposition 6.7 below are used to show that for a subset of regular LDPC codes, the rooted cost of universal cover pseudocodewords induced by connected graph cover pseudocodewords can be computed using a limit rather than a limit superior. Moreover, an exact formula for the rooted cost of such a configuration is given.

Theorem 6.6 (See Ortner and Woess [21], Theorem 1.2(ii))**.** *Let $T = (X \cup F, E)$ be a*

Tanner graph with minimum degree at least 3. Using the notation established above, we have

$$\lim_{m\to\infty} q_v^{(m)}(x) = \frac{\deg(x)}{|E|}$$

for all $v, x \in X$.

Recall from Definition 6.5 that $X_v^{(m)}$ is the set of variable nodes in the computation tree $R_v^{(m)}$ of depth m rooted at the variable node v of T, and write $X_v^{(m)}(x)$ for the set of copies of the variable node x of T in $X_v^{(m)}$. The next theorem shows that the distribution of variable nodes in the computation tree becomes uniform as the depth of the tree goes to infinity.

Theorem 6.7. *Let $T = (X \cup F, E)$ be the Tanner graph of a (d_X, d_F)-regular LDPC code with $d_X, d_F \geq 3$. For any positive integer m, let $R_v^{(m)}$ be the computation tree of depth m rooted at the variable node v of T. For any $x \in X$, we have*

$$\lim_{m\to\infty} \frac{\left|X_v^{(m)}(x)\right|}{\left|X_v^{(m)}\right|} = \frac{1}{|X|}.$$

Proof. For $m \geq 1$, the biregularity of T forces the number of non-backtracking walks in T of length $2m$ that start at any given variable node to be $\tau^{(m)} := \frac{d_X}{d_X - 1}(d_X - 1)^m (d_F - 1)^m$, with each walk equally probable. Let $\widehat{T}_v$ be the universal cover of T, realized as an infinite computation tree rooted at v, so that $R_v^{(m)}$ is the truncation of $\widehat{T}_v$ to depth m. Let $\eta_v^{(m)}(x)$ be the number of copies of variable node x in the $2m^{\text{th}}$ level of $\widehat{T}_v$. There is a natural bijection between non-backtracking walks in T that start at v and paths in $\widehat{T}_v$ that start at the root node; thus, $\eta_v^{(m)}(x)$ is precisely the number of non-backtracking walks in T of length $2m$ that start at v and end at x. Therefore $q_v^{(m)}(x) = \frac{\eta_v^{(m)}(x)}{\tau^{(m)}}$.

Let $p_v^{(m)}(k)$ be the probability of picking uniformly at random a variable node in the $2k^{\text{th}}$ level from all variable nodes in $R_v^{(m)}$. Since the probability of selecting uniformly at random a copy of variable node x from all variable nodes of $R_v^{(m)}$ is $\frac{\left|X_v^{(m)}(x)\right|}{\left|X_v^{(m)}\right|}$, we have

$$\frac{\left|X_v^{(m)}(x)\right|}{\left|X_v^{(m)}\right|} = q_v^{(0)}(x)p_v^{(m)}(0) + q_v^{(1)}(x)p_v^{(m)}(1) + \cdots + q_v^{(m)}(x)p_v^{(m)}(m).$$

Let $\epsilon > 0$ be given and set $L = \frac{\deg(x)}{|E|}$. By Theorem 6.6, there is a positive integer M_1 such that $|q_v^{(m)}(x) - L| < \frac{\epsilon}{3}$ for all $m \geq M_1$. Since the number of variable nodes from one level to the next in the computation tree grows exponentially by a factor of $(d_X - 1)(d_F - 1) \geq 4$, the probability of selecting a variable node from the first $2M_1$ levels of a computation tree diminishes to zero as the depth of the tree increases. Thus, we can find $M_2 > M_1$ such that for all $m \geq M_2$,

$$\sum_{i=0}^{M_1} p_v^{(m)}(i) < \frac{\epsilon}{3}.$$

Then for all $m \geq M_2$,

$$\begin{aligned}
\left|\frac{\left|X_v^{(m)}(x)\right|}{\left|X_v^{(m)}\right|} - L\right| &= \left|q_v^{(0)}(x)p_v^{(m)}(0) + \cdots + q_v^{(M_1)}(x)p_v^{(m)}(M_1) + \cdots + q_v^{(m)}(x)p_v^{(m)}(m) - L\right| \\
&\leq \sum_{i=0}^{M_1} |q_v^{(i)}(x)p_v^{(m)}(i)| + \Bigl|q_v^{(M_1+1)}(x)p_v^{(m)}(M_1+1) + \ldots \\
&\qquad \cdots + q_v^{(m)}(x)p_v^{(m)}(m) - \sum_{j=0}^{M_1} p_v^{(m)}(j)L - \sum_{k=M_1+1}^{m} p_v^{(m)}(k)L\Bigr| \\
&\leq \frac{\epsilon}{3} \quad + \quad \sum_{j=0}^{M_1} p_v^{(m)}(j)L \quad + \quad \sum_{k=M_1+1}^{m} p_v^{(m)}(k)\left|q_v^{(k)}(x) - L\right| \\
&\leq \frac{\epsilon}{3} \quad + \quad \frac{\epsilon}{3}L \quad + \quad \sum_{k=M_1+1}^{m} p_v^{(m)}(k)\frac{\epsilon}{3} \\
&\leq \frac{\epsilon}{3} + \frac{\epsilon}{3} + \frac{\epsilon}{3} = \epsilon.
\end{aligned}$$

Thus, we have

$$\lim_{m\to\infty} \frac{\left|X_v^{(m)}(x)\right|}{\left|X_v^{(m)}\right|} = \frac{\deg(x)}{|E|} = \frac{d_X}{|X|\, d_X} = \frac{1}{|X|}$$

as desired. ■

The next few results examine the behavior of the rooted cost function for certain special types of universal cover configurations. The exposition will be helped by the following definition, which is motivated by the definition of the *support* of a vector as the collection of coordinates in which the vector is nonzero.

Definition 6.8. Let T be a Tanner graph, let $\widehat{T}$ be the universal cover of T, and let $\widehat{X}$ be the set of variable nodes of $\widehat{T}$. The *support* of a configuration $\widehat{\mathbf{c}} = (\widehat{c}_{\widehat{x}})_{\widehat{x}\in\widehat{X}}$ on $\widehat{T}$ is

$$\operatorname{supp}(\widehat{\mathbf{c}}) := \{\widehat{x} \in \widehat{X} \mid \widehat{c}_{\widehat{x}} = 1\}.$$

Theorem 6.9 (see also [4], Theorem 6.6)**.** *Let $T = (X \cup F, E)$ be the Tanner graph of a (d_X, d_F)-regular LDPC code, with $d_X, d_F \geq 3$. Let $\widetilde{T}$ be a connected cover of T, let $\widetilde{\mathbf{c}}$ be a codeword in the code defined by $\widetilde{T}$, and let $\boldsymbol{\omega} = \boldsymbol{\omega}(\widetilde{\mathbf{c}})$ be the normalized pseudocodeword associated to $\widetilde{\mathbf{c}}$. Suppose that, on the universal cover $\widehat{T}_v$ of T realized as an infinite computation tree rooted at the variable node v of T, the configuration $\widehat{\mathbf{c}}_v$ is induced by $\widetilde{\mathbf{c}}$. Then*

$$G_v(\widehat{\mathbf{c}}_v) = \lim_{m\to\infty} \overline{G}_v^{(m)}(\widehat{\mathbf{c}}_v^{(m)}) = \boldsymbol{\lambda} \cdot \boldsymbol{\omega},$$

where $\boldsymbol{\lambda}$ is the vector of log-likelihood ratios.

Proof. Let M be the degree of the cover $\widetilde{T} = (\widetilde{X} \cup \widetilde{F}, \widetilde{E})$ of T. Note that $\widehat{T}_v$ is a universal cover of $\widetilde{T}$ and that $\widetilde{T}$ is finite and connected with $d_{\widetilde{X}} = d_X \geq 3$ and $d_{\widetilde{F}} = d_F \geq 3$, and let

$\widehat{\mathbf{c}}_v$ be a configuration on $\widehat{T}_v$ induced by $\widetilde{\mathbf{c}}$. Then we have

$$\begin{aligned}
G_v(\widehat{\mathbf{c}}_v) &= \limsup_{m\to\infty} \frac{|X|}{\left|X_v^{(m)}\right|} \sum_{\widetilde{x}\in\widetilde{X}} \lambda_{\widetilde{x}} \left|X_v^{(m)}(\widetilde{x}) \cap \operatorname{supp}(\widehat{\mathbf{c}}_v^{(m)})\right| \\
&= \limsup_{m\to\infty} \frac{|X|}{\left|X_v^{(m)}\right|} \sum_{\widetilde{x}\in\operatorname{supp}(\widetilde{\mathbf{c}})} \lambda_{\widetilde{x}} \left|X_v^{(m)}(\widetilde{x})\right| \\
&= |X| \sum_{\widetilde{x}\in\operatorname{supp}(\widetilde{\mathbf{c}})} \lambda_{\widetilde{x}} \limsup_{m\to\infty} \frac{\left|X_v^{(m)}(\widetilde{x})\right|}{\left|X_v^{(m)}\right|} \\
&= |X| \sum_{\widetilde{x}\in\operatorname{supp}(\widetilde{\mathbf{c}})} \lambda_{\widetilde{x}} \frac{1}{\left|\widetilde{X}\right|}
\end{aligned}$$

by Theorem 6.7. To continue the string of equalities, we use that $\left|\widetilde{X}\right| = M\,|X|$, $\lambda_{\widetilde{x}} = \lambda_x$ for each $\widetilde{x} \in \widetilde{X}$ in the inverse image of x under the covering map, and the number of such $\widetilde{x}$ in the support of $\widetilde{\mathbf{c}}$ is precisely $M\omega_x$, and so we have:

$$\begin{aligned}
&= |X| \sum_{x\in\operatorname{supp}(\boldsymbol{\omega})} M\omega_x\lambda_x \frac{1}{M\,|X|} \\
&= \sum_{x\in\operatorname{supp}(\boldsymbol{\omega})} \lambda_x\omega_x \\
&= \boldsymbol{\lambda}\cdot\boldsymbol{\omega},
\end{aligned}$$

as desired. ■

As a consequence of Theorems 6.9 and 6.7, we obtain the following result, which says that the distribution of 1's in a universal cover configuration induced by a codeword on a cover of the Tanner graph is given by the corresponding normalized graph cover pseudocodeword.

Proposition 6.10. *Let $T = (X \cup F, E)$ be the Tanner graph of a (d_X, d_F)-regular LDPC code with $d_X, d_F \geq 3$ and write $X = \{x_1, \ldots, x_n\}$. Let $\widetilde{\mathbf{c}}$ be a configuration on some finite* *connected cover of* T *and let* $\boldsymbol{\omega} = \boldsymbol{\omega}(\widetilde{\mathbf{c}}) = (\omega_1, \ldots, \omega_n)$ *be the normalized pseudocodeword associated to* $\widetilde{\mathbf{c}}$*. Let* $\widehat{\mathbf{c}}_v$ *be a configuration induced by* $\widetilde{\mathbf{c}}$ *on* $\widehat{T}_v$*, the universal cover of* T *realized as an infinite computation tree rooted at the variable node* v *of* T*. Then for* $1 \leq i \leq n$ *we have*

$$\lim_{m\to\infty} \frac{\left|X_v^{(m)}(x_i) \cap \operatorname{supp}(\widehat{\mathbf{c}}_v^{(m)})\right|}{\left|X_v^{(m)}(x_i)\right|} = \omega_i.$$

Proof. Let $\boldsymbol{\lambda}$ be the vector that has a value of $\frac{1}{n}$ in the i^{th} position and 0 elsewhere. Then, by using this $\boldsymbol{\lambda}$ as our log-likelihood vector in Theorem 6.9, we have

$$\begin{aligned}
\frac{1}{n}\omega_i = \boldsymbol{\lambda} \cdot \boldsymbol{\omega} &= \lim_{m\to\infty} \overline{G}_v^{(m)}(\widehat{\mathbf{c}}_v^{(m)}) \\
&= \lim_{m\to\infty} \frac{n}{\left|X_v^{(m)}\right|} G(\widehat{\mathbf{c}}_v^{(m)}) \\
&= \lim_{m\to\infty} \frac{n}{\left|X_v^{(m)}\right|} \frac{1}{n} \left|X_v^{(m)}(x_i) \cap \operatorname{supp}(\widehat{\mathbf{c}}_v^{(m)})\right| \\
&= \lim_{m\to\infty} \frac{\left|X_v^{(m)}(x_i) \cap \operatorname{supp}(\widehat{\mathbf{c}}_v^{(m)})\right|}{\left|X_v^{(m)}\right|}.
\end{aligned}$$

Hence, by Theorem 6.7, we have

$$\begin{aligned}
\lim_{m\to\infty} \frac{\left|X_v^{(m)}(x_i) \cap \operatorname{supp}(\widehat{\mathbf{c}}_v^{(m)})\right|}{\left|X_v^{(m)}(x_i)\right|} &= \lim_{m\to\infty} \left(\frac{\left|X_v^{(m)}(x_i) \cap \operatorname{supp}(\widehat{\mathbf{c}}_v^{(m)})\right|}{\left|X_v^{(m)}\right|}\right)\left(\frac{\left|X_v^{(m)}\right|}{\left|X_v^{(m)}(x_i)\right|}\right) \\
&= \left(\lim_{m\to\infty} \frac{\left|X_v^{(m)}(x_i) \cap \operatorname{supp}(\widehat{\mathbf{c}}_v^{(m)})\right|}{\left|X_v^{(m)}\right|}\right)\left(\frac{1}{\lim_{m\to\infty} \frac{\left|X_v^{(m)}(x_i)\right|}{\left|X_v^{(m)}\right|}}\right) \\
&= \left(\frac{1}{n}\omega_i\right)\left(\frac{1}{\frac{1}{n}}\right) \\
&= \omega_i
\end{aligned}$$

as desired. ■

In light of Proposition 6.10, Theorem 6.9 may now be interpreted as saying that, in the case of universal cover configurations induced by graph cover pseudocodewords, the rooted cost can be computed by simply taking the dot product of the vector of log-likelihoods with the vector representing the distribution of 1's in the configuration. Our next result is a generalization of this.

Proposition 6.11. *Let $T = (X \cup F, E)$ be the Tanner graph of a (d_X, d_F)-regular LDPC code with $d_X, d_F \geq 3$ and write $X = \{x_1, x_2, \ldots, x_n\}$. Let $\widehat{\mathbf{c}}_v$ be a configuration on $\widehat{T}_v$, the universal cover of T realized as an infinite computation tree rooted at the variable node v of T. Suppose the limit*

$$\xi_i := \lim_{m\to\infty} \frac{\left|X_v^{(m)}(x_i) \cap \operatorname{supp}(\widehat{\mathbf{c}}_v^{(m)})\right|}{\left|X_v^{(m)}(x_i)\right|}$$

exists for $1 \leq i \leq n$. Then the rooted cost of $\widehat{\mathbf{c}}_v$ on $\widehat{T}_v$ is given by

$$G_v(\widehat{\mathbf{c}}_v) = \boldsymbol{\lambda} \cdot \boldsymbol{\xi},$$

where $\boldsymbol{\lambda}$ is the vector of log-likelihood ratios and $\boldsymbol{\xi}$ is the vector $(\xi_1, \ldots, \xi_n)$.

Proof. From Definition 6.5, Theorem 6.7 and our hypothesis on the existence of $\xi_1, \ldots, \xi_n$, we have

$$\begin{aligned}
G_v(\widehat{\mathbf{c}}_v) &= \limsup_{m\to\infty} \frac{n}{\left|X_v^{(m)}\right|} G(\widehat{\mathbf{c}}_v^{(m)}) \\
&= \limsup_{m\to\infty} \frac{n}{\left|X_v^{(m)}\right|} \sum_{i=1}^{n} \lambda_i \left|X_v^{(m)}(x_i) \cap \operatorname{supp}(\widehat{\mathbf{c}}_v^{(m)})\right| \\
&= n \limsup_{m\to\infty} \sum_{i=1}^{n} \lambda_i \frac{\left|X_v^{(m)}(x_i) \cap \operatorname{supp}(\widehat{\mathbf{c}}_v^{(m)})\right|}{\left|X_v^{(m)}(x_i)\right|} \frac{\left|X_v^{(m)}(x_i)\right|}{\left|X_v^{(m)}\right|} \\
&= n \sum_{i=1}^{n} \lambda_i \xi_i \frac{1}{n} \\
&= \boldsymbol{\lambda} \cdot \boldsymbol{\xi},
\end{aligned}$$

as desired. ■

The results of this section will be used in Section 6.3 below to define two variants of min-sum, which will in turn lead to a proposed universal cover decoder in Section 7. First, however, we take a brief detour to examine minimal pseudocodewords on the universal cover.

6.2 Minimal Universal Cover Pseudocodewords

Recall that the theory of minimal linear programming and graph cover pseudocodewords is discussed in Section 4. The following definition gives a straightforward generalization of minimal codewords, allowing for further parallels between the analyses of universal cover pseudocodewords and LP/graph cover pseudocodewords.

Definition 6.12. A *minimal universal cover pseudocodeword* is a configuration on $\widehat{T}$ whose support does not properly contain the support of any nonzero universal cover pseudocodeword.

A deviation, as defined by Wiberg [31], is simply a finite truncation of a minimal universal cover pseudocodeword that assigns a bit value of 1 to the root node. To construct a minimal configuration $\widehat{\mathbf{c}}$ that does not assign a bit value of 1 to the root node, start with the all-zeros configuration. Pick any check node and assign a bit value of 1 to precisely two of the variable nodes immediately below it, *i.e.*, to two of its *children*. Consider these two variable nodes now as *parents*. For each check node adjacent to and below each of these two parents, assign exactly one of its children a bit value of 1. Then, considering the child variable nodes as parents at the next level of the tree, repeat this process indefinitely on the universal cover. The end result is a configuration such that, with the exception of the initially chosen check node, every check node that is adjacent to at least one variable node in the support of $\widehat{\mathbf{c}}$ has precisely one parent variable node that is assigned a 1 and one child variable node that is assigned a 1. The next proposition describes minimal configurations more precisely.

Proposition 6.13. *Let T be a Tanner graph such that each check node has degree at least 2, and let $\widehat{T}$ be its universal cover. Let $\widehat{\mathbf{c}}$ be a configuration on $\widehat{T}$, let A be the neighborhood of $\operatorname{supp}(\widehat{\mathbf{c}})$, and let S be the subgraph of $\widehat{T}$ induced by $\operatorname{supp}(\widehat{\mathbf{c}}) \cup A$. Then $\widehat{\mathbf{c}}$ is minimal if and only if S is connected and each check node in A is adjacent to exactly two variable nodes in $\operatorname{supp}(\widehat{\mathbf{c}})$.*

Proof. Suppose first that $\widehat{\mathbf{c}}$ is minimal. Then, since each connected component of S corresponds to a universal cover configuration, S must have precisely one connected component, *i.e.*, S must be connected. Further, since $\widehat{\mathbf{c}}$ is a valid configuration, every check node in A must be adjacent to a positive, even number of variable nodes in $\operatorname{supp}(\widehat{\mathbf{c}})$. By way of contradiction, suppose that some check node $f \in A$ is adjacent to at least four variable nodes in $\operatorname{supp}(\widehat{\mathbf{c}})$; call them v_1, v_2, v_3, v_4. We give an algorithm to construct a nonzero universal cover configuration from $\widehat{\mathbf{c}}$ whose support is a strict subset of the support of $\widehat{\mathbf{c}}$. First, set v_1 and v_2 equal to 0; check node f is still satisfied. Inductively, assign 0's to all the variable nodes in the sub-trees of $\widehat{T}$ that emanate from f through variable nodes v_1 and v_2. Since $\widehat{T}$ is a tree, this assignment of 0's will have no effect on the binary values of variable nodes v_3 and v_4, so the resulting configuration is a nonzero configuration whose support is properly contained in $\mathrm{supp}(\widehat{\mathbf{c}})$. This contradicts the minimality of $\widehat{\mathbf{c}}$, so each check node in A must be adjacent to exactly two variable nodes in $\mathrm{supp}(\widehat{\mathbf{c}})$.

Conversely, assume that S is connected and that each check node in A is adjacent to exactly two variable nodes in $\mathrm{supp}(\widehat{\mathbf{c}})$. Let $\widehat{\mathbf{c}}'$ be a universal cover configuration with $\mathrm{supp}(\widehat{\mathbf{c}}')$ properly contained in $\mathrm{supp}(\widehat{\mathbf{c}})$. Then there is a variable node $\widehat{x}$ in $\mathrm{supp}(\widehat{\mathbf{c}})$ such that $\widehat{c}_{\widehat{x}} = 1$ and $\widehat{c}'_{\widehat{x}} = 0$. Since $\widehat{\mathbf{c}}'$ is a valid configuration, each of the neighboring check nodes of $\widehat{x}$ must be satisfied. By assumption, each of these neighboring check nodes is connected to a unique variable node $\widehat{y} \in \mathrm{supp}(\widehat{\mathbf{c}})$. Since $\mathrm{supp}(\widehat{\mathbf{c}}')$ is a subset of $\mathrm{supp}(\widehat{\mathbf{c}})$, we must have $\widehat{c}'_{\widehat{y}} = 0$. Applying this argument inductively and using the assumption that S is connected, we see that $\widehat{\mathbf{c}}'$ must be the all-zeros configuration. Thus, $\widehat{\mathbf{c}}$ is minimal. ■

With this characterization of minimal universal cover configurations, we show in Proposition 6.14 below that, in the case of (d_X, d_F)-regular LDPC codes with $d_X, d_F \geq 3$, every minimal universal cover configuration has rooted cost equal to zero and cannot be induced by a graph cover pseudocodeword.

Proposition 6.14. *Let $T = (X \cup F, E)$ be the Tanner graph of a (d_X, d_F)-regular LDPC code of length n with $d_X, d_F \geq 3$. Let $\widehat{T}_v$ be the universal cover of T, realized as the infinite computation tree rooted at the variable node v of T. Let $\widehat{\mathbf{c}}_v$ be a minimal configuration on $\widehat{T}_v$ and assume that the coordinates of the log-likelihood vector are all finite. Then*

$$G_v(\widehat{\mathbf{c}}_v) = \lim_{m\to\infty} \overline{G}_v^{(m)}(\widehat{\mathbf{c}}_v^{(m)}) = 0.$$

Moreover, $\widehat{\mathbf{c}}_v$ is not induced by any graph cover pseudocodeword.

Proof. By Proposition 6.13, we have that $\left|X_v^{(m)} \cap \operatorname{supp}(\widehat{\mathbf{c}}_v^{(m)})\right|$ grows on the order of $(d_X-1)^m$, but the size of $X_v^{(m)}$ grows on the order of $(d_X-1)^m(d_F-1)^m$. Since $(d_X-1) < (d_X-1)(d_F-1)$, it follows that

$$\lim_{m\to\infty} \frac{\left|X_v^{(m)} \cap \operatorname{supp}(\widehat{\mathbf{c}}_v^{(m)})\right|}{\left|X_v^{(m)}\right|} = 0. \tag{6.1}$$

Set $\lambda_0 = \max\{|\lambda_i| : i = 1, \ldots, n\}$, where $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_n)$ is the log-likelihood vector. Then

$$\left|\overline{G}_v^{(m)}(\widehat{\mathbf{c}}_v^{(m)})\right| = \frac{n}{\left|X_v^{(m)}\right|}\left|G(\widehat{\mathbf{c}}_v^{(m)})\right| \leq \lambda_0 n \frac{\left|X_v^{(m)} \cap \operatorname{supp}(\widehat{\mathbf{c}}_v^{(m)})\right|}{\left|X_v^{(m)}\right|}.$$

Using (6.1), we see that $\lim_{m\to\infty} \overline{G}_v^{(m)}(\widehat{\mathbf{c}}_v^{(m)}) = 0$, and so $G_v(\widehat{\mathbf{c}}) = 0$.

It remains to be shown that $\widehat{\mathbf{c}}$ is not induced by any graph cover pseudocodeword. If it were, by Theorem 6.9 the rooted cost $G_v(\widehat{\mathbf{c}})$ would be equal to $\boldsymbol{\lambda} \cdot \boldsymbol{\omega}$, where $\boldsymbol{\omega}$ is some normalized, connected, graph cover pseudocodeword. But we have shown that $G_v(\widehat{\mathbf{c}}) = 0$ for all vectors $\boldsymbol{\lambda}$ in which all coordinates are finite, so it must be that $\boldsymbol{\omega} = 0$. Since $\widehat{\mathbf{c}}$ is not the all-zeros configuration, it is not induced by the all-zeros graph cover pseudocodeword. ■

6.3 Two Variants of Min-Sum

In this section, we propose two different variants of the min-sum decoder, with the aim of understanding how one should define universal cover decoding; see Section 7 below. Work remains in utilizing the decoders to establish a relationship between iterative message-passing decoders and the LP/ graph cover decoder. Below, we discuss currently known properties of the two decoders and conjectures related to them.

As discussed in Section 5.1, Wiberg [31] shows that for each variable node v of T, the value that min-sum assigns to v after m iterations is the same value assigned to the root node of $R_v^{(m)}$ by a minimal cost configuration on $R_v^{(m)}$. Min-sum performs this task by computing, for each $v \in X(T)$, the minimal cost $\kappa_v^{(m)}(z)$ over all configurations on $R_v^{(m)}$ that assign a binary value of z to the root node. It then assigns to the variable node v of T the binary value $\arg\min_{z=0,1} \kappa_v^{(m)}(z)$. In the case of a tie, *i.e.*, if $\kappa_v^{(m)}(0) = \kappa_v^{(m)}(1)$, the algorithm chooses a binary value arbitrarily [31].

Thus, there is a strong connection between the output of min-sum and the binary value assigned to a particular copy of variable node v in the computation tree by a minimal cost configuration. The following decoder explores the relationships between the bit-wise decision made by min-sum and the binary values assigned to all copies of v in $R_v^{(m)}$ by a minimal cost configuration.

Definition 6.15. Let T be a Tanner graph with variable nodes $x_1, \ldots, x_n$. For each $m \in \mathbb{N}$ and for $1 \leq i \leq n$, let $R_{x_i}^{(m)}$ be the computation tree of depth m rooted at x_i, and let $\mathbf{c}_i^{(m)}$ be a configuration of minimal cost on $R_{x_i}^{(m)}$. The *extended min-sum decoder* is defined to be the iterative decoder that, after m iterations, returns the vector $(\alpha_1^{(m)}, \alpha_2^{(m)}, \ldots, \alpha_n^{(m)})$, where, for $1 \leq i \leq n$,

$$\alpha_i^{(m)} = \frac{\left| X_{x_i}^{(m)}(x_i) \cap \operatorname{supp}(\mathbf{c}_i^{(m)}) \right|}{\left| X_{x_i}^{(m)}(x_i) \right|}$$

is the proportion of copies of x_i in $R_{x_i}^{(m)}$ that are assigned a bit value of 1 by $\mathbf{c}_i^{(m)}$.

Suppose, for each i and for sufficiently large m, the configurations $\mathbf{c}_i^{(m)}$ are all truncations of the same universal cover pseudocodeword $\widehat{\mathbf{c}}_i$ on $\widehat{T}_{x_i}$. Moreover, suppose that each $\widehat{\mathbf{c}}_i$ is induced by the same graph cover pseudocodeword. Then, if we set $\alpha_i = \lim_{m\to\infty} \alpha_i^{(m)}$ and $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$, Theorem 6.9 and Propositions 6.10 and 6.11 say that the vector $\boldsymbol{\alpha}$ exists and is finite and that the rooted cost of each $\widehat{\mathbf{c}}_i$ is $\boldsymbol{\lambda} \cdot \boldsymbol{\alpha}$, where $\boldsymbol{\lambda}$ is the vector of log-likelihoods. Motivated by this very special case, we make the following conjecture:

Conjecture 6.16. *Suppose there is a configuration $\widehat{\mathbf{c}}_v$ on the universal cover $\widehat{T}$ of T, realized as an infinite computation tree rooted at the variable node v of T such that*

1. $\widehat{\mathbf{c}}_v$ is induced by a codeword $\widetilde{\mathbf{c}}$ on some finite connected cover $\widetilde{T}$ of T, and

2. $G_v(\widehat{\mathbf{c}}_v) \leq G_v(\widehat{\mathbf{c}}'_v)$ for every configuration $\widehat{\mathbf{c}}'_v$ on $\widehat{T}$.

Then, if $(\omega_1, \ldots, \omega_n)$ is the normalized graph cover pseudocodeword corresponding to $\widetilde{\mathbf{c}}$ and $\alpha_1^{(m)}$, …, $\alpha_n^{(m)}$ are as in Definition 6.15, we have

$$\lim_{m \to \infty} \alpha_i^{(m)} = \omega_i.$$

In other words, if there is a configuration of minimal rooted cost on the universal cover that happens to be induced by a graph cover pseudocodeword, then the outputs of the extended min-sum decoder will converge to that (normalized) graph cover pseudocodeword as the number of iterations goes to infinity.

Of course, although a configuration $\widehat{\mathbf{c}}_v$ may have minimal rooted cost on the universal cover, the truncation of $\widehat{\mathbf{c}}_v$ to $R_v^{(m)}$ does not necessarily have minimal cost on $R_v^{(m)}$. This presents a large obstacle to proving Conjecture 6.16.

The fact that the min-sum algorithm must make an arbitrary choice in the case of a tie motivates our second decoder.

Definition 6.17. Let T be a Tanner graph with variable nodes x_1, …, x_n and, for $1 \leq i \leq n$, let $\Gamma_{x_i}^{(m)}$ be the set of all configurations on $R_{x_i}^{(m)}$ of minimal cost. For a given number of iterations m, let $\beta_i^{(m)}$ be the number of configurations in $\Gamma_{x_i}(m)$ that assign a 1 to the root node of $R_{x_i}^{(m)}$, divided by the size of $\Gamma_{x_i}^{(m)}$. The *probabilistic min-sum decoder* is defined to be the iterative decoder that, after m iterations, returns the vector $(\beta_1^{(m)}, \beta_2^{(m)}, \ldots, \beta_n^{(m)})$.

It is important to investigate whether the output of probabilistic min-sum tends to a limit as the number of iterations goes to infinity. If the aforementioned limit exists, one should also examine whether the limit has a connection either with graph cover decoding or with the outputs of the min-sum algorithm as the number of iterations increases.

7 Decoding on the Universal Cover

Definition 7.1 gives a working definition of universal cover decoding. This definition is motivated by the probabilistic min-sum decoder of Definition 6.17, with the rationale being that, *a priori*, the universal cover of a Tanner graph does not have a root node, making the idea of taking the value of a configuration at the root node artificial when working on the universal cover. With this definition of decoding, the results of restricting the decoder to universal cover configurations induced by connected graph cover configurations are examined in Proposition 7.3.

Definition 7.1. Let T be a Tanner graph with variable nodes $x_1, \ldots, x_n$ and, for $1 \leq i \leq n$, let $\widehat{T}_{x_i}$ be the universal cover of T realized as an infinite computation tree rooted at x_i. For a given received vector $\mathbf{y}$, let θ_i be the probability that a randomly chosen configuration of minimal rooted cost on $\widehat{T}_{x_i}$ has assigned a 1 to the root node x_i. *Universal cover (UC) decoding* is defined as the decoder that returns the vector

$$\mathrm{UC}(\mathbf{y}) = (\theta_1, \ldots, \theta_n).$$

To make the probabilities $\theta_1, \ldots, \theta_n$ of Definition 7.1 well-defined, one needs a positive, finite measure defined on certain subsets of the set of universal cover configurations on the universal covers $\widehat{T}_{x_i}$. The search for a meaningful probability measure is an area of current study. One particular property that this probability measure should display is given in the next definition, which is motivated by Proposition 6.10.

Definition 7.2. Let $T = (X \cup F, E)$ be a Tanner graph and let $\widehat{T}_v$ be the universal cover of T realized as an infinite computation tree rooted at the variable node v of T. A probability measure on the set of configurations on $\widehat{T}_v$ is called *admissible* if, for every normalized, connected graph cover pseudocodeword $\boldsymbol{\omega} = (\omega_x)_{x \in X}$, the probability that an arbitrarily chosen configuration on $\widehat{T}_v$ that is induced by $\boldsymbol{\omega}$ assigns a 1 to the root node of $\widehat{T}_v$ is ω_v.

Suppose an admissible measure exists for the Tanner graph $T = (X \cup F, E)$ with $X = \{x_1, \ldots, x_n\}$. Under this hypothesis, we wish to relate the output of the universal cover decoder to that of LP/graph cover decoding. To do this, we will restrict the universal cover decoder in the following manner. For each $i = 1, 2, \ldots, n$, first consider only the set of configurations on $\widehat{T}_{x_i}$ induced by connected graph cover pseudocodewords. From this set, find the set of minimal cost configurations. Let θ_i be the probability that a randomly chosen minimal cost configuration has assigned a 1 to the root node x_i, as in Definition 7.1. Define

$$\mathrm{UC}\,|_{\mathrm{GC}}(\mathbf{y}) := (\theta_1, \theta_2, \ldots, \theta_n).$$

From this point on, we will consider only (d_X, d_F)-regular LDPC codes of length n with $d_X, d_F \geq 3$ so as to utilize a number of earlier results. Let $\boldsymbol{\lambda}$ be the log-likelihood vector for the received vector $\mathbf{y}$. Theorem 6.9 shows that the rooted cost of a configuration induced by a configuration $\widetilde{\mathbf{c}}$ on a finite connected cover of T is equal to $\boldsymbol{\lambda} \cdot \boldsymbol{\omega}(\widetilde{\mathbf{c}})$, and this value is independent of the root node of $\widehat{T}$. Thus, a configuration on $\widehat{T}_{x_i}$ induced by a connected graph cover pseudocodeword $\widetilde{\mathbf{c}}$ will have minimal rooted cost if it minimizes $\boldsymbol{\lambda} \cdot \boldsymbol{\omega}$ where $\boldsymbol{\omega}$ ranges over all possible normalized connected graph cover pseudocodewords. In our situation, every graph cover pseudocodeword has a connected realization by Proposition 2.10. Thus, the set over which we are minimizing is precisely the set of rational points in the fundamental polytope $\mathcal{P}$. With this motivation, we have:

Proposition 7.3. *Let $T = (X \cup F, E)$ be the Tanner graph of a (d_X, d_F)-regular LDPC code with $d_X, d_F \geq 3$. Let $\mathcal{P}$ be the fundamental polytope of the parity-check matrix defined by T. Suppose that some $\mathbf{v} \in \mathcal{P}$ satisfies*

$$\boldsymbol{\lambda} \cdot \mathbf{v} < \boldsymbol{\lambda} \cdot \boldsymbol{\omega}$$

for every $\boldsymbol{\omega} \in \mathcal{P} \backslash \{\mathbf{v}\}$*, and that an admissible probability measure exists. Then* $\mathbf{v}$ *is a vertex of* $\mathcal{P}$*, and universal cover decoding restricted to graph cover configurations, as described above, agrees with LP/graph cover decoding; in other words,* $\mathrm{UC}\,|_{\mathrm{GC}}(\mathbf{y}) = \mathbf{v}$*, where* $\mathbf{y}$ *is the channel output.*

Proof. That $\mathbf{v}$ must be a vertex of $\mathcal{P}$ is clear. Write $X = \{x_1, \ldots, x_n\}$. Since $\mathbf{v}$ is the unique value of $\mathrm{argmin}\{\boldsymbol{\lambda} \cdot \boldsymbol{\omega} \,|\, \boldsymbol{\omega} \in \mathcal{P}\}$, a configuration $\widetilde{\mathbf{c}}$ on a finite connected cover of T induces a configuration on $\widehat{T}_{x_i}$ of minimal rooted cost among all pseudocodewords induced by connected graph cover pseudocodewords if and only if $\boldsymbol{\omega}(\widetilde{\mathbf{c}}) = \mathbf{v}$. Since the probability measure used for universal cover decoding is admissible, we have that the probability that an arbitrarily chosen element of the minimal rooted cost configurations on $\widehat{T}_{x_i}$ assigns a binary value of 1 to the root node x_i is v_i. Thus, $\mathrm{UC}\,|_{\mathrm{GC}}(\mathbf{y}) = \mathbf{v}$.

■

Thus, the proposed universal cover decoder agrees with linear programming/graph cover decoding under the conditions described in Proposition 7.3. Furthermore, this notion of universal cover decoding has an obvious connection to min-sum decoding by way of its relationship to probabilistic min-sum decoding. Further research on universal cover decoding should help to solidify our understanding of both LP/graph cover decoding and iterative message-passing decoding by providing the missing link between these two sets of decoders.

Appendix A Simulation Results for Example 3.2.

SNR	Words sent	errors due to $\mathbf{p}_1$	errors due to $\mathbf{p}_3$
0.0 dB	1990	118	31
0.5 dB	2387	135	40
1.0 dB	2792	157	43
1.5 dB	3201	133	42
2.0 dB	3424	160	48
2.5 dB	4437	173	44
3.0 dB	5833	155	44
3.5 dB	6880	187	35
4.0 dB	8234	176	40
4.5 dB	$11,865$	223	37
5.0 dB	$17,601$	258	31
5.5 dB	$23,099$	266	23
6.0 dB	$33,019$	256	31

Table 3: Simulation results for the code of Example 3.2.

Appendix B Simulation Results for Example 3.5.

word sent	total errors	errors due to $\mathbf{p}_1$	errors due to $\mathbf{p}_2$
00000000000000	6518	669	336
11100000000000	6625	689	332
00001111100000	6597	56	316
11101111100000	6545	61	299
$\mathbf{p}_1$	7018	—	215
$\mathbf{p}_2$	16438	1021	—

Table 4: Simulation results for the code of Example 3.5 at 0.0 dB.

word sent	total errors	errors due to $\mathbf{p}_1$	errors due to $\mathbf{p}_2$
00000000000000	5179	505	206
11100000000000	5406	493	254
00001111100000	5369	20	234
11101111100000	5298	32	244
$\mathbf{p}_1$	5807	—	137
$\mathbf{p}_2$	16200	833	—

Table 5: Simulation results for the code of Example 3.5 at 1.0 dB.

word sent	total errors	errors due to $\mathbf{p}_1$	errors due to $\mathbf{p}_2$
00000000000000	4084	338	151
11100000000000	4137	324	183
00001111100000	4058	14	143
11101111100000	4149	14	175
$\mathbf{p}_1$	4589	—	77
$\mathbf{p}_2$	15590	662	—

Table 6: Simulation results for the code of Example 3.5 at 2.0 dB.

word sent	total errors	errors due to $\mathbf{p}_1$	errors due to $\mathbf{p}_2$
00000000000000	2106	113	47
11100000000000	2139	109	52
00001111100000	2156	2	48
11101111100000	2174	1	49
$\mathbf{p}_1$	2437	—	29
$\mathbf{p}_2$	14394	345	—

Table 7: Simulation results for the code of Example 3.5 at 4.0 dB.

word sent	total errors	errors due to $\mathbf{p}_1$	errors due to $\mathbf{p}_2$
00000000000000	1416	41	23
11100000000000	1460	54	22
00001111100000	1393	0	21
11101111100000	1304	0	14
$\mathbf{p}_1$	1543	—	10
$\mathbf{p}_2$	13736	217	—

Table 8: Simulation results for the code of Example 3.5 at 5.0 dB.

word sent	total errors	errors due to $\mathbf{p}_1$	errors due to $\mathbf{p}_2$
00000000000000	832	17	7
11100000000000	848	15	5
00001111100000	816	0	5
11101111100000	844	0	8
$\mathbf{p}_1$	838	—	5
$\mathbf{p}_2$	13014	118	—

Table 9: Simulation results for the code of Example 3.5 at 6.0 dB.

References

[1] N. Axvig, D. Dreher, K. Morrison, E. Psota, L.C. Pérez, and J.L. Walker. Analysis of connections between pseudocodewords. Submitted for publication, March 2008.

[2] N. Axvig, D. Dreher, K. Morrison, E. Psota, L.C. Pérez, and J.L. Walker. Average min-sum decoding of LDPC codes. Accepted June, 2008; to appear in 5th International Symposium on Turbo Codes and Related Topics, December 2008.

[3] N. Axvig, D. Dreher, K. Morrison, E. Psota, L.C. Pérez, and J.L. Walker. Towards universal cover decoding. Submitted to 2008 International Symposium on Information Theory and its Applications (ISITA2008), May 2008.

[4] N. Axvig, E. Price, E. Psota, D. Turk, L.C. Pérez, and J.L. Walker. A universal theory of pseudocodewords. In *Proceedings of the 45th Annual Allerton Conference on Communication, Control, and Computing*, September 2007.

[5] C. Berrou, A. Glavieux, and P. Thitimajshima. Near Shannon limit error-correcting coding and decoding: Turbo codes 1. In *Proceedings of the 1993 IEEE International Conference on Communications*, pages 1064–1070, Geneva, Switzerland, 1993.

[6] D. Changyan, D. Proietti, 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 Transactions on Information Theory*, 48:1570–1579, June 2002.

[7] J. Feldman. *Decoding Error-Correcting Codes via Linear Programming.* PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 2003.

[8] 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 (Minneapolis, MN, 1999), volume 123 of *IMA Vol. Math. Appl.*, pages 101–112. Springer, New York, 2001.

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

[10] J.L. Gross and T.W. Tucker. *Topological Graph Theory*. Dover Publications Inc., Mineola, NY, 2001.

[11] F. Jiang. *Time-Varying and Capacity Approaching Codes*. PhD thesis, University of Nebraska, Lincoln, 2006.

[12] C. Kelley and D. Sridhara. Pseudocodewords of Tanner graphs. *IEEE Transactions on Information Theory*, 53:4013–4038, November 2007.

[13] C. Kelley, D. Sridhara, J. Xu, and J. Rosenthal. Pseudocodeword weights and stopping sets. In *Proceedings of the 2004 IEEE International Symposium on Information Theory*, page 67, Chicago, IL, 2004.

[14] R. Koetter, W.-C.W. Li, P.O. Vontobel, and J.L. Walker. Characterizations of pseudo-codewords of (low-density) parity-check codes. *Advances in Mathematics*, 213:205–229, 2007.

[15] R. Koetter and P.O. Vontobel. Graph covers and iterative decoding of finite-length codes. In *Proceedings of the IEEE International Symposium on Turbo Codes and Applications*, pages 75–82, Brest, France, 2003.

[16] Y. Li, B. Vucetic, F. Jiang, and L.C. Pérez. Recent advances in turbo code design and theory. *Proceedings of the IEEE, Special Issue on Turbo-Information Processing: Algorithms, Implementations and Applications*, 95(6):1323–1344, June 2007.

[17] D.J.C. MacKay. Good error-correcting codes based on very sparse matrices. *IEEE Transactions on Information Theory*, 45(2):399–431, March 1999.

[18] 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.

[19] E. Miller and B. Sturmfels. *Combinatorial Commutative Algebra*, volume 227 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 2005.

[20] J.R. Munkres. *Topology*. Prentice Hall, Inc., Upper Saddle River, New Jersey, 2000.

[21] R. Ortner and W. Woess. Non-backtracking random walks and cogrowth of graphs. *Canadian Journal of Mathematics*, 59(4):828–844, 2007.

[22] 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.

[23] P. Radosavljevic, A. de Baynast, and J.R. Cavallaro. Optimized message passing schedules for LDPC decoding. In *Proceedings of the 39th Asilomar Conference on Signals, Systems, and Computers*, pages 591–595, 2005.

[24] T. Richardson, A. Shokrollahi, and R. Urbanke. Design of capacity-approaching irregular low-density parity-check codes. *IEEE Transactions on Information Theory*, 47(2):619–637, February 2001.

[25] T. Richardson and R. Urbanke. The capacity of low-density parity-check codes under message-passing decoding. *IEEE Transactions on Information Theory*, 47(2):599–618, February 2001.

[26] R. Smarandache and P.O. Vontobel. Pseudo-codeword analysis of Tanner graphs from projective and Euclidean planes. *IEEE Transactions on Information Theory*, IT-53(7):2376–2393, July 2007.

[27] H.M. Stark and A.A. Terras. Zeta functions of finite graphs and coverings. *Advances in Mathematics*, 121(1):124–165, 1996.

[28] R.M. Tanner. A recursive approach to low complexity codes. *IEEE Transactions on Information Theory*, 27(5):533–547, 1981.

[29] 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*.

[30] D.B. West. *Introduction to Graph Theory*. Prentice-Hall Inc., Upper Saddle River, NJ, 2001.

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