Multidimensional Decoding Networks for Trapping Set Analysis

Allison Beemer(✉) and Christine A. Kelley

University of Nebraska-Lincoln, Lincoln, NE, USA
allison.beemer@huskers.unl.edu

Abstract. We present a novel multidimensional network model as a means to analyze decoder failure and characterize trapping sets of graph-based codes. We identify a special class of these decoding networks, which we call transitive networks, and show how they may be used to identify trapping sets and inducing sets. Many codes have transitive decoding network representations. We conclude by investigating the decoding networks of codes arising from product, half-product, and protograph code constructions.

Keywords: Trapping sets · Iterative decoding · Multidimensional decoding networks · Finite state machines · LDPC codes · Redundancy

1 Introduction

Failure of message-passing decoders of low-density parity-check (LDPC) codes has been shown to be characterized by graphical (sub)structures in the code's Tanner graph, such as pseudocodewords, absorbing sets, trapping sets, and stopping sets (a subclass of trapping sets) [1–4]. In particular, these structures contribute to persistent error floors in the Bit Error Rate (BER) curves of these codes. The presence of such structures naturally depends on the choice of Tanner graph representation used in decoding. However, while absorbing sets are combinatorially-defined, trapping sets are heavily decoder dependent, making them more difficult to analyze. Nevertheless, for many practical channels, the error floor behavior is dominated by the harmful trapping sets in the graph [3].

In this work, we present a multidimensional network model for analyzing hard-decision message-passing decoders. The structure of this network is dependent on the code, as well as the choice of Tanner graph representation and decoder. Thus, our model takes into account all parameters determining the presence of harmful trapping sets. We show how these decoding networks may be used to identify trapping sets, and therefore analyze decoder behavior of LDPC codes, as well as provide insight into the optimal number of iterations to be run on a given code (and representation) with a chosen decoder. We show that this analysis is simplified for networks with a transitivity property, and discuss the connection between transitive networks and redundancy in their corresponding parity-check matrices. Finally, we relate the decoding networks of product and

Á.I. Barbero et al. (Eds.): ICMCTA 2017, LNCS 10495, pp. 11–20, 2017.
DOI: 10.1007/978-3-319-66278-7_2

half-product codes to those of their underlying component codes, and examine the connection between the decoding networks of a protograph and its lift.

While the results herein present the case of binary linear codes transmitted over the binary erasure and symmetric channels (BEC and BSC), the definitions and results may be extended for linear codes over larger fields. Moreover, the decoding network concept and its applications may be extended to other channels and decoding algorithms. This paper is organized as follows. In Sect. 2, we provide notation and background. We present multidimensional decoding networks for decoder analysis in Sect. 3. In Sect. 4, we define trapping sets and describe a method for identifying them using this model. In Sect. 5, we discuss the connection between redundancy and transitivity, and apply the decoding network framework to various codes.

2 Preliminaries

In this section we introduce relevant background. Let $\mathcal{C}$ be a binary LDPC code of length n with associated Tanner graph G, to be decoded with some chosen hard- or soft-decision decoder. Following the notation and definitions in [5], suppose that the codeword $\mathbf{x}$ is transmitted, and $\hat{\mathbf{x}}$ is received. Let $\hat{\mathbf{x}}^\ell$ be the output after ℓ iterations of the decoder are run on G. A node $\hat{x}_i$, with $1 \leq i \leq n$, is said to be *eventually correct* if there exists $L \in \mathbb{Z}_{\geq 0}$ such that $\hat{x}_i^\ell = x_i$ for all $\ell \geq L$.

Definition 1. Let $\mathcal{T}(\hat{\mathbf{x}})$ denote the set of variable nodes that are not eventually correct for a received word $\hat{\mathbf{x}}$, and let $G[\mathcal{T}]$ denote the subgraph induced by $\mathcal{T}(\hat{\mathbf{x}})$ and its neighbors in the graph G. If $G[\mathcal{T}]$ has a variable nodes and b odd-degree check nodes, $\mathcal{T}(\hat{\mathbf{x}})$ is said to be an (a, b)-*trapping set*. In this case, the set of variable nodes in error in the received word $\hat{\mathbf{x}}$ is called an *inducing set* for $\mathcal{T}(\hat{\mathbf{x}})$. The *critical number* of a trapping set $\mathcal{T}$, denoted $m(\mathcal{T})$, is the minimum number of variable nodes in an inducing set of $\mathcal{T}$.

Note that the critical number may arise from an inducing set not fully contained in $\mathcal{T}$. Moreover, observe that stopping sets [4] are simply trapping sets over the BEC, all of whose inducing sets contain the stopping set itself. Since a stopping set is an inducing set for itself, the critical number of a stopping set is equal to its size.

Due to their complexity, trapping sets are typically analyzed under hard-decision decoders, such as Gallager A/B [6,7], although interesting work has also been done on trapping set analysis for soft-decision decoders on the AWGN channel [8,9]. Simulation results suggest that trapping sets with respect to hard decision decoders also affect the error floor performance of soft-decision decoders [5,10]. Recall that the Gallager A algorithm operates for transmission over the BSC; messages are calculated at nodes and sent across edges of the associated Tanner graph. Variable to check ($\mu_{v\to c}$) and check to variable ($\mu_{c\to v}$) messages are calculated as follows:

$$\mu_{v\to c}(r, m_1, \ldots, m_{d(v)-1}) = \begin{cases} r+1 \text{ if } m_1 = \cdots = m_{d(v)-1} = r+1 \\ r \text{ else,} \end{cases} \tag{1}$$

where r is the received value at variable node v, d_v is the degree of v, and $m_1, \ldots, m_{d(v)-1}$ are the most recent messages received from the check nodes which are not those to which the current message is being calculated. In the other direction,

$$\mu_{c \to v}(m_1, \ldots, m_{d(c)-1}) = \sum_{i=1}^{d(c)-1} m_i \ , \tag{2}$$

where here $m_1, \ldots, m_{d(c)-1}$ are messages received from variable nodes. Note that calculations are performed in $\mathbb{F}_2$. Gallager B relaxes the unanimity condition of $\mu_{v \to c}$. Throughout the examples in this work which use Gallager decoding, we consider Gallager A decoding with the final decoder output at a variable node v given by the received value at v, unless incoming messages $\mu_{c \to v}$ for c adjacent to v unanimously disagree with this received value. Further results for Gallager B and more relaxed decoding rules will be treated in a forthcoming work.

3 Multidimensional Network Framework

Suppose we decode a code $\mathcal{C}$ using a hard-decision decoder, and consider the labeled directed graph (digraph) for a fixed $\ell \in \mathbb{Z}_{>0}$, denoted $\mathcal{D}_\ell$, with vertex and edge sets $V = \{\mathbf{x} : \mathbf{x} \in S\}$ and $E = \{(\mathbf{x}_i, \mathbf{x}_j, \ell) : \mathbf{x}_i^\ell = \mathbf{x}_j\}$, respectively, where S is the set of possible received words, $(\mathbf{x}_i, \mathbf{x}_j, \ell)$ denotes an edge from $\mathbf{x}_i$ to $\mathbf{x}_j$ with edge label ℓ, and $\mathbf{x}_i^\ell$ is the output of the decoder after ℓ iterations with input $\mathbf{x}_i$. Note that we allow loops, which are edges of the form $(\mathbf{x}_i, \mathbf{x}_i, \ell)$. For simplicity, we refer to the label of a vertex – that is, its corresponding word in S – interchangeably with the vertex itself. There will be a potentially distinct digraph on this same vertex set for each choice of $\ell \in \mathbb{Z}_{>0}$. We call the union of these digraphs for all $\ell \in \mathbb{Z}_{>0}$ the *(multidimensional) decoding network* corresponding to the code $\mathcal{C}$ and the specific choice of decoder, as we may consider the digraph which incorporates the information for all ℓ as a *multidimensional network*.

Definition 2 ([11]). A *multidimensional network* is an edge-labeled directed graph $\mathcal{D} = (V, E, D)$, where V is a set of vertices, D a set of edge labels, called *dimensions*, and E is a set of triples (u, v, d) where $u, v \in V$ and $d \in D$. We say that an edge (or vertex) *belongs to* a given dimension d if it is labeled d (or is incident to an edge labeled d).

In this framework, the decoding network is a multidimensional network with $D = \mathbb{Z}_{>0}$, and each edge labeled with the number of decoder iterations, ℓ, to which it corresponds. Notice that, in any given dimension (i.e. iteration), every vertex has outdegree equal to one. Next, we introduce an important type of decoding network.

Definition 3. We say that a decoding network is *transitive* if $(v_1, v_2, \ell) \in E$ if and only if for every choice of $1 \leq k \leq \ell - 1$, there exists $v_k \in V$ such that $(v_1, v_k, k), (v_k, v_2, \ell - k) \in E$. We say a decoder is transitive for a code $\mathcal{C}$ and a choice of representation of $\mathcal{C}$ if its resulting decoding network is transitive.

Transitivity corresponds to the case in which running i iterations of the decoder on a received word, and using that output as input to a subsequent j decoding iterations, is equivalent to running $i + j$ iterations on the original received word.

Let $\mathcal{D}_\ell$ denote the digraph corresponding to the ℓth dimension of the decoding network $\mathcal{D}$, and let $A(\mathcal{D}_\ell)$ denote the adjacency matrix of the digraph $\mathcal{D}_\ell$. Observe that a decoding network $\mathcal{D}$ is transitive if and only if $A(\mathcal{D}_\ell) = (A(\mathcal{D}_1))^\ell$ for all $\ell \geq 1$, as the product $(A(\mathcal{D}_1))^\ell$ gives directed paths of length ℓ in dimension 1 of $\mathcal{D}$.

Example 1. Consider a simple binary parity-check code of length n, with parity-check matrix given by the $1 \times n$ all-ones matrix. The Tanner graph representation of such a code is a single check node with n variable node leaves. Thus, if codewords are sent over the BSC and decoded with the Gallager A algorithm, the message the solitary check node sends to each of its adjacent variable nodes is either (a) the node's channel value, if a codeword is received, or (b) the opposite of its originally-received value, otherwise. Each variable node will always send back its channel value. For any number of iterations, codewords will decode to codewords, and any non-codeword $\hat{\mathbf{x}}$ will be decoded to $\hat{\mathbf{x}} + \mathbf{1}$. If n is odd, every received word will decode to a codeword, and the network will be transitive. If n is even, $\hat{\mathbf{x}} + \mathbf{1}$ will not be a codeword, and the network will not be transitive. The cases $n = 3$ and $n = 4$ are shown in Fig. 1; in both networks, edges belonging to higher dimensions are suppressed, as all dimensions are identical.

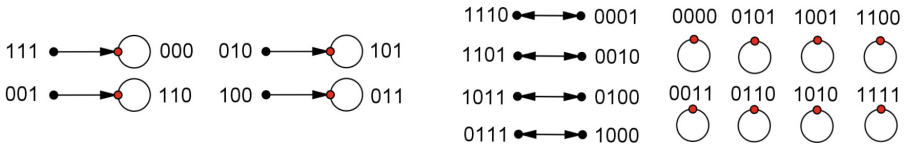
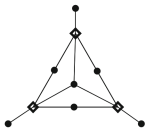


Fig. 1. The decoding networks of parity-check codes of lengths 3 (left) and 4 (right).

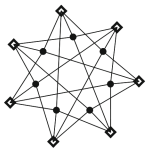
Example 2. Consider the binary Hamming code of length $7 = 2^3 - 1$, denoted $\mathscr{H}_3$. Recall that this code's canonical 3×7 parity-check matrix has columns consisting of all nonzero binary words of length 3. The corresponding Tanner graph may be seen in representation A of Fig. 2. However, $\mathscr{H}_3$ may also be defined by the parity-check matrix whose Tanner graph is representation B in Fig. 2. Under Gallager A decoding, representation A does not yield a transitive decoding network. However, if representation B is decoded via Gallager A, the resulting decoding network is transitive: every word decodes under a single iteration to a codeword, and decodes identically for any higher number of iterations.

If a decoder is transitive for all representations of all codes $\mathcal{C}$, we say that it is *universally transitive*. Any decoder which ignores channel values at each subsequent iteration will be universally transitive; some examples are given next.

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



A



B

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

Fig. 2. Two distinct Tanner graph representations of $\mathscr{H}_3$, along with their parity-check matrices. Variable nodes are denoted by $\bullet$, and check nodes are denoted by $\Diamond$.

Example 3. If codewords from a code $\mathcal{C}$ are sent over the BEC, and words are decoded using a peeling decoder which iteratively corrects erasures, then the corresponding decoding network of $\mathcal{C}$ is universally transitive. Indeed, corrections at each iteration are performed regardless of previous iterations. Similarly, iterative bit-flipping decoders over the BSC are universally transitive.

4 Trapping Set Characterization

Within this multidimensional decoding network framework, trapping sets of a code transmitted over the BSC may be determined by looking at the supports of the words corresponding to vertices in the decoding network that have nonzero indegree in an infinite number of dimensions. That is,

Theorem 1. *For each vertex $\mathbf{x} \in V = \mathbb{F}_2^n$ in a decoding network $\mathcal{D} = (V, E, D)$ for a code $\mathcal{C}$, let $M_{\mathbf{x}}$ be the set of vertices $\mathbf{y} \in V$ for which there is an edge $(\mathbf{x}, \mathbf{y}, \ell) \in E$ for infinitely many choices of ℓ. Then the set of variable nodes corresponding to*

$$\bigcup_{\mathbf{y} \in M_{\mathbf{x}}} \operatorname{supp}(\mathbf{y}) \ ,$$

denoted $\mathcal{T}(\mathbf{x})$, is a trapping set with an inducing set given by the variable nodes corresponding to $\operatorname{supp}(\mathbf{x})$*. Furthermore, the set of trapping sets of the code $\mathcal{C}$ is*

$$\{\mathcal{T}(\mathbf{x}) : \mathbf{x} \in \mathbb{F}_2^n\} \ ,$$

and, given a trapping set $\mathcal{T}$, its set of inducing sets is given by the variable nodes corresponding to

$$\{\operatorname{supp}(\mathbf{x}) : \mathcal{T}(\mathbf{x}) = \mathcal{T}\} \ ,$$

and its critical number is

$$m(\mathcal{T}) = \min\{|\operatorname{supp}(\mathbf{x})| : \mathcal{T}(\mathbf{x}) = \mathcal{T}\} \ .$$

Proof. Assuming that the all-zero codeword was sent over the BSC, any decoding errors will be given by 1's. If, during the process of decoding the received word $\mathbf{x}$, there is some word $\mathbf{y}$ such that an edge from $\mathbf{x}$ to $\mathbf{y}$ occurs in an infinite number of network dimensions, the support of $\mathbf{y}$ gives variable node positions which are

not eventually correct. By definition, these variable nodes belong to a trapping set induced by the variable nodes of the support of $\mathbf{x}$. However, these may not be the only variable nodes that are not eventually correct given the received word $\mathbf{x}$. Taking the union of the supports of all such $\mathbf{y}$ gives us our expression for $\mathcal{T}(\mathbf{x})$, the trapping set induced by $\mathbf{x}$. Repeating this for each possible received word, we find all trapping sets of the code. Note that each trapping set may be induced by multiple received words.

Example 4. Let $\mathcal{C}$ be the binary repetition code of length 3, with the parity-check matrix

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

The associated Tanner graph is a path of length 4 with variable nodes v_1, v_2, and v_3. Let $\mathcal{D}$ be the (non-transitive) decoding network of $\mathcal{C}$ under Gallager A, shown in Fig. 3. Assuming $\mathbf{0}$ was transmitted, the received word 011, for example, decodes in one iteration to the codeword 111, but decodes for any number of iterations greater than 1 to the (non-code)word 110. Thus, $\{v_1, v_2\}$ is a (2,1)-trapping set. Note that the support of 011, which induces this trapping set, is not contained in the trapping set. Similarly, $\{v_1, v_2, v_3\}$ is a trapping set corresponding to the codeword 111, with inducing sets $\{v_1, v_3\}$ and $\{v_1, v_2, v_3\}$. Other trapping sets of the code include $\{v_2, v_3\}$ (induced by $\{v_1, v_2\}$), $\{v_1\}$ (induced by $\{v_3\}$), and $\{v_3\}$ (induced by $\{v_1\}$).

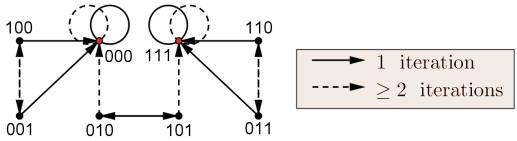


Fig. 3. The decoding network of a binary repetition code of length 3.

In the case of transitive decoding networks, trapping sets may be identified by looking only at dimension 1, as follows:

Corollary 1. *If the decoding network, $\mathcal{D}$, of a code $\mathcal{C}$ is transitive, then the trapping sets are given by (1) the sets of variable nodes corresponding to supports of vertices with loops in $\mathcal{D}_1$, and (2) the sets of variable nodes corresponding to unions of the supports of vertices forming directed cycles in $\mathcal{D}_1$. Furthermore, inducing sets of trapping sets in a transitive decoding network are given by the variable nodes corresponding to the support of any vertex which has a directed path to either a (nonzero) vertex with a loop, or to a directed cycle, regardless of where that path enters the cycle.*

Proof. In a transitive decoding network, the edges in dimension ℓ correspond to directed paths of length ℓ in $\mathcal{D}_1$. Thus, in order for a word to appear as

the output of the decoder in an infinite number of dimensions, it must be the terminating vertex of infinitely many directed paths (of distinct lengths) from the received word. Because the decoding network for a code is finite, and the outdegree of every vertex in $\mathcal{D}_1$ is equal to 1, this can only occur if there is a loop at that vertex, or if it belongs to a directed cycle. The result follows from Theorem 1.

In the adjacency matrix of dimension 1 of a decoding network, nonzero diagonal entries indicate loops. There are numerous algorithms for finding directed cycles in a digraph, such as Depth-First Search (DFS) [12]. We further note that several works have addressed the computational aspects of finding and/or removing trapping sets [10,13–15].

We conclude this section by discussing the ideal number of iterations given a fixed code and decoder. Note that in Example 4, if the all-zero codeword was sent, more received words are decoded correctly if only a single iteration of the decoder is run, rather than multiple iterations. To this end, we introduce a parameter we call the *decoding diameter*.

Definition 4. Let $\mathcal{D} = (V, E, D)$ be the decoding network of a code $\mathcal{C}$ under a fixed decoder. For $\mathbf{x} \in V$, let $L_{\mathbf{x}}$ be the minimum nonnegative integer such that for all $\ell \geq L_{\mathbf{x}}$, $\mathbf{x}^\ell$, the output of the decoder after ℓ iterations, appears an infinite number of times in the sequence $\{\mathbf{x}^k\}_{k=L_{\mathbf{x}}}^{\infty}$. Then, the *decoding diameter* of $\mathcal{D}$ is given by $\Delta(\mathcal{D}) = \max_{\mathbf{x}\in V} L_{\mathbf{x}}$.

After running $\Delta(\mathcal{D})$ iterations, all errors will be contained in trapping sets of the code. In Example 2, $\mathscr{H}_3$ with canonical graph representation A decoded via the Gallager A algorithm has decoding diameter 3, and $\mathscr{H}_3$ with representation B has decoding diameter 1. The decoding diameter of a transitive decoding network is the maximum distance in $\mathcal{D}_1$ of any vertex to either a vertex with loop or to a directed cycle.

5 Representations Yielding Transitivity

Due to the effect of the choice of representation on a decoding network's structure, it is natural to ask which representations, if any, ensure that a code's decoding network is transitive under a fixed decoder. Recall from Example 2 that the canonical representation of the Hamming code $\mathscr{H}_3$ does not yield a transitive decoding network under Gallager A, while the decoding network arising from the representation given by the parity-check matrix including all nonzero codewords of the dual code is transitive. In fact,

Theorem 2. *Every binary linear code has a parity-check matrix representation whose corresponding decoding network is transitive under Gallager A decoding.*

In particular, adding exactly the nonzero codewords of the dual to the parity-check matrix of a code (a representation which we will refer to as the *complete representation*) will result in a transitive decoding network under Gallager

A decoding: using symmetries of the complete representation, we may show that after a single iteration of the decoder, the received word either decodes to a codeword and continues decoding to that codeword for any number of iterations, or it will decode to itself under any number of iterations. The full proof of this result will appear in the full version of this paper.

While the complete representation establishes the existence of a representation yielding a transitive network for any code, this level of redundancy is not necessarily required, and in fact may create an excess of trapping sets in the code. In Example 2, adding row seven of representation B to the canonical representation A gives a transitive network, as does adding any three additional rows to the canonical representation. However, any other combination of row additions does not yield a transitive network. Thus, it is interesting to determine the minimum level of redundancy needed to yield a transitive decoding network for a code with a fixed choice of decoder.

In the remainder of this section, we apply the decoding network model to product and half-product codes, which have been subject to renewed interest [16–18], as well as to codes constructed from protographs. By phrasing the decoding networks of these classes of codes in terms of the decoding networks of their component codes, we can identify trapping sets in the larger codes with fewer up-front computations. Proofs have been omitted for space.

Definition 5. Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be binary linear codes of lengths n and m, respectively. Then the *product code* $\mathcal{C}_1 \times \mathcal{C}_2$ is the set of $m \times n$ binary arrays such that each row forms a codeword in $\mathcal{C}_1$, and each column forms a codeword in $\mathcal{C}_2$.

Consider a product code, $\mathcal{C}_1 \times \mathcal{C}_2$, with a decoder that operates by iteratively decoding one component code at a time. At each iteration, channel information is dispensed with and decoding is performed based solely on the current estimate.

Theorem 3. *Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be codes of lengths n and m, respectively, with decoding networks $\mathcal{D}^1$ and $\mathcal{D}^2$. Let A_1 be the adjacency matrix of the directed graph product $(\mathcal{D}_1^1)^m$, and let A_2 be the adjacency matrix of $(\mathcal{D}_1^2)^n$. Then, the adjacency matrix of dimension ℓ of the transitive decoding network, $\mathcal{D}$, of the product code $\mathcal{C}_1 \times \mathcal{C}_2$ is given by $(A_1 A_2)^\ell$.*

Definition 6 ([17]). Let $\mathcal{C}$ be a binary linear code of length n, and let $\tilde{\mathcal{C}}_H = \{X - X^T : X \in \mathcal{C} \times \mathcal{C}\}$. The *half-product code* with component code $\mathcal{C}$, denoted $\mathcal{C}_H$, is obtained from $\tilde{\mathcal{C}}_H$ by setting the symbols below the diagonal of each element of $\tilde{\mathcal{C}}_H$ equal to zero.

A decoder runs by iteratively decoding at each of the n constraints corresponding to "folded" codewords, as in [17,18]. Again, channel information is dispensed with at each subsequent iteration. Let A_i be the adjacency matrix of the digraph on the vertex set of the decoding network of the half-product code, $\mathcal{D}^H$, which gives the behavior of a single decoder iteration run on the ith constraint code. While decoding is performed on the ith constraint, all $(n-1)(n-2)/2$ symbols not participating in constraint i are fixed. Let $\mathcal{D}^i$ be the decoding network associated with the ith constraint code. Then, A_i is the adjacency matrix

of a disjoint union of $2^{(n-1)(n-2)/2}$ copies of $\mathcal{D}^i$, corresponding to all the ways non-participating symbols may be fixed. Permute the rows and columns of the A_i's so that they all correspond to a single ordering of the vertices in $\mathcal{D}^H$, and let S_n denote the symmetric group on n elements.

Theorem 4. *The product $(A_{\sigma(1)} \cdots A_{\sigma(n)})^\ell$, where $\sigma \in S_n$, gives the adjacency matrix of $\mathcal{D}_\ell^H$, dimension ℓ of the decoding network of the half-product code $\mathcal{C}_H$, where the component constraints are decoded in the order determined by σ.*

Theorem 5. *Let $\mathcal{C}$ be a binary linear code with Tanner graph G and decoding network $\mathcal{D}$ with respect to a fixed decoder. Viewing G as a protograph, let $\hat{\mathcal{C}}$ be the code corresponding to a degree h lift of G, denoted $\hat{G}$, and (with an abuse of notation), let $\hat{\mathcal{D}}$ be the decoding network of $\hat{\mathcal{C}}$ with respect to the same decoder. Then, there exists a subgraph of $\hat{\mathcal{D}}$ that is isomorphic to $\mathcal{D}$. In particular, if $\mathcal{D}$ is not transitive, then $\hat{\mathcal{D}}$ is not transitive. However, transitivity of $\mathcal{D}$ does not necessarily imply transitivity of $\hat{\mathcal{D}}$.*

6 Conclusions

We presented a multidimensional network framework for the analysis of trapping sets under hard-decision message-passing decoders. We showed that when the decoding network of a code is transitive, trapping sets and their inducing sets can be found by examining the behavior of the decoder in a single iteration. We further showed that all codes have a transitive decoding network representation under Gallager A decoding.

This work leads to many interesting avenues for future pursuit: determining, for certain classes of codes, the minimum number of parity-check matrix rows needed to obtain a transitive decoding network, using the transitive networks of these codes to improve existing results on their trapping and inducing sets, characterizing which lifts of transitive decoding networks preserve transitivity, developing an efficient method of finding trapping sets and predicting the decoding diameter for non-transitive networks, and extending these results to Gallager B and other decoding algorithms. As a first direction, an interesting open problem is to determine a graph theoretic condition (or equivalently, a condition on the parity-check matrix) that guarantees transitivity.

Acknowledgements. We thank the anonymous reviewers for their useful comments that improved the quality of the paper.

References

1. Koetter, R., Vontobel, P.: Graph-covers and iterative decoding of finite length codes. In: Proceedings of the 3rd International Symposium on Turbo Codes (2003)
2. Dolecek, L., Zhang, Z., Anantharam, V., Wainwright, M., Nikolic, B.: Analysis of absorbing sets for array-based LDPC codes. In: IEEE International Conference on Communications, pp. 6261–6268 (2007)

3. Richardson, T.: Error-floors of LDPC codes. In: Proceedings of the 41st Allerton Conference on Communication, Control and Computing, pp. 1426–1435 (2003)
4. Di, C., Proietti, D., Telatar, I.E., Richardson, T.J., Urbanke, R.L.: Finite-length analysis of low-density parity-check codes on the binary erasure channel. IEEE Trans. Inf. Theor. **48**, 1570–1579 (2002)
5. Nguyen, D.V., Chilappagari, S.K., Marcellin, M.W., Vasic, B.: On the construction of structured LDPC codes free of small trapping sets. IEEE Trans. Inf. Theor. **58**, 2280–2302 (2012)
6. Sankaranarayanan, S., Chilappagari, S.K., Radhakrishnan, R., Vasić, B.: Failures of the Gallager B decoder: analysis and applications. In: IEEE International Conference on Communications (2006)
7. Gallager, R.: Low-density parity-check codes. IRE Trans. Inf. Theor. **8**, 21–28 (1962)
8. Sun, J.: Studies on graph-based coding systems. Ph.D. Dissertation, Department of Electrical and Computer Engineering, University of California, Columbus (2004)
9. Butler, B.K., Siegel, P.H.: Error floor approximation for LDPC codes in the AWGN channel. IEEE Trans. Inf. Theor. **60**, 7416–7441 (2014)
10. Vasic, B., Chilappagari, S.K., Nguyen, D.V., Planjery, S.K.: Trapping set ontology. In: Proceedings of the 47th Allerton Conference on Communication, Control, and Computing, pp. 1–7 (2009)
11. Berlingerio, M., Coscia, M., Giannotti, F., Monreale, A., Pedreschi, D.: Foundations of multidimensional network analysis. In: International Conference on Advances in Social Networks Analysis and Mining, pp. 485–489 (2011)
12. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 3rd edn. MIT Press, Cambridge (2009)
13. Beemer, A., Kelley, C.A.: Avoiding trapping sets in SC-LDPC codes under windowed decoding. In: Proceedings IEEE International Symposium on Information Theory and Applications (2016)
14. Karimi, M., Banihashemi, A.H.: Efficient algorithm for finding dominant trapping sets of LDPC codes. IEEE Trans. Inf. Theor. **58**, 6942–6958 (2012)
15. Hashemi, Y., Banihashemi, A.H.: Characterization and efficient exhaustive search algorithm for elementary trapping sets of irregular LDPC codes. In: Proceedings of IEEE International Symposium on Information Theory (2017)
16. Justesen, J.: Performance of product codes and related structures with iterated decoding. IEEE Trans. Commun. **59**, 407–415 (2011)
17. Mittelholzer, T., Parnell, T., Papandreou, N., Pozidis, H.: Improving the error-floor performance of binary half-product codes. In: Proceedings of IEEE International Symposium Information Theory and Applications (2016)
18. Pfister, H.D., Emmadi, S.K., Narayanan, K.: Symmetric product codes. In: Information Theory and Applications Workshop, pp. 282–290 (2015)

Springer

http://www.springer.com/978-3-319-66277-0

Coding Theory and Applications
5th International Castle Meeting, ICMCTA 2017, Vihula, Estonia, August 28-31, 2017, Proceedings
Barbero, Á.I.; Skachek, V.; Ytrehus, Ø. (Eds.)

2017, X, 297 p. 29 illus., Softcover
ISBN: 978-3-319-66277-0