The Extraction and Complexity Limits of Graphical Models for Linear Codes

Thomas R. Halford, *Member, IEEE*, and Keith M. Chugg, *Member, IEEE*

***Abstract*—Two broad classes of graphical modeling problems for codes can be identified in the literature: constructive and extractive problems. The former class of problems concern the *construction* of a graphical model in order to define a new code. The latter class of problems concern the *extraction* of a graphical model for a (fixed) given code. The design of a new low-density parity-check code for some given criteria (e.g., target block length and code rate) is an example of a constructive problem. The determination of a graphical model for a classical linear block code that implies a decoding algorithm with desired performance and complexity characteristics is an example of an extractive problem. This work focuses on extractive graphical model problems and aims to lay out some of the foundations of the theory of such problems for linear codes. The primary focus of this work is a study of the space of all graphical models for a (fixed) given code. The tradeoff between cyclic topology and complexity in this space is characterized via the introduction of a new bound: the forest-inducing cut-set bound (FI-CSB). The proposed bound provides a more precise characterization of this tradeoff than that which can be obtained using existing tools (e.g., the CSB) and can be viewed as a generalization of the square-root bound for tail-biting trellises to graphical models with arbitrary cyclic topologies. Searching the space of graphical models for a given code is then enabled by introducing a set of basic graphical model transformation operations that are shown to span this space. Finally, heuristics for extracting novel graphical models for linear block codes using these transformations are investigated.**

***Index Terms*—Codes on graphs, complexity measures, cut-set bound (CSB), graphical model complexity, graphical model extraction, graphical model transformation, linear codes, normal realizations, square-root bound.**

I. Introduction

GRAPHICAL models of codes have been studied since the 1960s and this study has intensified in recent years due to the discovery of turbo codes by Berrou *et al.* [1], the rediscovery of Gallager's low-density parity-check (LDPC) codes [2] by Spielman *et al.* [3] and MacKay *et al.* [4], and the pioneering work of Wiberg, Loeliger, and Koetter [5], [6]. It is now well known that together with a suitable message passing schedule, a graphical model implies a soft-in–soft-out (SISO) decoding algorithm, which is optimal for cycle-free models and suboptimal, yet often substantially less complex, for cyclic models (cf., [6]–[10]). It has been observed empirically in the literature that there exists a correlation between the cyclic topology of a graphical model and the performance of the decoding algorithms implied by that graphical model (cf., [5] and [10]–[16]). To summarize this empirical "folk-knowledge," those graphical models that imply near-optimal decoding algorithms tend to have large girth, a small number of short cycles, and a cycle structure that is not overly regular.[1]

Two broad classes of graphical modeling problems can be identified in the literature:

- *constructive* problems: given a set of design requirements, design a suitable code by constructing a good graphical model (i.e., a model that implies a low-complexity, near-optimal decoding algorithm);
- *extractive* problems: given a specific (fixed) code, extract a graphical model for that code that implies a decoding algorithm with desired complexity and performance characteristics.

Constructive graphical modeling problems have been widely addressed by the coding theory community. Capacity-approaching LDPC codes have been designed for both the additive white Gaussian noise (AWGN) channel (cf., [19] and [20]) and the binary erasure channel (cf., [21]–[23]). Other classes of modern codes have been successfully designed for a wide range of practically motivated block lengths and rates (cf., [24]–[28]).

Less is understood about extractive graphical modeling problems, however. The extractive problems that have received the most attention are those concerning Tanner graph [11] and trellis representations of block codes. Tanner graphs imply low-complexity decoding algorithms; however, the Tanner graphs corresponding to many block codes of practical interest, e.g., high-rate Reed–Muller (RM), Reed–Solomon (RS), and Bose–Chaudhuri–Hocquenghem (BCH) codes, necessarily contain many short cycles [29] and thus usually imply poorly performing decoding algorithms. There is a well-developed theory of conventional trellises [30] and tail-biting trellises [31], [32] for linear block codes. Conventional and tail-biting trellises imply optimal and, respectively, near-optimal decoding algorithms; however, for many block codes of practical interest, these decoding algorithms are prohibitively complex thus

Manuscript received November 17, 2006; revised February 18, 2008. Published August 27, 2008 (projected). This work was supported in part by the Powell Foundation and by the U.S. Army Research Office under MURI Contract DAAD19-01-1-0477. The material in this paper was presented in part at the 44th Allerton Conference on Communication, Control, and Computing, Monticello, IL, September 2006. The work of T. R. Halford was carried out at the University of Southern California.

T. R. Halford is with TrellisWare Technologies, Inc., San Diego, CA 92127-1708 USA (e-mail: thalford@trellisware.com).

K. M. Chugg is with the Communication Sciences Institute, University of Southern California, Los Angeles, CA 90089-2565 USA (e-mail: chugg@usc.edu).

Communicated by G. Seroussi, Associate Editor for Coding Theory.

Digital Object Identifier 10.1109/TIT.2008.928271

[1]There are a number of notable exceptions to this "folk-knowledge," e.g., LDPC codes based on finite geometries (cf., [17], [18]), which perform well despite having Tanner graphs with many short cycles.

motivating the study of more general graphical models (i.e., models with a richer cyclic topology than a single cycle).

The goal of this work is to lay out some of the foundations of the theory of extractive graphical modeling problems. Following a review of graphical models for codes in Section II, a complexity measure for graphical models is introduced in Section III. A number of properties of graphical models related to this measure are described in Section III and defined precisely in the Appendix. The proposed measure captures a cyclic graphical model analog of the familiar notions of state and branch complexity for trellises [30]. The *minimal tree complexity* of a code, which is a natural generalization of the well-understood minimal trellis complexity of a code to arbitrary cycle-free models, is then defined using this measure.

The tradeoff between cyclic topology and complexity in graphical models is studied in Section IV. Wiberg's cut-set bound (CSB) is the existing tool that best characterizes this fundamental tradeoff [6]. While the CSB can be used to establish the square-root bound for tail-biting trellises [31] and thus provides a precise characterization of the potential tradeoff between cyclic topology and complexity for single-cycle models, as was first noted by Wiberg *et al.* [5], it is very challenging to use the CSB to characterize this tradeoff for graphical models with cyclic topologies richer than a single cycle. To provide a more precise characterization of this tradeoff than that offered by the CSB alone, this work introduces a new bound in Section IV—the forest-inducing cut-set bound (FI-CSB)—which may be viewed as a generalization of the square-root bound to graphical models with arbitrary cyclic topologies. Specifically, it is shown that an rth-root complexity reduction (with respect to the minimal tree complexity as defined in Section III) requires the introduction of *at least* $r(r-1)/2$ cycles. The proposed bound can thus be viewed as an extension of the square-root bound to graphical models with arbitrary cyclic topologies.

Much as there are many valid complexity measures for conventional trellises, there are many reasonable metrics for the measurement of cyclic graphical model complexity (cf., [33]). While there exists a unique minimal trellis for any linear block code that simultaneously minimizes all reasonable measures of trellis complexity [34], even for the class of cyclic graphical models with the most basic cyclic topology—tail-biting trellises—minimal models are not unique [32], thus motivating the consideration of complexity measures other that introduced in Section III. In Section V, it is shown that, provided a given complexity measure obeys some reasonable properties, then a generalization of the FI-CSB for that particular measure can be made. In particular, a measure that is a slight relaxation of that introduced in Section III is examined in detail.

The transformation of graphical models is studied in Sections VI and VII. Whereas minimal conventional and tail-biting trellis models can be characterized algebraically via trellis-oriented generator matrices [30], there is, in general, no known analog of such algebraic characterizations for arbitrary cycle-free graphical models [35], let alone cyclic models. In the absence of such an algebraic characterization, it is initially unclear as to how cyclic graphical models can be extracted. In Section VI, a set of basic transformation operations on graphical models for codes is introduced and it is shown that any graphical model for a given code can be transformed into any other graphical model for that same code via the application of a finite number of these basic transformations. The transformations studied in Section VI thus provide a mechanism for searching the space of *all* graphical models for a given code. The Appendix provides a number of examples that illustrate these basic transformations. In Section VII, the basic transformations introduced in Section VI are used to extract novel graphical models for linear block codes. Starting with an initial Tanner graph for a given code, heuristics for extracting other Tanner graphs, generalized Tanner graphs, and more complex cyclic graphical models are investigated. Concluding remarks and directions for future work are given in Section VIII.

II. Background

A. Notation

The binomial coefficient is denoted $\binom{a}{b}$ where $a, b \in \mathbb{Z}$ are integers. The finite field with q elements is denoted $\mathbb{F}_q$. Given a finite index set I, the vector space over $\mathbb{F}_q$ defined on I is the set of vectors

$$\mathbb{F}_q^I = \{\boldsymbol{f} = (f_i \in \mathbb{F}_q, i \in I)\}. \tag{1}$$

Suppose that $J \subseteq I$ is some subset of the index set I. The *projection* of a vector $\boldsymbol{f} \in \mathbb{F}_q^I$ onto J is denoted

$$\boldsymbol{f}_{|J} = (f_i, i \in J). \tag{2}$$

B. Codes, Projections, and Subcodes

Given a finite index set I, a *linear code* over $\mathbb{F}_q$ defined on I is some vector subspace $\mathcal{C} \subseteq \mathbb{F}_q^I$. The *block length*, *dimension*, and *rate* of $\mathcal{C}$ are denoted $n(\mathcal{C}) = |I|, k(\mathcal{C}) = \dim \mathcal{C}$, and $R(\mathcal{C}) = k(\mathcal{C})/n(\mathcal{C})$, respectively. If known, the minimum Hamming distance of $\mathcal{C}$ is denoted $d(\mathcal{C})$ and $\mathcal{C}$ may be described by the triplet $[n(\mathcal{C}), k(\mathcal{C}), d(\mathcal{C})]$. This work considers only linear codes and the terms code and linear code are used interchangeably.

A code $\mathcal{C}$ can be described by an $r_G \times n(\mathcal{C}), r_G \geq k(\mathcal{C})$, *generator matrix* $G_{\mathcal{C}}$ over $\mathbb{F}_q$, the rows of which span $\mathcal{C}$. An $r_G \times n(\mathcal{C})$ generator matrix is *redundant* if r_G is strictly greater than $k(\mathcal{C})$. A code $\mathcal{C}$ can also be described by an $r_H \times n(\mathcal{C}), r_H \geq n(\mathcal{C}) - k(\mathcal{C})$, *parity-check matrix* $H_{\mathcal{C}}$ over $\mathbb{F}_q$, the rows of which span the null space of $\mathcal{C}$ (i.e., the dual code $\mathcal{C}^{\perp}$). Each row of $H_{\mathcal{C}}$ defines a *q-ary single parity-check equation*, which every codeword in $\mathcal{C}$ must satisfy. An $r_H \times n(\mathcal{C})$ parity-check matrix is *redundant* if r_H is strictly greater than

$$k(\mathcal{C}^{\perp}) = n(\mathcal{C}) - k(\mathcal{C}). \tag{3}$$

Given a subset $J \subseteq I$ of the index set I, the *projection* of $\mathcal{C}$ onto J is the set of all codeword projections

$$\mathcal{C}_{|J} = \{\boldsymbol{c}_{|J}, \boldsymbol{c} \in \mathcal{C}\}. \tag{4}$$

Note that $\mathcal{C}_{|J}$ can be interpreted as the code $\mathcal{C}$ punctured at $I \setminus J$. Closely related to $\mathcal{C}_{|J}$ is the *subcode* $\mathcal{C}_J$: the projection onto J of the subset of codewords satisfying $c_i = 0$ for $i \in I \setminus J$. Note

that $\mathcal{C}_J$ can be interpreted as the code $\mathcal{C}$ shortened at $I \setminus J$. Both $\mathcal{C}_{|J}$ and $\mathcal{C}_J$ are linear codes.

While code projections and subcodes correspond to codes defined on a subset of the code index set, it is also useful to consider codes defined on a superset the index set. Let $K \supseteq I$ be a superset of I. The *protracted* code $\mathcal{C}_{\uparrow K}$ is defined as the code of largest dimension such that the projection of $\mathcal{C}_{\uparrow K}$ onto I is precisely $\mathcal{C}$. Specifically, the dimension of $\mathcal{C}_{\uparrow K}$ is $k(\mathcal{C}) + |K \setminus I|$.

Suppose that $\mathcal{C}_1$ and $\mathcal{C}_2$ are two codes over $\mathbb{F}_q$ defined on the same index set I. The intersection $\mathcal{C}_1 \cap \mathcal{C}_2$ of $\mathcal{C}_1$ and $\mathcal{C}_2$ is a linear code defined on I comprising the vectors in $\mathbb{F}_q^I$ that are contained in both $\mathcal{C}_1$ and $\mathcal{C}_2$.

Finally, suppose that $\mathcal{C}_a$ and $\mathcal{C}_b$ are two codes defined on the disjoint index sets J_a and J_b, respectively. The Cartesian product $\mathcal{C} = \mathcal{C}_a \times \mathcal{C}_b$ is the code defined on the index set $J_{a,b} = \{J_a, J_b\}$ such that $\mathcal{C}_{|J_a} = \mathcal{C}_{J_a} = \mathcal{C}_a$ and $\mathcal{C}_{|J_b} = \mathcal{C}_{J_b} = \mathcal{C}_b$. Equivalently, in terms of protracted codes, it is readily verified that $\mathcal{C}_a \times \mathcal{C}_b = \mathcal{C}_{a,\uparrow J_{a,b}} \cap \mathcal{C}_{b,\uparrow J_{a,b}}$.

C. Generalized Extension Codes

Let $\mathcal{C}$ be a linear code over $\mathbb{F}_q$ defined on the index set I. Let $J \subseteq I$ be some subset of I and let

$$\boldsymbol{\beta} = (\beta_j \neq 0 \in \mathbb{F}_q, j \in J) \tag{5}$$

be a vector of nonzero elements of $\mathbb{F}_q$. A *generalized extension* of $\mathcal{C}$ is formed by adding a q-ary parity-check on the subset of codeword coordinates indexed by J to $\mathcal{C}$ (i.e., a q-ary *partial* parity symbol, rather than the parity-check on all codeword coordinates used to define classical code extensions [36]). The generalized extension code $\tilde{\mathcal{C}}$ is defined on the index set $\tilde{I} = I \cup \{p\}$ such that if $\boldsymbol{c} = (c_i, i \in I) \in \mathcal{C}$, then $\tilde{\boldsymbol{c}} = (\tilde{c}_i, i \in \tilde{I}) \in \tilde{\mathcal{C}}$ where $\tilde{c}_i = c_i$ if $i \in I$ and

$$\tilde{c}_p = \sum_{j \in J} \beta_j c_j. \tag{6}$$

The length and dimension of $\tilde{\mathcal{C}}$ are $n(\tilde{\mathcal{C}}) = n(\mathcal{C}) + 1$ and $k(\tilde{\mathcal{C}}) = k(\mathcal{C})$, respectively, and the minimum distance of $\tilde{\mathcal{C}}$ satisfies $d(\tilde{\mathcal{C}}) \in \{d(\mathcal{C}), d(\mathcal{C}) + 1\}$. Note that if $J = I$ and $\beta_j = 1$ for all $j \in J$, then $\tilde{\mathcal{C}}$ is simply a classically defined extended code [36]. More generally, a *degree-g generalized extension* of $\mathcal{C}$ is formed by adding g q-ary partial parity symbols to $\mathcal{C}$ and is defined on the index set $I \cup \{p_1, p_2, \ldots, p_g\}$. The jth partial parity symbol c_{p_j} in such an extension is defined as a partial parity on some subset of $I \cup \{p_1, \ldots, p_{j-1}\}$.

D. Graph Theory

A *graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{H})$ consists of the following:

- a finite nonempty set of vertices $\mathcal{V}$;
- a set of edges $\mathcal{E}$, which is some subset of the pairs $\{\{u, v\} : u, v \in \mathcal{V}, u \neq v\}$;
- a set of *half-edges* $\mathcal{H}$, which is any subset of $\mathcal{V}$.

Note that the graphs considered in this work do not contain parallel edges. It is nonstandard to define graphs with half-edges; however, as will be demonstrated in Section II-E, half-edges are useful in the context of graphical models for codes. A walk of length n in $\mathcal{G}$ is a sequence of vertices $v_1, v_2, \ldots, v_n, v_{n+1}$ in $\mathcal{V}$ such that $\{v_i, v_{i+1}\} \in \mathcal{E}$ for all $i \in \{1, \ldots, n\}$. A path is a walk on distinct vertices while a *cycle* of length n is a walk such that v_1 through v_n are distinct and $v_1 = v_{n+1}$. Cycles of length n are often denoted n-cycles. Two vertices $u, v \in \mathcal{V}$ are *adjacent* if a single edge $\{u, v\} \in \mathcal{E}$ connects u to v. A graph is *connected* if any two of its vertices are linked by a walk. A *forest* is a graph containing no cycles (i.e., a *cycle-free* graph) and a *tree* is a connected forest. A *cut* in a connected graph $\mathcal{G}$ is some subset of edges $\mathcal{X} \subseteq \mathcal{E}$ the removal of which yields a disconnected graph. Cuts thus partition the vertex set $\mathcal{V}$. Finally, a graph is bipartite if its vertex set can be partitioned $\mathcal{V} = \mathcal{U} \cup \mathcal{W}, \mathcal{U} \cap \mathcal{W} = \emptyset$, such that any edge in $\mathcal{E}$ joins a vertex in $\mathcal{U}$ to one in $\mathcal{W}$.

E. Graphical Models of Codes

Graphical models for codes have been described by a number of different authors using a wide variety of notation (e.g., [6]–[11]). This work uses the notation described below, which was established by Forney in his *codes on graphs* papers [10], [35].

A *linear behavioral realization* of a linear code $\mathcal{C} \subseteq \mathbb{F}_q^I$ comprises three sets indexed by I, I_S, and I_C, respectively, the latter two of which are disjoint and unrelated to I as follows:

- a set of *visible* (or symbol) variables $\{V_i, i \in I\}$ corresponding to the codeword coordinates[2] with alphabets $\mathbb{F}_q$;
- a set of *hidden* (or state) variables $\{S_i, i \in I_S\}$ with alphabets $\{\mathbb{F}_q^{T_i}, i \in I_S\}$;
- a set of linear local constraint codes $\{\mathcal{C}_i, i \in I_C\}$.

Each visible variable is q-ary while the hidden variable S_i with alphabet $\mathbb{F}_q^{T_i}$ is $q^{|T_i|}$-ary. The hidden variable alphabet index sets $\{T_i, i \in I_S\}$ are disjoint and unrelated to I. Each local constraint code $\mathcal{C}_i$ involves a certain subset of the visible $I_V(i) \subseteq I$ and hidden $I_S(i) \subseteq I_S$ variables and defines a subspace of the local configuration space

$$\mathcal{C}_i \subseteq \left(\prod_{j \in I_V(i)} \mathbb{F}_q \right) \left(\prod_{j \in I_S(i)} \mathbb{F}_q^{T_j} \right). \tag{7}$$

Each local constraint $\mathcal{C}_i$ is a linear code over $\mathbb{F}_q$ defined on the local index set

$$I_L(i) = I_V(i) \cup \left(\bigcup_{j \in I_S(i)} T_j \right) \tag{8}$$

with well-defined block length

$$n(\mathcal{C}_i) = |I_V(i)| + \sum_{j \in I_S(i)} |T_j| \tag{9}$$

and dimension $k(\mathcal{C}_i) = \dim \mathcal{C}_i$. Local constraints that involve only hidden variables are *internal* constraints while those involving visible variables are *interface* constraints. The full behavior of the realization is the set $\mathfrak{B}$ of all visible and hidden

[2]Observe that this definition is slightly different than that proposed in [35], which permitted the use of q^r-ary visible variables corresponding to r codeword coordinates. By appropriately introducing equality constraints and q-ary hidden variables, it can be seen that these two definitions are essentially equivalent.

variable configurations, which simultaneously satisfy all local constraint codes

$$\mathfrak{B} \subseteq \left(\prod_{i\in I}\mathbb{F}_q\right)\left(\prod_{j\in I_S}\mathbb{F}_q^{T_j}\right) = \mathbb{F}_q^I\left(\prod_{j\in I_S}\mathbb{F}_q^{T_j}\right). \quad (10)$$

The projection of the linear code $\mathfrak{B}$ onto I is precisely $\mathcal{C}$.[3]

Forney demonstrated in [10] that it is sufficient to consider only those realizations in which all visible variables are involved in a single local constraint and all hidden variables are involved in two local constraints. Furthermore, it is sufficient to consider only those realizations in which no two hidden variables are involved in the same pair of local constraints. Such *normal* realizations have a natural graphical representation in which local constraints are represented by vertices, visible variables by half-edges, and hidden variables by edges. The half-edge corresponding to the visible variable V_i is incident on the vertex corresponding to the single local constraint that involves V_i. The edge corresponding to the hidden variable S_j is incident on the vertices corresponding to the two local constraints that involve S_j. The notation $\mathcal{G}_\mathcal{C}$ and term *graphical model* are used throughout this work to denote both a normal realization of a code $\mathcal{C}$ and its associated graphical representation.

It is assumed throughout that the graphical models considered are connected. Equivalently, it is assumed throughout that the codes studied cannot be decomposed into Cartesian products of shorter codes [10]. Note that this restriction will apply only to the global code considered and not to the local constraints in a given graphical model.

Finally, because different local constraints are defined on different index sets, care must be taken in defining the intersection of local constraints. Let $\mathcal{G}_\mathcal{C}$ be a graphical model for a code $\mathcal{C}$ defined on the index set I, and let $\mathcal{C}_i$ and $\mathcal{C}_j$ be two local constraints in $\mathcal{G}_\mathcal{C}$ defined on the local index sets $I_L(i)$ and $I_L(j)$, respectively. Denote by $I_L(i,j) = I_L(i) \cup I_L(j)$ the union of the respective local index sets. The intersection of $\mathcal{C}_i$ and $\mathcal{C}_j$ protracted to $I_L(i,j)$

$$\mathcal{C}_{i,\uparrow I_L(i,j)} \cap \mathcal{C}_{j,\uparrow I_L(i,j)} \quad (11)$$

is well defined because $\mathcal{C}_{i,\uparrow I_L(i,j)}$ and $\mathcal{C}_{j,\uparrow I_L(i,j)}$ are defined on a common index set. When it is clear in context, the notation $\mathcal{C}_i \cap \mathcal{C}_j$ is used in place of (11) for brevity's sake.

F. Tanner Graphs and Generalized Tanner Graphs

The term *Tanner graph* has been used to describe different classes of graphical models by different authors. Tanner graphs denote those graphical models corresponding to parity-check matrices in this work. Specifically, let $H_\mathcal{C}$ be an $r_H \times n(\mathcal{C})$ parity-check matrix for the code $\mathcal{C}$ over $\mathbb{F}_q$ defined on the index set I. The Tanner graph corresponding to $H_\mathcal{C}$ contains $r_H + n(\mathcal{C})$ local constraints of which $n(\mathcal{C})$ are interface repetition constraints, one corresponding to each codeword coordinate, and r_H are internal q-ary single parity-check constraints, one corresponding to each row of $H_\mathcal{C}$. An edge (hidden variable) connects a repetition constraint $\mathcal{C}_i$ to a single parity-check constraint $\mathcal{C}_j$ if and only if the codeword coordinate corresponding to $\mathcal{C}_i$ is involved in the single parity-check equation defined by the row corresponding to $\mathcal{C}_j$. A Tanner graph for $\mathcal{C}$ is *redundant* if it corresponds to a redundant parity-check matrix. A *degree-g generalized Tanner graph* for $\mathcal{C}$ is simply a Tanner graph corresponding to some degree-g generalized extension of $\mathcal{C}$ in which the visible variables corresponding to the partial parity symbols have been removed. Generalized Tanner graphs have been studied previously in the literature under the rubric of generalized parity-check matrices [37], [38].

[3]Note that it assumed throughout this work that if $\boldsymbol{b} \in \mathfrak{B}$ is such that $\boldsymbol{b}_{|I} = 0$, then $\boldsymbol{b} = 0$.

III. Complexity Measure for Graphical Models

A. q^m-ary Graphical Models

This work introduces the term *q^m-ary graphical model* to denote a normal realization of a linear code $\mathcal{C}$ over $\mathbb{F}_q$ that satisfies the following constraints:

- the alphabet index size of every hidden variable $S_i, i \in I_S$, satisfies $|T_i| \leq m$;
- every local constraint $\mathcal{C}_i, i \in I_C$, either satisfies

$$\min(k(\mathcal{C}_i), n(\mathcal{C}_i) - k(\mathcal{C}_i)) \leq m \quad (12)$$

  or can be decomposed as a Cartesian product of codes, each of which satisfies this condition.

The complexity measure m simultaneously captures a cyclic graphical model analog of the familiar notions of state and branch complexity for trellises [30]. From the above definition, it is clear that Tanner graphs and generalized Tanner graphs for codes over $\mathbb{F}_q$ are q-ary graphical models. The efficacy of this complexity measure is discussed further in Section V.

B. Properties of q^m-ary Graphical Models

The following three properties of q^m-ary graphical models will be used in the proof of Theorem 3 in Section IV. These properties are defined in detail in Section B of the Appendix (which, in turn, uses notation established in Section A of the Appendix).

1) *Internal Local Constraint Involvement Property:* Any hidden variable in a q^m-ary graphical model can be made to be incident (on at least one end) on an *internal* local constraint $\mathcal{C}_i$, which satisfies $n(\mathcal{C}_i) - k(\mathcal{C}_i) \leq m$ without fundamentally altering the complexity or cyclic topology of that graphical model.
2) *Internal Local Constraint Removal Property:* The removal of an internal local constraint from a q^m-ary graphical model results in a q^m-ary graphical model for a new code defined on the same index set.
3) *Internal Local Constraint Redefinition Property:* Any internal local constraint $\mathcal{C}_i$ in a q^m-ary graphical model satisfying $n(\mathcal{C}_i) - k(\mathcal{C}_i) = m' \leq m$ can be equivalently represented by $m'q$-ary single parity-check equations over the visible variable index set.

These properties are particularly useful in concert. Specifically, let $\mathcal{G}_{\mathcal{C}}$ be a q^m-ary graphical model for the linear code $\mathcal{C}$ over $\mathbb{F}_q$ defined on an index set I. Suppose that the internal constraint $\mathcal{C}_r$ satisfying $n(\mathcal{C}_r) - k(\mathcal{C}_r) = m' \leq m$ is removed from $\mathcal{G}_{\mathcal{C}}$ resulting in the new code $\mathcal{C}^{\backslash r}$. Denote by $\mathcal{C}_r^{(1)}, \ldots, \mathcal{C}_r^{(m')}$ the set of m' q-ary single parity-check equations that result when $\mathcal{C}_r$ is redefined over I. A vector in $\mathbb{F}_q^I$ is a codeword in $\mathcal{C}$ if and only if it is contained in $\mathcal{C}^{\backslash r}$ and satisfies each of these m' single parity-check equations so that

$$\mathcal{C} = \mathcal{C}^{\backslash r} \cap \mathcal{C}_r^{(1)} \cap \cdots \cap \mathcal{C}_r^{(m')}. \tag{13}$$

The internal local constraint redefinition property affords the useful notion of *local constraint equivalence*. Suppose that $\mathcal{G}_{\mathcal{C}}$ and $\tilde{\mathcal{G}}_{\mathcal{C}}$ are two distinct graphical models for the code $\mathcal{C}$ defined on the index set I. Let $\mathcal{C}_i$ and $\mathcal{C}_j$ be local constraints in $\mathcal{G}_{\mathcal{C}}$ and $\tilde{\mathcal{G}}_{\mathcal{C}}$, respectively. Denote by $\mathcal{C}(i)(\mathcal{C}(j))$ the code that is formed by the intersection of the single parity-check equations that result when $\mathcal{C}_i(\mathcal{C}_j)$ is redefined over I. The local constraints $\mathcal{C}_i$ and $\mathcal{C}_j$ are said to be *equivalent* (denoted $\mathcal{C}_i \triangleq \mathcal{C}_j$ throughout) if and only if $\mathcal{C}(i) = \mathcal{C}(j)$. That is, two local constraint codes are equivalent if they impose identical constraints on the visible variable set.

C. The Minimal Tree Complexity of a Code

The minimal trellis complexity $s(\mathcal{C})$ of a linear code $\mathcal{C}$ over $\mathbb{F}_q$ is defined as the base-q logarithm of the maximum hidden variable alphabet size in its minimal (unsectionalized) trellis [39]. Considerable attention has been paid to this quantity (cf., [39]–[44]) as it is closely related to the important, and difficult, study of determining the minimum possible complexity of optimal SISO decoding of a given code. This work introduces the minimal tree complexity of a linear code as a generalization of minimal trellis complexity to arbitrary cycle-free graphical model topologies.

Definition 1: The *minimal tree complexity* of a linear code $\mathcal{C}$ over $\mathbb{F}_q$ is the smallest integer $t(\mathcal{C})$ such that there exists a cycle-free $q^{t(\mathcal{C})}$-ary graphical model for $\mathcal{C}$.

Much as $s(\mathcal{C}) = s(\mathcal{C}^{\perp})$, the minimal tree complexity of a code $\mathcal{C}$ is equal to that of its dual.

Proposition 1: Let $\mathcal{C}$ be a linear code over $\mathbb{F}_q$ with dual $\mathcal{C}^{\perp}$. Then

$$t(\mathcal{C}) = t(\mathcal{C}^{\perp}). \tag{14}$$

Proof: The dualizing procedure described by Forney [10] can be applied to a $q^{t(\mathcal{C})}$-ary graphical model for $\mathcal{C}$ in order to obtain a graphical model for $\mathcal{C}^{\perp}$, which is readily shown to be $q^{t(\mathcal{C})}$-ary. □

The following propositions establish upper and lower bounds on the tree complexity of linear codes.

Proposition 2: The tree complexity $t(\mathcal{C})$ of a linear code $\mathcal{C}$ is upper-bounded by its minimal trellis complexity $s(\mathcal{C})$ and thus all known upper bounds on $s(\mathcal{C})$ extend to $t(\mathcal{C})$.

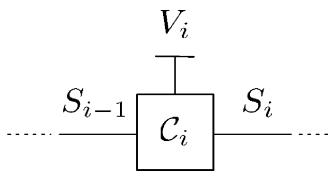

Fig. 1. The $q^{s(\mathcal{C})}$-ary graphical model representation of a trellis section.

Proof: Consider the section of a minimal trellis for $\mathcal{C}$ illustrated in Fig. 1. The hidden (state) variables have alphabet index sizes $|T_i| \leq s(\mathcal{C})$ and $|T_{i-1}| \leq s(\mathcal{C})$, respectively. Because $|T_i|$ and $|T_{i-1}|$ differ by at most 1 in a minimal trellis, and because

$$k(\mathcal{C}_i) + (n(\mathcal{C}_i) - k(\mathcal{C}_i)) = n(\mathcal{C}_i) = |T_i| + |T_{i-1}| + 1 \tag{15}$$

it is readily shown that for all i

$$\begin{aligned} \min(k(\mathcal{C}_i), n(\mathcal{C}_i) - k(\mathcal{C}_i)) &\leq \left\lfloor \frac{|T_i| + |T_{i-1}| + 1}{2} \right\rfloor \\ &\leq s(\mathcal{C}) \end{aligned} \tag{16}$$

completing the proof. □

Proposition 3: Let $\mathcal{C}$ be an $[n, k, d]$ linear code over $\mathbb{F}_q$ defined on the index set I. Denote by $k_{\max}(i; \mathcal{C})$ the maximum dimension of any subcode of $\mathcal{C}$ with support size i (cf., [44]). The tree complexity $t(\mathcal{C})$ of $\mathcal{C}$ is lower-bounded by

$$t(\mathcal{C}) \geq k - \min_{i \in [1,n]} \{k_{\max}(i; \mathcal{C}) + k_{\max}(n - i; \mathcal{C})\}. \tag{17}$$

Proof: Let S_j be a hidden variable in a $q^{t(\mathcal{C})}$ cycle-free graphical model $\mathcal{G}_{\mathcal{C}}$ for $\mathcal{C}$. Because $\mathcal{G}_{\mathcal{C}}$ is cycle-free, the edge corresponding to S_j constitutes a cut-set in $\mathcal{G}_{\mathcal{C}}$ that partitions the visible variable index set I into the disjoint subsets $J_1 \subsetneq I$ and $J_2 \subsetneq I$. Construct a two-section trellis for $\mathcal{C}$ with the sections corresponding to the visible variables indexed by J_1 and J_2, respectively. Wiberg's CSB [6] in conjunction with a result due to Forney [45] can then be used to lower-bound the alphabet index size of S_j by

$$|T_j| \geq k - k(\mathcal{C}_{J_1}) - k(\mathcal{C}_{J_2}). \tag{18}$$

The desired bound is obtained by noting that

$$k(\mathcal{C}_{J_1}) \leq k_{\max}(|J_1|; \mathcal{C}) \tag{19}$$

and

$$k(\mathcal{C}_{J_2}) \leq k_{\max}(n - |J_1|; \mathcal{C}). \tag{20}$$

□

The lower bound established by Proposition 3 is simply an extension of the dimension-length profile (DLP) bound (cf., [39] and [44]) (which, in turn, is an improvement of Muder's bound [40]). However, not all lower bounds on $s(\mathcal{C})$ readily extend to $t(\mathcal{C})$. For example, it is not clear how to extend Lafourcade and Vardy's results [39], [46] to bounds for $t(\mathcal{C})$ due to the difficulty of considering all possible cycle-free topologies rather than only the line graphs implied by trellises.

An important question for future study is, therefore, the development of tight lower bounds on $t(\mathcal{C})$. An example of a code for which $t(\mathcal{C})$ is *strictly smaller* than $s(\mathcal{C})$ was provided by Forney in [35]. Specifically, let $\mathcal{C}_{[9,2,6]}$ be the $[9, 2, 6]$ binary linear code generated by

$$G_{[9,2,6]} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}. \tag{21}$$

The minimal (unsectionalized) trellis complexity of this code can be shown to be

$$s(\mathcal{C}_{[9,2,6]}) = 2 \tag{22}$$

whereas Forney illustrated a cycle-free Tanner graph for $\mathcal{C}_{[9,2,6]}$ so that

$$t(\mathcal{C}_{[9,2,6]}) = 1. \tag{23}$$

The following lemma concerning minimal tree complexity will be used in the proof of Theorem 3 in Section IV. The proof of Lemma 1 is detailed further by example in Section C of the Appendix.

Lemma 1: Let $\mathcal{C}$ and $\mathcal{C}^{\mathrm{SPC}}$ be linear codes over $\mathbb{F}_q$ defined on the index set I such that $\mathcal{C}^{\mathrm{SPC}}$ comprises a q-ary single parity-check code on some subset of the index set $J \subseteq I$. Define by $\tilde{\mathcal{C}}$ the intersection of $\mathcal{C}$ and $\mathcal{C}^{\mathrm{SPC}}$

$$\tilde{\mathcal{C}} = \mathcal{C} \cap \mathcal{C}^{\mathrm{SPC}}. \tag{24}$$

The minimal tree complexity of $\tilde{\mathcal{C}}$ is upper-bounded by

$$t(\tilde{\mathcal{C}}) \leq t(\mathcal{C}) + 1. \tag{25}$$

Proof: The result is proved by explicit construction of a $q^{t(\mathcal{C})+1}$-ary cycle-free graphical model for $\tilde{\mathcal{C}}$ as follows. Let $\mathcal{G}_{\mathcal{C}}$ be some $q^{t(\mathcal{C})}$-ary cycle-free graphical model for $\mathcal{C}$ and let $\mathcal{T}$ be a minimal connected subtree of $\mathcal{G}_{\mathcal{C}}$ containing the set of $|J|$ interface constraints, which involve the visible variables in J. Denote by $I_S(\mathcal{T}) \subseteq I_S$ and $I_C(\mathcal{T}) \subseteq I_C$ the subset of hidden variables and local constraints, respectively, contained in $\mathcal{T}$. Choose some local constraint vertex $\mathcal{C}_\Lambda, \Lambda \in I_C(\mathcal{T})$, as a root for $\mathcal{T}$. Observe that the choice of $\mathcal{C}_\Lambda$, while arbitrary, induces a directionality in $\mathcal{T}$: *downstream* toward the root vertex or *upstream* away from the root vertex. For every $S_i, i \in I_S(\mathcal{T})$, denote by $J_{i,\uparrow} \subseteq J$ the subset of visible variables in J, which are upstream from that hidden variable edge.

A $q^{t(\mathcal{C})+1}$-ary graphical model for $\tilde{\mathcal{C}}$ is then constructed from $\mathcal{G}_{\mathcal{C}}$ by updating each hidden variable $S_i, i \in I_S(\mathcal{T})$, to also contain the q-ary partial parity of the upstream visible variables in $J_{i,\uparrow} \subseteq J$. The local constraints $\mathcal{C}_j, j \in I_C(\mathcal{T}) \setminus \Lambda$, are updated accordingly. Finally, $\mathcal{C}_\Lambda$ is updated to enforce the q-ary single parity constraint defined by $\mathcal{C}^{\mathrm{SPC}}$. This updating procedure increases the alphabet size of each hidden variable $S_i, i \in I_S(\mathcal{T})$, by at most one and adds at most one single parity-check (or repetition) constraint to the definition of each $\mathcal{C}_j, j \in I_C(\mathcal{T})$, and the resulting cycle-free graphical model is thus at most $q^{t(\mathcal{C})+1}$-ary. □

IV. The Tradeoff Between Cyclic Topology and Complexity

A. The Cut-Set and Square-Root Bounds

Wiberg's CSB [5], [6] is stated below without proof in the language of Section II.

Theorem 1 (CSB): Let $\mathcal{C}$ be a linear code over $\mathbb{F}_q$ defined on the index set I. Let $\mathcal{G}_{\mathcal{C}}$ be a graphical model for $\mathcal{C}$ containing a cut $\mathcal{X}$ corresponding to the hidden variables $S_i, i \in I_S(\mathcal{X})$, which partitions the index set into $J_1 \subsetneq I$ and $J_2 \subsetneq I$. Let the base-q logarithm of the midpoint hidden variable alphabet size of the minimal two-section trellis for $\mathcal{C}$ on the two-section time axis $\{J_1, J_2\}$ be $s_{\mathcal{X},\min}$. The sum of the base-q logarithm of the hidden variable alphabet sizes corresponding to the cut $\mathcal{X}$ is lower-bounded by

$$\sum_{i \in I_S(\mathcal{X})} |T_i| \geq s_{\mathcal{X},\min}. \tag{26}$$

The CSB provides insight into the tradeoff between cyclic topology and complexity in graphical models for codes and it is natural to explore its power to quantify this tradeoff. Two questions that arise for a given linear code $\mathcal{C}$ over $\mathbb{F}_q$ in such an exploration are as follows.

1) For a given complexity m, how many cycles must be contained in a q^m-ary graphical model for $\mathcal{C}$?
2) For a given number of cycles N, what is the smallest m such that a q^m-ary model containing N cycles for $\mathcal{C}$ can exist?

For a fixed cyclic topology, the CSB can be simultaneously applied to all cuts yielding a linear programming lower bound on the hidden variable alphabet sizes [5]. For the special case of a single-cycle graphical model (i.e., a tail-biting trellis), this technique yields a simple solution [31].

Theorem 2 (Square-Root Bound): Let $\mathcal{C}$ be a linear code over $\mathbb{F}_q$ of even length and let $s_{\mathrm{mid},\min}(\mathcal{C})$ be the base-q logarithm of the minimum possible hidden variable alphabet size of a conventional trellis for $\mathcal{C}$ at its midpoint over all coordinate orderings. The base-q logarithm of the minimum possible hidden variable alphabet size $s_{\mathrm{TB}}(\mathcal{C})$ of a tail-biting trellis for $\mathcal{C}$ is lower-bounded by

$$s_{\mathrm{TB}}(\mathcal{C}) \geq s_{\mathrm{mid},\min}(\mathcal{C})/2. \tag{27}$$

The square-root bound can thus be used to answer the questions posed above for a specific class of single-cycle graphical models. For topologies richer than a single cycle, however, the aforementioned linear programming technique quickly becomes intractable. Specifically, there are

$$2^{n(\mathcal{C})-1} - 1 \tag{28}$$

ways to partition a size $n(\mathcal{C})$ visible variable index set into two nonempty, disjoint, subsets. The number of cuts to be considered by the linear programming technique for a given cyclic topology

thus grows exponentially with block length and a different minimal two-stage trellis must be constructed to bound the size of each of those cuts.

B. Forest-Inducing Cuts

Recall that a cut in a graph $\mathcal{G}$ is some subset of the edges $\mathcal{X} \subseteq \mathcal{E}$ the removal of which yields a disconnected graph. A cut is thus defined without regard to the cyclic topology of the disconnected components, which remain after its removal. To provide a characterization of the tradeoff between cyclic topology and complexity that is more precise than that provided by the CSB alone, this work focuses on a specific type of cut which is defined below. Two useful properties of such cuts are established by Propositions 4 and 5.

Definition 2: Let $\mathcal{G}$ be a connected graph. A *forest-inducing cut*[4] is some subset of edges $\mathcal{X}_F \subseteq \mathcal{E}$ the removal of which yields a forest with precisely two components.

Proposition 4: Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{H})$ be a connected graph. The size X_F of *any* forest-inducing cut $\mathcal{X}_F$ in $\mathcal{G}$ is precisely

$$X_F = |\mathcal{E}| - |\mathcal{V}| + 2. \tag{29}$$

Proof: It is well known that a connected graph is a tree if and only if (cf., [49])

$$|\mathcal{E}| = |\mathcal{V}| - 1. \tag{30}$$

Similarly, a graph composed of two cycle-free components satisfies

$$|\mathcal{E}| = |\mathcal{V}| - 2. \tag{31}$$

The result then follows from the observation that the size of a forest-inducing cut is the number of edges, which must be removed to satisfy (31). □

Proposition 5: Let $\mathcal{G}$ be a connected graph with forest-inducing cut size X_F. The number of cycles $N_{\mathcal{G}}$ in $\mathcal{G}$ is lower-bounded by

$$N_{\mathcal{G}} \geq \binom{X_F}{2}. \tag{32}$$

Proof: Let the removal of a forest-inducing cut $\mathcal{X}_F$ in the connected graph $\mathcal{G}$ yield the cycle-free components $\mathcal{G}_1$ and $\mathcal{G}_2$ and let $e_i, e_j \in \mathcal{X}_F$ with $e_i \neq e_j$. Because $\mathcal{G}_1(\mathcal{G}_2)$ is a tree, there is a unique path in $\mathcal{G}_1(\mathcal{G}_2)$ connecting e_i and e_j. There is thus a unique cycle in $\mathcal{G}$ corresponding to the edge pair $\{e_i, e_j\}$. There are $\binom{X_F}{2}$ such distinct edge pairs, which yield the lower bound. Note that this is a lower bound because for certain graphs, there can exist cycles that contain more than two edges from a forest-inducing cut. □

Note that the forest-inducing cut size of a graph $\mathcal{G}$ provides a lower bound on the number of cycles in $\mathcal{G}$, in contrast to the upper bound provided by the more familiar measure of *cycle rank*. Specifically, the cycle rank c of a connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{H})$ is equal to

$$c = |\mathcal{E}| - |\mathcal{V}| + 1 = X_F - 1 \tag{33}$$

and the number of cycles and unions of disjoint cycles in $\mathcal{G}$ is upper-bounded by 2^c (cf., [49]).

C. The Forest-Inducing Cut-Set Bound

With forest-inducing cuts defined, the required properties of q^m-ary graphical models described, and Lemma 1 established, the main result concerning the tradeoff between cyclic topology and graphical model complexity can now be stated and proved.

Theorem 3: Let $\mathcal{C}$ be a linear code over $\mathbb{F}_q$ defined on the index set I and suppose that $\mathcal{G}_{\mathcal{C}}$ is a q^m-ary graphical model for $\mathcal{C}$ with forest-inducing cut size X_F. The minimal tree complexity of $\mathcal{C}$ is upper-bounded by

$$t(\mathcal{C}) \leq mX_F. \tag{34}$$

Proof: The result is proved by induction on X_F. Let $X_F = 1$ and suppose that $e \in \mathcal{X}_F$ is the sole edge in some forest-inducing cut $\mathcal{X}_F$ in $\mathcal{G}_{\mathcal{C}}$. Because the removal of e partitions $\mathcal{G}_{\mathcal{C}}$ into disconnected cycle-free components, $\mathcal{G}_{\mathcal{C}}$ must be cycle-free and $t(\mathcal{C}) \leq m$ by construction.

Now suppose that $X_F = x > 1$ and let $e \in \mathcal{X}_F$ be an edge in some forest-inducing cut $\mathcal{X}_F$ in $\mathcal{G}_{\mathcal{C}}$. By the first q^m-ary graphical model property of Section III-B, e is (after suitable transformations) incident on some internal local constraint $\mathcal{C}_i$ satisfying $n(\mathcal{C}_i) - k(\mathcal{C}_i) = m' \leq m$. Denote by $\mathcal{G}_{\mathcal{C}^{\setminus i}}$ the q^m-ary graphical model that results when $\mathcal{C}_i$ is removed from $\mathcal{G}_{\mathcal{C}}$, and by $\mathcal{C}^{\setminus i}$ the corresponding code over I. The forest-inducing cut size of $\mathcal{G}_{\mathcal{C}^{\setminus i}}$ is at most $x - 1$ because the removal of $\mathcal{C}_i$ from $\mathcal{G}_{\mathcal{C}}$ results in the removal a single vertex and at least two edges. By the induction hypothesis, the minimal tree complexity of $\mathcal{C}^{\setminus i}$ is upper-bounded by

$$t(\mathcal{C}^{\setminus i}) \leq m(x-1). \tag{35}$$

From the discussion of Section III-B, it is clear that $\mathcal{C}_i$ can be redefined as $m' \leq m$ single parity-check equations, $\mathcal{C}_i^{(j)}$ for $j \in [1, m']$, over $\mathbb{F}_q$ on I such that

$$\mathcal{C} = \mathcal{C}^{\setminus i} \cap \mathcal{C}_i^{(1)} \cap \cdots \cap \mathcal{C}_i^{(m')}. \tag{36}$$

It follows from Lemma 1 that

$$t(\mathcal{C}) \leq t(\mathcal{C}^{\setminus i}) + m' \leq mx \tag{37}$$

completing the proof. □

An immediate corollary to Theorem 3 results when Proposition 5 is applied in conjunction with Theorem 3:

Corollary 1: Let $\mathcal{C}$ be a linear code over $\mathbb{F}_q$ with minimal tree complexity $t(\mathcal{C})$. The number of cycles N_m in any q^m-ary graphical model for $\mathcal{C}$ is lower-bounded by

$$N_m \geq \binom{\lfloor t(\mathcal{C})/m \rfloor}{2}. \tag{38}$$

[4]Note that such cuts were previously described as "tree-inducing" in [47] and [48]. In this work, the terminology "forest-inducing" has been adopted for the cuts and the resulting FI-CSB to emphasize that the graph resulting from the removal of such a cut is disconnected.

D. Interpretation of the FI-CSB

Provided $t(\mathcal{C})$ is known or can be lower-bounded, the forest-inducing cut-set bound (FI-CSB) (and more specifically Corollary 1) can be used to answer the questions posed in Section IV-A. The FI-CSB is further discussed below.

1) The FI-CSB and the CSB: On the surface, the FI-CSB and the CSB are similar in statement; however, there are three important differences between the two. First, the CSB does not explicitly address the complexity of the local constraints on either side of a given cut. Forney provided a number of illustrative examples in [35] that stress the importance of characterizing graphical model complexity in terms of both hidden variable size and local constraint complexity. Second, the CSB does not explicitly address the cyclic topology of the graphical model that results when the edges in a cut are removed. The removal of a forest-inducing cut results in two cycle-free disconnected components and the size of a forest-inducing cut can thus be used to make statements about the complexity of optimal SISO decoding using variable conditioning in a cyclic graphical model (cf., [10] and [50]–[54]). Finally, and most fundamentally, the FI-CSB addresses the aforementioned intractability of applying the CSB to graphical models with rich cyclic topologies.

2) The FI-CSB and the Square-Root Bound: Theorem 3 can be used to make a statement similar to Theorem 2, which is valid for all graphical models containing a single cycle.

Corollary 2: Let $\mathcal{C}$ be a linear code over $\mathbb{F}_q$ with minimal tree complexity $t(\mathcal{C})$ and let m_2 be the smallest integer such that there exists a q^{m_2}-ary graphical model for $\mathcal{C}$, which contains at most one cycle. Then

$$m_2 \geq t(\mathcal{C})/2. \tag{39}$$

More generally, Theorem 3 can be used to establish the following generalization of the square-root bound to graphical models with arbitrary cyclic topologies.

Corollary 3: Let $\mathcal{C}$ be a linear code over $\mathbb{F}_q$ with minimal tree complexity $t(\mathcal{C})$. For some positive integer r, let m_r be the smallest integer such that there exists a q^{m_r}-ary graphical model for $\mathcal{C}$, which contains at most $\binom{r}{2}$ cycles. Then

$$m_r \geq t(\mathcal{C})/r. \tag{40}$$

The desired generalization of the square-root bound is obtained by noting that m_r measures the logarithm of decoding complexity in Corollary 3: an rth-root complexity reduction with respect to the minimal tree complexity requires the introduction of at least $r(r-1)/2$ cycles.

There are few known examples of classical linear block codes that meet the square-root bound with equality. Shany and Be'ery proved that many RM codes cannot meet this bound under *any* bit ordering [55]. There does, however, exist a tail-biting trellis for the extended binary Golay code $\mathcal{C}_G$, which meets the square-root bound with equality so that [31]

$$s_{\text{mid,min}}(\mathcal{C}_G) = 8 \quad \text{and} \quad s_{\text{TB}}(\mathcal{C}_G) = 4. \tag{41}$$

This tail-biting trellis model *cannot*, however, be used as the basis for a new result on the minimal tree complexity of the Golay code. Calderbank *et al.*'s tail-biting trellis representation is sectionalized so that there are two codeword coordinates per trellis section and two state transitions per trellis state (see [31, Fig. 5]). While the state complexity of this tail-biting trellis is indeed 4, each trellis section is described by a length 10, dimension 5 code over $\mathbb{F}_2$ so that the corresponding graphical model is 2^5-ary. Corollary 2 can, therefore, be used to show that $t(\mathcal{C}_G)$ is at most 10. However, it is known that $t(\mathcal{C}_G) \leq 9$. The minimal bit-level conventional trellis for $\mathcal{C}_G$ contains (noncentral) state variables with alphabet size 512 and is thus a 2^9-ary graphical model [40].

3) Aymptotics of the FI-CSB: Denote by N_m the minimum number of cycles in any q^m-ary graphical model for a linear code $\mathcal{C}$ over $\mathbb{F}_q$ with minimal tree complexity $t(\mathcal{C})$. For large values of $t(\mathcal{C})/m$, the lower bound on N_m established by Corollary 1 becomes

$$N_m \geq \binom{\lfloor t(\mathcal{C})/m \rfloor}{2} \approx \frac{t(\mathcal{C})^2}{2m^2}. \tag{42}$$

The ratio of the minimal complexity of a cycle-free model for $\mathcal{C}$ to that of a q^m-ary graphical model is thus upper-bounded by

$$\frac{q^{t(\mathcal{C})}}{q^m} \lessapprox q^{m(\sqrt{2N_m}-1)}. \tag{43}$$

The FI-CSB can be used to argue that q-ary graphical models cannot support asymptotically good codes over $\mathbb{F}_q$ unless the number of cycles increases with the square of the block length. Specifically, consider a family of codes over $\mathbb{F}_q$ with increasing length and constant rate. To aid aymptotic analysis, assume that the DLP bound [44] on trellis complexity of any given code $\mathcal{C}$ in this family is tight so that

$$s(\mathcal{C}) - \left(k - \min_{i \in [1, n(\mathcal{C})]} \{ k_{\max}(i;\mathcal{C}) + k_{\max}(n(\mathcal{C})-i;\mathcal{C}) \} \right) \leq \delta \tag{44}$$

where δ is some small constant that does not depend on $n(\mathcal{C})$. Under this assumption, the difference between $t(\mathcal{C})$ and $s(\mathcal{C})$ is bounded (by Proposition 3). The FI-CSB can then be used in conjunction with Lafourcade and Vardy's lower bound on trellis complexity [46]

$$s(\mathcal{C}) \geq \left\lceil \frac{k(\mathcal{C})\,((d(\mathcal{C})-1)}{n(\mathcal{C})} \right\rceil \tag{45}$$

to show that $d(\mathcal{C}) \lessapprox X_F/R(\mathcal{C})$. To support an aymptotically good sequence of codes for which the assumption in (44) holds, X_F must thus grow linearly with $n(\mathcal{C})$ and the number of cycles must, therefore, grow with the square of $n(\mathcal{C})$. This result is consistent with the work of Etzion *et al.* [56] who proved that Tanner graphs must have cycle rank that increases linearly with block size to support asymptotically good codes. However, the focus on forest-inducing cut size rather than cycle rank in this work affords a tighter resulting bound on the number of cycles. Note that it remains open as to whether such a statement can be made for families of codes for which the DLP lower bound on trellis complexity is not tight or, more generally, for families of codes for which the difference between $s(\mathcal{C})$ and $t(\mathcal{C})$ is not bounded.

To further explore the asymptotics of the FI-CSB, consider a code of particular practical interest: the binary image $\mathcal{C}_{\text{RS}|\mathbb{F}_2}$

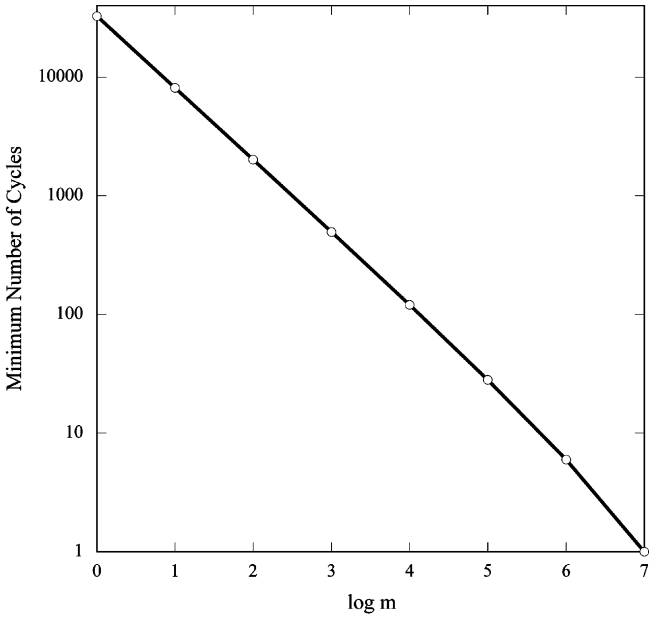

Fig. 2. Minimum number of cycles required for 2^m-ary graphical models of the binary image of the $[255, 223, 33]$ Reed–Solomon code.

of the $[255, 223, 33]$ Reed–Solomon code $\mathcal{C}_{\mathrm{RS}}$. Because $\mathcal{C}_{\mathrm{RS}}$ is maximum distance separable, a reasonable estimate for the minimal tree complexity of this code is obtained from Wolf's bound [57]

$$t(\mathcal{C}_{\mathrm{RS}|\mathbb{F}_2}) \approx 8(n(\mathcal{C}_{\mathrm{RS}}) - k(\mathcal{C}_{\mathrm{RS}})) = 256. \tag{46}$$

Fig. 2 plots N_m as a function of m for $\mathcal{C}_{\mathrm{RS}|\mathbb{F}_2}$ assuming (46). Note that because the complexity of the decoding algorithms implied by 2^m-ary graphical models grow roughly as $2^m, \log m$ is roughly a $\log \log$ decoding complexity measure.

V. On Complexity Measures and Generalizations of the Forest-Inducing Cut-Set Bound

A. Proper Complexity Measures

Recall that the aim of graphical model extraction is the obtention of a model that implies a decoding algorithm with desired complexity and performance characteristics. Complexity measures for graphical models are, therefore, useful inasmuch as they are indicative of the complexity of the iterative message-passing algorithms implied by those models. Formally, a graphical model complexity measure is simply a map $\mathcal{M}$ from the space of all graphical models to the set of nonnegative integers. Associated with any given graphical model complexity measure $\mathcal{M}$ is the following generalization of the minimal tree complexity for that measure.

Definition 3: The *$\mathcal{M}$-induced tree complexity* of a linear code $\mathcal{C}$ over $\mathbb{F}_q$ is the smallest integer $t_{\mathcal{M}}(\mathcal{C})$ such that there exists a cycle-free model for $\mathcal{C}$ with $\mathcal{M}$-complexity $t_{\mathcal{M}}(\mathcal{C})$.

Results on the $\mathcal{M}$-induced tree complexity are clearly germane to the question of how small the complexity of optimal SISO decoding of a given code can be.

Wiberg's CSB and the square-root bound for tail-biting trellises are statements that employ hidden variable alphabet size as a complexity measure. Forney demonstrated in [35] that it is insufficient to consider only hidden variable size, which can be viewed as a generalization of trellis state complexity, and argued that a suitable generalization of trellis branch complexity should instead be studied. To this end, the *constraint complexity* of a cycle-free graphical model was defined in [35] as the maximum dimension of any of its component local constraint codes. Constraint complexity was further studied by Kashyap who introduced the term *treewidth* to denote the tree complexity induced by this measure [58]. While the constraint complexity measure does indeed prevent local constraints from "hiding" complexity, the dimension of local constraints is a somewhat unsatisfactory proxy for decoding complexity because, unlike minimal trellis and tree complexities, the treewidth of a code and its dual need not be identical.

The complexity measure introduced in Section III-A was motivated by the desire to simultaneously capture hidden variable complexity and an indicator of local constraint complexity that is a more accurate gauge of decoding complexity than dimension alone. Specifically, the local constraint complexity measure used to define q^m-ary graphical models constitutes an albeit loose upper bound on trellis state complexity over the base field $(\mathbb{F}_q)$. There are many conceivable alternative measures of local constraint complexity: one could upper-bound the state complexity of the local constraints or even their $\mathcal{M}$-induced tree complexity for some measure $\mathcal{M}$ (thus defining tree complexity recursively). Given this range of possible proxies for local constraint decoding complexity, it is useful to consider the family of *proper* graphical model complexity measures defined below.

Definition 4: A graphical model complexity measure $\mathcal{M}$ is said to be proper if it obeys the following four properties.

- P1) A graphical model with $\mathcal{M}$-complexity m has maximum hidden variable alphabet index set size at most m.
- P2) The insertion of a degree-2 repetition constraint does not increase the $\mathcal{M}$-complexity of a model.
- P3) The removal of a local constraint does not increase the $\mathcal{M}$-complexity of a model.
- P4) The $\mathcal{M}$-induced tree complexity of the intersection of a code $\mathcal{C}$ defined on the index set I with a single parity-check constraint $\mathcal{C}^{\mathrm{SPC}}$ defined on some subset of the index set $J \subseteq I$ is upper-bounded by

$$t_{\mathcal{M}}\left(\mathcal{C} \cap \mathcal{C}^{\mathrm{SPC}}\right) \leq t_{\mathcal{M}}(\mathcal{C}) + \delta_{\mathcal{M}} \tag{47}$$

where $\delta_{\mathcal{M}} \geq 1$ is a constant that depends only on $\mathcal{M}$.

Properties P1)–P3) of proper graphical model complexity measures reflect the complexity characteristics of the message-passing algorithms implied by those models. For example, hidden variable alphabet size dictates message size, while it has been noted previously that degree-2 repetition constraints add no complexity cost to decoding algorithms [35], and thus ought not impact model complexity. Property P4) serves to bound the growth in the $\mathcal{M}$-induced tree complexity of a code as it is constructed as the successive intersection of single parity-check constraints. Note that this property is consistent with existing

measures for graphical model complexity. For example, the addition of a single parity-check constraint to a code increases its minimal trellis complexity by at most one. Furthermore, the proof of Lemma 1 can be used to show that the addition of a single parity-check constraint can increase the maximum hidden variable alphabet index set size of a cycle-free graphical model by at most one.

B. A Generalized FI-CSB

The forest-inducing cut-set bound is generalized to all proper graphical model complexity measures in this section. The Proof of Theorem 4 uses the fact that the local constraint redefinition property studied in Section III-B is not confined to the complexity measure introduced in Section III-A. Rather, as illustrated in Section B of Appendix, local constraints can be redefined as equivalent sets of single parity-check constraints on the visible variable set regardless of the chosen graphical model complexity measure.

Theorem 4: Let $\mathcal{M}$ be a proper graphical model complexity measure. Let $\mathcal{C}$ be a linear code over $\mathbb{F}_q$ defined on the index set I and suppose that $\mathcal{G}_\mathcal{C}$ is graphical model with $\mathcal{M}$-complexity m and forest-inducing cut size X_F. The minimal $\mathcal{M}$-induced tree complexity of $\mathcal{C}$ is upper-bounded by

$$t_\mathcal{M}(\mathcal{C}) \leq m((X_F - 1)\delta_\mathcal{M} + 1). \tag{48}$$

Proof: The result is proved by induction of X_F. If $X_F = 1$, then $\mathcal{G}_\mathcal{C}$ is cycle-free and (48) reduces to $t_\mathcal{M}(\mathcal{C}) \leq m$. Suppose that $X_F = x > 1$. As per the proof of Theorem 3, there exists some local constraint $\mathcal{C}_i$ satisfying $n(\mathcal{C}_i) - k(\mathcal{C}_i) = m' \leq m$ that can be removed from $\mathcal{G}_\mathcal{C}$. Denote by $\mathcal{G}_{\mathcal{C}^{\setminus i}}$ the resulting graphical model and by $\mathcal{C}^{\setminus i}$ the corresponding code over I. Note that properties P1)–P3) of proper complexity measures ensure that such a local constraint $\mathcal{C}_i$ exists without loss of generality and that the complexity of the resulting graphical model $\mathcal{G}_\mathcal{C}$ is at most m. By the induction hypothesis, the $\mathcal{M}$-induced tree complexity of $\mathcal{C}^{\setminus i}$ is upper-bounded by

$$t_\mathcal{M}(\mathcal{C}) \leq m\left((x-2)\delta_\mathcal{M} + 1\right). \tag{49}$$

Because $\mathcal{C}_i$ can be redefined as $m' \leq m$ single parity-check constraints over I, it follows from property P4) of proper graphical model complexity measures that

$$\begin{aligned} t_\mathcal{M}(\mathcal{C}) &\leq m\left((x-2)\delta_\mathcal{M} + 1\right) + m'\delta_\mathcal{M} \\ &\leq m\left((x-1)\delta_\mathcal{M} + 1\right) \end{aligned} \tag{50}$$

completing the proof. □

Theorem 4 implies the following generalization of Corollary 1 to arbitrary proper graphical model complexity measures.

Corollary 4: Let $\mathcal{M}$ be a proper graphical model complexity measure. Let $\mathcal{C}$ be a linear code over $\mathbb{F}_q$ with $\mathcal{M}$-induced tree complexity $t_\mathcal{M}(\mathcal{C})$. The number of cycles N_m in any graphical model for $\mathcal{C}$ with $\mathcal{M}$-complexity m is lower-bounded by

$$N_m \geq \binom{\lfloor t_\mathcal{M}(\mathcal{C})/\delta_\mathcal{M} m \rfloor}{2}. \tag{51}$$

Proof: The result follows immediately from the application of Proposition 5 in conjunction with the observation that because $\delta_\mathcal{M} \geq 1$, the upper bound of Theorem 4 is further upper-bounded by $t_\mathcal{M}(\mathcal{C}) \leq m\delta_\mathcal{M} X_F$. □

C. The Wolf Measure for Graphical Model Complexity

The local constraint complexity measure used to define q^m-ary graphical models in Section III-A constitutes an upper bound on trellis state complexity over $\mathbb{F}_q$. It was established by Lemma 1 that $\delta_\mathcal{M} = 1$ for this measure and, as a result, the FI-CSB reduces to a form similar to the square-root bound for single-cycle models.[5] The specific upper bound on trellis state complexity considered in Section III.A, however, may not always be the best bound to consider in the context of message-passing decoding algorithms. Specifically, let $\mathcal{G}_\mathcal{C}$ be a graphical model for the linear code $\mathcal{C}$ over $\mathbb{F}_q$. Suppose that $\mathcal{C}_i$ is some local constraint in $\mathcal{G}_\mathcal{C}$ incident on the hidden variable S_j [i.e., $j \in I_S(i)$]. If $\mathcal{C}_i$ is to be decoded optimally via a trellis, then the time axis of that trellis must be ordered in such a way that the $|T_j|$ trellis stages corresponding to S_j are consecutive. The upper bound on trellis state complexity considered in Section III-A does not necessarily respect this ordering requirement. For example, the bit reordering considered in Section C of the Appendix illustrates a violation of this requirement for 4-ary hidden variables.

In light of the above discussion, a graphical complexity measure that is a slight relaxation of that studied in Sections III and IV is examined in detail in this section. Because the relaxed measure uses Wolf's upper bound on trellis state complexity as a measure of local constraint complexity [57], it is denoted the *Wolf measure*.

Definition 5: A graphical model $\mathcal{G}_\mathcal{C}$ for a linear code $\mathcal{C}$ over $\mathbb{F}_q$ has Wolf measure $(\mathcal{W})m$ if:

- the alphabet index size of every hidden variable $S_i, i \in I_S$, satisfies $|T_i| \leq m$;
- every local constraint $\mathcal{C}_i, i \in I_C$, satisfies

$$\min(k(\mathcal{C}_i), n(\mathcal{C}_i) - k(\mathcal{C}_i)) \leq m. \tag{52}$$

In the following, the term *Wolf complexity* is used as shorthand for the $\mathcal{W}$-induced tree complexity $t_\mathcal{W}(\mathcal{C})$.

By definition, the Wolf measure obeys property P1) of proper graphical model complexity measures. Following arguments similar to those for q^m-ary graphical models, it is readily verified that the Wolf measure obeys properties P2) and P3) as well. It, therefore, remains to specify how Wolf complexity grows with the addition of a single parity-check constraint. Before stating and proving Lemma 2 below, a useful property of the Wolf measure is first described. Proposition 6 is an analog of the constraint refinement studied by Forney in the context of the constraint complexity measure [35].

Proposition 6: Let $\mathcal{G}_\mathcal{C}$ be a graphical model for the code $\mathcal{C}$ with Wolf measure m. Without loss of generality, the maximum degree of any local constraint in $\mathcal{G}_\mathcal{C}$ is 3.

[5]Indeed, for any proper graphical complexity measure $\mathcal{M}$ satisfying $\delta_\mathcal{M} = 1$, the generalized FI-CSB reduces to Corollary 2 for single-cycle models.

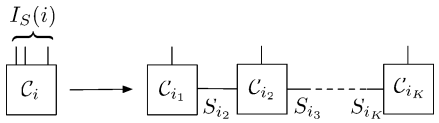

Fig. 3. Replacement of the local constraint $\mathcal{C}_i$ by an equivalent trellis realization.

Proof: The result is proved explicitly for an internal local constraint; however, the same argument can be made for interface constraints. Let $\mathcal{C}_i$ be an internal local constraint in $\mathcal{G}_\mathcal{C}$ such that $|I_S(i)| = K \geq 3$. As illustrated in Fig. 3, $\mathcal{C}_i$ can be replaced by the degree-3 constraints $\mathcal{C}_{i_1}, \ldots, \mathcal{C}_{i_K}$ corresponding to a trellis realization of $\mathcal{C}_i$. If $k(\mathcal{C}_i) \leq m$, then the trellis realization of $\mathcal{C}_i$ can be constructed via the generator matrix method, while if $n(\mathcal{C}_i) - k(\mathcal{C}_i) \leq m$, then the parity-check matrix method may be employed (cf., [59]). In either case, the maximum alphabet index size of the new hidden variables $S_{i_2}, \ldots, S_{i_K}$ is m. Following Proposition 2, it is readily verified that for all $j \in [1, K]$

$$\min(k(\mathcal{C}_{i_j}), n(\mathcal{C}_{i_j}) - k(\mathcal{C}_{i_j})) \leq m. \tag{53}$$

Therefore, the replacement of $\mathcal{C}_i$ in $\mathcal{G}_\mathcal{C}$ by an equivalent trellis realization neither increases graphical model complexity in terms of the Wolf measure, nor fundamentally alters the cyclic topology of the model. □

Lemma 2: Let $\mathcal{C}$ and $\mathcal{C}^{\mathrm{SPC}}$ be linear codes over $\mathbb{F}_q$ defined on the index set I such that $\mathcal{C}^{\mathrm{SPC}}$ comprises a q-ary single parity-check code on some subset of the index set $J \subseteq I$. The Wolf complexity of $\tilde{\mathcal{C}} = \mathcal{C} \cap \mathcal{C}^{\mathrm{SPC}}$ is upper-bounded by

$$t_\mathcal{W}(\tilde{\mathcal{C}}) \leq t_\mathcal{W}(\mathcal{C}) + 2 \tag{54}$$

so that $\delta_\mathcal{W} = 2$.

Proof: The result is proved by explicit construction of a cycle-free model for $\tilde{\mathcal{C}}$ with Wolf measure at most $t_\mathcal{W}(\mathcal{C})+2$ as follows. Let $\mathcal{G}_\mathcal{C}$ be a cycle-free graphical model for $\mathcal{C}$ with Wolf measure $t_\mathcal{W}(\mathcal{C})$, wherein $\mathcal{T}$ is the minimal connected subtree of $\mathcal{G}_\mathcal{C}$ containing the interface constraints that involve the visible variables in J. Construct a cycle-free model $\mathcal{G}_{\tilde{\mathcal{C}}}$ for $\tilde{\mathcal{C}}$ following the same procedure as used in the proof of Lemma 1. It is clear that the maximum hidden variable alphabet index size in $\mathcal{G}_{\tilde{\mathcal{C}}}$ is at most $t_\mathcal{W}(\mathcal{C}) + 1$. It, therefore, remains to consider the local constraints in $\mathcal{G}_{\tilde{\mathcal{C}}}$.

Let $\mathcal{C}_i$ be a local constraint in $\mathcal{G}_\mathcal{C}$ that is updated as per the proof of Lemma 1 and denote the resulting updated local constraint in $\mathcal{G}_{\tilde{\mathcal{C}}}$ by $\tilde{\mathcal{C}}_i$. By Proposition 6, the degree of $\mathcal{C}_i$ is at most 3. There are two cases to consider. First, suppose that $\mathcal{C}_i$ is updated via the addition of a single repetition constraint. In this case, the degree of the vertex corresponding to $\mathcal{C}_i$ in $\mathcal{T}$ is two so that $n(\tilde{\mathcal{C}}_i) = n(\mathcal{C}_i) + 2, k(\tilde{\mathcal{C}}_i) = k(\mathcal{C}_i) + 1$, and

$$\begin{aligned}\min(k(\tilde{\mathcal{C}}_i), n(\tilde{\mathcal{C}}_i) - k(\tilde{\mathcal{C}}_i)) &= \min(k(\mathcal{C}_i), n(\mathcal{C}_i) - k(\mathcal{C}_i)) + 1 \\ &\leq t_\mathcal{W}(\mathcal{C}) + 1.\end{aligned} \tag{55}$$

Next, suppose that $\mathcal{C}_i$ is updated via the addition of a q-ary single parity-check constraint. In this case, the degree of the vertex corresponding to $\mathcal{C}_i$ in $\mathcal{T}$ is three so that $n(\tilde{\mathcal{C}}_i) = n(\mathcal{C}_i)+3, k(\tilde{\mathcal{C}}_i) = k(\mathcal{C}_i) + 2$, and

$$\begin{aligned}\min(k(\tilde{\mathcal{C}}_i), n(\tilde{\mathcal{C}}_i) - k(\tilde{\mathcal{C}}_i)) &\leq \min(k(\mathcal{C}_i), n(\mathcal{C}_i) - k(\mathcal{C}_i)) + 2 \\ &\leq t_\mathcal{W}(\mathcal{C}) + 2\end{aligned} \tag{56}$$

completing the proof. □

Theorem 4 and Corollary 4 can now be immediately specialized to the Wolf measure.

Theorem 5: Let $\mathcal{G}_\mathcal{C}$ be a graphical model for the linear code $\mathcal{C}$ with Wolf measure m and forest-inducing cut size X_F. The Wolf complexity of $\mathcal{C}$ is upper-bounded by

$$t_\mathcal{W}(\mathcal{C}) \leq 2(X_F - 1)m + m. \tag{57}$$

Corollary 5: Let $\mathcal{C}$ be a linear code over $\mathbb{F}_q$ with Wolf complexity $t_\mathcal{W}(\mathcal{C})$. The number of cycles N_m in any graphical model for $\mathcal{C}$ with Wolf measure m is lower-bounded by

$$N_m \geq \binom{\lfloor t_\mathcal{W}(\mathcal{C})/2m \rfloor}{2}. \tag{58}$$

Comparing Theorems 3 and 5, the Wolf measure and the complexity measure introduced in Section III yield similar interpretations of the tradeoff between cyclic topology and complexity. Specializing Theorem 5 to single-cycle models, however, illustrates a difference between the two respective graphical model complexity measures.

Corollary 6: Let $\mathcal{C}$ be a linear code over $\mathbb{F}_q$ with Wolf complexity $t_\mathcal{W}(\mathcal{C})$ and let $m_\mathcal{W}$ be the smallest integer such that there exists a graphical model for $\mathcal{C}$ with Wolf measure $m_\mathcal{W}$ containing at most a single cycle. Then

$$m_\mathcal{W} \geq t_\mathcal{W}(\mathcal{C})/3. \tag{59}$$

Thus, the Wolf complexity measure yields a cube-root bound rather than a square-root bound for single-cycle models. In light of the CSB, however, it is clear that this cube-root bound cannot be met so that the complexity measure introduced in Section III yields a bound that may be in some sense tighter than that afforded by the Wolf measure for single-cycle models.

Note finally that the proof of Lemma 2 can also be used to show that $\delta_\mathcal{M} \geq 2$ for the constraint complexity measure [35], [58]. Statements identical to Theorem 5 and Corollary 5 can, therefore, be made for constraint complexity and treewidth.

VI. Graphical Model Transformation

Let $\mathcal{G}_\mathcal{C}$ be a graphical model for the linear code $\mathcal{C}$ over $\mathbb{F}_q$. This work introduces eight *basic graphical model operations* the application of which to $\mathcal{G}_\mathcal{C}$ results in a new graphical model for $\mathcal{C}$.

1) The merging of two local constraints $\mathcal{C}_{i_1}$ and $\mathcal{C}_{i_2}$ into the new local constraint $\mathcal{C}_i$, which satisfies
$$\mathcal{C}_i \triangleq \mathcal{C}_{i_1} \cap \mathcal{C}_{i_2}. \tag{60}$$
2) The splitting of a local constraint $\mathcal{C}_j$ into two new local constraints $\mathcal{C}_{j_1}$ and $\mathcal{C}_{j_2}$, which satisfy
$$\mathcal{C}_{j_1} \cap \mathcal{C}_{j_2} \triangleq \mathcal{C}_j. \tag{61}$$

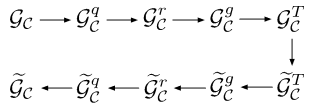

Fig. 4. Transformation of $\mathcal{G}_\mathcal{C}$ into $\tilde{\mathcal{G}}_\mathcal{C}$ via nine subtransformations.

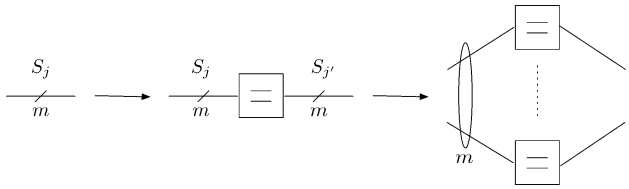

Fig. 5. Transformation of the q^m-ary hidden variable S_j into q-ary hidden variables.

3) The insertion/removal of a degree-2 repetition constraint.
4) The insertion/removal of a trivial length 0, dimension 0 local constraint.
5) The insertion/removal of an isolated partial parity-check constraint.

Note that some of these operations have been introduced implicitly in this paper and in other publications. For example, the proof of the local constraint involvement property of q^m-ary graphical models presented in Section III-B utilizes degree-2 repetition constraint insertion. Local constraint merging has been considered by a number of authors under the rubric of clustering (e.g., [9] and [10]). This work introduces the term merging specifically so that it can be contrasted with its inverse operation: splitting. Detailed definitions of each of the eight basic graphical model operations are given in Section D of the Appendix. In this section, it is shown that these basic operations span the entire space of graphical models for $\mathcal{C}$.

Theorem 6: Let $\mathcal{G}_\mathcal{C}$ and $\tilde{\mathcal{G}}_\mathcal{C}$ be two graphical models for the linear code $\mathcal{C}$ over $\mathbb{F}_q$. Then, $\mathcal{G}_\mathcal{C}$ can be transformed into $\tilde{\mathcal{G}}_\mathcal{C}$ via the application of a *finite* number of basic graphical model operations.

Proof: Define the following four subtransformations, which can be used to transform $\mathcal{G}_\mathcal{C}$ into a Tanner graph $\mathcal{G}_\mathcal{C}^T$:

1) the transformation of $\mathcal{G}_\mathcal{C}$ into a q-ary model $\mathcal{G}_\mathcal{C}^q$;
2) the transformation of $\mathcal{G}_\mathcal{C}^q$ into a (possibly) redundant generalized Tanner graph $\mathcal{G}_\mathcal{C}^r$;
3) the transformation of $\mathcal{G}_\mathcal{C}^r$ into a nonredundant generalized Tanner graph $\mathcal{G}_\mathcal{C}^g$;
4) the transformation of $\mathcal{G}_\mathcal{C}^g$ into a Tanner graph $\mathcal{G}_\mathcal{C}^T$.

Because each basic graphical model operation has an inverse, $\mathcal{G}_\mathcal{C}^T$ can be transformed into $\mathcal{G}_\mathcal{C}$ by inverting each of the four subtransformations. To prove that $\mathcal{G}_\mathcal{C}$ can be transformed into $\tilde{\mathcal{G}}_\mathcal{C}$ via the application of a finite number of basic graphical model operations, it suffices to show that each of the four sub-transformations requires a finite number of operations and that the transformation of the Tanner graph $\mathcal{G}_\mathcal{C}^T$ into a Tanner graph $\tilde{\mathcal{G}}_\mathcal{C}^T$ corresponding to $\tilde{\mathcal{G}}_\mathcal{C}$ requires a finite number of operations. This proof summary is illustrated in Fig. 4.

That each of the five subtransformations from $\mathcal{G}_\mathcal{C}$ to $\tilde{\mathcal{G}}_\mathcal{C}^T$ illustrated in Fig. 4 requires only a finite number of basic graphical model operations is proved below.

1) $\mathcal{G}_\mathcal{C} \rightarrow \mathcal{G}_\mathcal{C}^q$: The graphical model $\mathcal{G}_\mathcal{C}$ is transformed into the q-ary model $\mathcal{G}_\mathcal{C}^q$ as follows. Each local constraint $\mathcal{C}_i$ in $\mathcal{G}_\mathcal{C}$ is split into the $n(\mathcal{C}_i) - k(\mathcal{C}_i)q$-ary single parity-check constraints that define it. A degree-2 repetition constraint is then inserted into every hidden variable with alphabet index set size $m > 1$ and these repetition constraints are then each split into mq-ary repetition constraints as illustrated in Fig. 5. Each local constraint $\mathcal{C}_j$ in the resulting graphical model satisfies $n(\mathcal{C}_j) - k(\mathcal{C}_j) = 1$. Similarly, each hidden variable S_j in the resulting graphical model satisfies $|T_j| = 1$.

2) $\mathcal{G}_\mathcal{C}^q \rightarrow \mathcal{G}_\mathcal{C}^r$: A (possibly redundant) generalized Tanner graph is simply a bipartite q-ary graphical model with one vertex class corresponding to repetition constraints and one to single parity-check constraints in which visible variables are incident only on repetition constraints. By appropriately inserting degree-2 repetition constraints, the q-ary model $\mathcal{G}_\mathcal{C}^q$ can be transformed into $\mathcal{G}_\mathcal{C}^r$.

3) $\mathcal{G}_\mathcal{C}^r \rightarrow \mathcal{G}_\mathcal{C}^g$: Let the generalized Tanner graph $\mathcal{G}_\mathcal{C}^r$ correspond to an $r_H \times (n(\mathcal{C}) + g)$ redundant parity-check matrix $H_\mathcal{C}^{(r,g)}$ for a degree-g generalized extension of $\mathcal{C}$ with rank

$$\operatorname{rank}\left(H_\mathcal{C}^{(r,g)}\right) = n(\mathcal{C}) - k(\mathcal{C}) + g. \tag{62}$$

A finite number of row operations can be applied to $H_\mathcal{C}^{(r,g)}$ resulting in a new parity-check matrix the last $r_H - \operatorname{rank}(H_\mathcal{C}^{(r,g)})$ rows of which are all zero. Similarly, a finite number of basic operations can be applied to $\mathcal{G}_\mathcal{C}^r$ resulting in a generalized Tanner graph containing $r_H - \operatorname{rank}(H_\mathcal{C}^{(r,g)})$ trivial constraints, which can then be removed to yield $\mathcal{G}_\mathcal{C}^g$. Specifically, consider the row operation on $H_\mathcal{C}^{(r,g)}$, which replaces a row $\boldsymbol{h}_i$ by

$$\tilde{\boldsymbol{h}}_i = \boldsymbol{h}_i + \beta_j \boldsymbol{h}_j \tag{63}$$

where $\beta_j \in \mathbb{F}_q$. The graphical model transformation corresponding to this row operation first merges the q-ary single parity-check constraints $\mathcal{C}_i$ and $\mathcal{C}_j$ (which correspond to rows $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$, respectively) and then splits the resulting check into the constraints $\tilde{\mathcal{C}}_i$ and $\mathcal{C}_j$ (which correspond to rows $\tilde{\boldsymbol{h}}_i$ and $\boldsymbol{h}_j$, respectively). Note that this procedure is valid because

$$\mathcal{C}_i \cap \mathcal{C}_j \triangleq \tilde{\mathcal{C}}_i \cap \mathcal{C}_j. \tag{64}$$

4) $\mathcal{G}_\mathcal{C}^g \rightarrow \mathcal{G}_\mathcal{C}^T$: Let the degree-$g$ generalized Tanner graph $\mathcal{G}_\mathcal{C}^g$ correspond to an $(n(\mathcal{C}) - k(\mathcal{C}) + g) \times (n(\mathcal{C}) + g)$ parity-check matrix $H_\mathcal{C}^{(g)}$. A degree-$(g-1)$ generalized Tanner graph $\mathcal{G}_\mathcal{C}^{g-1}$ is obtained from $\mathcal{G}_\mathcal{C}^g$ as follows. Denote by $\widehat{H}_\mathcal{C}^{(g)}$ the parity-check matrix for the degree-g generalized extension defined by $H_\mathcal{C}^{(g)}$, which is systematic in the position corresponding to the gth partial parity symbol. Because a finite number of row operations can be applied to $H_\mathcal{C}^{(g)}$ to yield $\widehat{H}_\mathcal{C}^{(g)}$, a finite number of local constraint merge and split operations can be applied to $\mathcal{G}_\mathcal{C}^g$ to yield the corresponding generalized Tanner graph $\widehat{\mathcal{G}}_\mathcal{C}^g$. Removing the now isolated partial-parity check constraint corresponding to the gth partial parity symbol in $\widehat{\mathcal{G}}_\mathcal{C}^g$ yields the desired degree-$(g-1)$ generalized Tanner graph $\mathcal{G}_\mathcal{C}^{g-1}$. By repeatedly applying this procedure, all partial parity symbols can be removed from $\mathcal{G}_\mathcal{C}^g$ resulting in $\mathcal{G}_\mathcal{C}^T$.

Input: $r_H \times n(\mathcal{C})$ binary parity-check matrix $H_{\mathcal{C}}$.
Output: $r_H \times n(\mathcal{C})$ binary parity-check matrix $H'_{\mathcal{C}}$.

$H'_{\mathcal{C}} \leftarrow H_{\mathcal{C}}$; $i^\star \leftarrow -1$; $j^\star \leftarrow -1$; $g^\star \leftarrow$ girth of $\mathrm{TG}\,(H'_{\mathcal{C}})$;
$N^\star_{g^\star} \leftarrow$ number of $g^\star$-cycles in $\mathrm{TG}\,(H'_{\mathcal{C}})$;
$N^\star_{g^\star+2} \leftarrow$ number of $g^\star+2$-cycles in $\mathrm{TG}\,(H'_{\mathcal{C}})$;
repeat
  if $i^\star \neq j^\star$ **then** Replace row $\boldsymbol{h}_{j^\star}$ in $H'_{\mathcal{C}}$ with binary sum of rows $\boldsymbol{h}_{i^\star}$ and $\boldsymbol{h}_{j^\star}$;
  $i^\star \leftarrow -1$; $j^\star \leftarrow -1$;
  for $i, j = 0, \ldots, r_H - 1$, $i \neq j$ **do**
    Replace row $\boldsymbol{h}_j$ in $H'_{\mathcal{C}}$ with binary sum of rows $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$;
    $g \leftarrow$ girth of $\mathrm{TG}\,(H'_{\mathcal{C}})$;
    $N_g \leftarrow$ number of g-cycles in $\mathrm{TG}\,(H'_{\mathcal{C}})$;
    $N_{g+2} \leftarrow$ number of $g+2$-cycles in $\mathrm{TG}\,(H'_{\mathcal{C}})$;
    if $g > g^\star$ **then**
      $g^\star \leftarrow g$; $i^\star \leftarrow i$; $j^\star \leftarrow j$; $N^\star_g \leftarrow N_g$;
      $N^\star_{g+2} \leftarrow N_{g+2}$;
    end
    else if $g = g^\star$ *AND* $N_g < N^\star_g$ **then** $i^\star \leftarrow i$;
    $j^\star \leftarrow j$; $N^\star_g \leftarrow N_g$; $N^\star_{g+2} \leftarrow N_{g+2}$;
    else if $g = g^\star$ *AND* $N_g = N^\star_g$ **then**
      if $N_{g+2} < N^\star_{g+2}$ **then** $i^\star \leftarrow i$; $j^\star \leftarrow j$;
      $N^\star_{g+2} \leftarrow N_{g+2}$;
    end
    Undo row replacement;
  end
until $i^\star = -1$ & $j^\star = -1$;
return $H'_{\mathcal{C}}$

Algorithm 1: Greedy heuristic for the reduction of short cycles in Tanner graphs for binary codes.

5) $\mathcal{G}^T_{\mathcal{C}} \rightarrow \tilde{\mathcal{G}}^T_{\mathcal{C}}$: Let the Tanner graphs $\mathcal{G}^T_{\mathcal{C}}$ and $\tilde{\mathcal{G}}^T_{\mathcal{C}}$ correspond to the parity-check matrices $H_{\mathcal{C}}$ and $\tilde{H}_{\mathcal{C}}$, respectively. Because $H_{\mathcal{C}}$ can be transformed into $\tilde{H}_{\mathcal{C}}$ via a finite number of row operations, $\mathcal{G}^T_{\mathcal{C}}$ can be similarly transformed into $\tilde{\mathcal{G}}^T_{\mathcal{C}}$ via the application of a finite number of local constraint merge and split operations. □

VII. Graphical Model Extraction via Transformation

The set of basic model operations introduced in the previous section enables the space of all graphical models for a given code $\mathcal{C}$ to be searched, thus allowing for model extraction to be expressed as an optimization problem. The challenges of defining extraction as optimization are twofold. First, a cost measure on the space of graphical models must be found, which is simultaneously meaningful in some real sense (e.g., highly correlated with decoding performance) and computationally tractable. Second, given that discrete optimization problems are, in general, very hard, heuristics for extraction must be found. In this section, heuristics are investigated for the extraction of graphical models for binary linear block codes from an initial Tanner graph. The cost measures considered are functions of the short cycle structure of graphical models. The use of such cost measures is motivated first by empirical evidence concerning the detrimental effect of short cycles on decoding performance (cf., [6], [10]–[16]), and second, by the existence of an efficient algorithm for counting short cycles in bipartite graphs [16]. Simulation results for the models extracted via these heuristics for a number of extended BCH codes are presented and discussed in Section VII-D.

A. A Greedy Heuristic for Tanner Graph Extraction

The Tanner graphs corresponding to many linear block codes of practical interest *necessarily* contain many short cycles [29]. Suppose that any Tanner graph for a given code $\mathcal{C}$ must have girth at least $g_{\min}(\mathcal{C})$; an interesting problem is the extraction of a Tanner graph for $\mathcal{C}$ containing the smallest number of $g_{\min}(\mathcal{C})$-cycles. The extraction of such Tanner graphs is especially useful in the context of ad hoc decoding algorithms that utilize Tanner graphs such as Jiang and Narayanan's stochastic shifting-based iterative decoding algorithm for cyclic codes [60] and the random redundant iterative decoding algorithm presented in [61].

The procedure defined by Algorithm 1 performs a greedy search for a Tanner graph for $\mathcal{C}$ with girth $g_{\min}(\mathcal{C})$ and the smallest number of $g_{\min}(\mathcal{C})$-cycles starting with an initial Tanner graph $\mathrm{TG}(H_{\mathcal{C}})$, which corresponds to some binary parity-check matrix $H_{\mathcal{C}}$. Define an (i,j)-row operation as the replacement of row $\boldsymbol{h}_j$ in $H_{\mathcal{C}}$ by the binary sum of rows $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$. As detailed in the proof of Theorem 6, if $\mathcal{C}_i$ and $\mathcal{C}_j$ are the single parity-check constraints in $\mathrm{TG}(H_{\mathcal{C}})$ corresponding to $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$, respectively, then an (i,j)-row operation in $H_{\mathcal{C}}$ is equivalent to merging $\mathcal{C}_i$ and $\mathcal{C}_j$ to form a new constraint $\mathcal{C}_{i,j} \triangleq \mathcal{C}_i \cap \mathcal{C}_j$ and then splitting $\mathcal{C}_{i,j}$ into $\mathcal{C}_i$ and $\tilde{\mathcal{C}}_j$ (where $\tilde{\mathcal{C}}_j$ enforces the binary sum of rows $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$). Algorithm 1 iteratively finds the rows $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$ in $H_{\mathcal{C}}$ with corresponding (i,j)-row operation that results in the largest short cycle reduction in $\mathrm{TG}(H_{\mathcal{C}})$ at every step. This greedy search continues until there are no more row operations that improve the short cycle structure of $\mathrm{TG}(H_{\mathcal{C}})$.

B. A Greedy Heuristic for Generalized Tanner Graph Extraction

The study of generalized Tanner graphs (GTGs) was introduced by Yedidia *et al.* in [38] to obtain sparse representations for codes with necessarily dense Tanner graphs. A number of authors have studied the extraction of GTGs of codes for which $g_{\min}(\mathcal{C}) = 4$ with a particular focus on models that are four-cycle-free and that correspond to generalized code extensions of minimal degree [62], [63]. Minimal degree extensions are sought because no information is available to the decoder about the partial parity symbols in a generalized Tanner graph and the introduction of too many such symbols has been observed empirically to adversely affect decoding performance [63].

Generalized Tanner graph extraction algorithms proceed via the insertion of partial parity symbols, an operation which is most readily described as a parity-check matrix manipulation.[6]

[6]Note that partial parity insertion can also be viewed through the lens of graphical model transformation. The insertion of a partial parity symbol proceeds via the insertion of an isolated partial parity check followed by a series of local constraint merge and split operations.

Following the notation introduced in Section II-F, suppose that a partial parity on the coordinates indexed by

$$J \subseteq I \cup \{p_1, p_2, \ldots, p_g\} \tag{65}$$

is to be introduced to a GTG for $\mathcal{C}$ corresponding to a degree-g generalized extension $\tilde{\mathcal{C}}$ with parity-check matrix $H_{\tilde{\mathcal{C}}}$. A row $\boldsymbol{h}_p$ is first appended to $H_{\tilde{\mathcal{C}}}$ with 1 in the positions corresponding to coordinates indexed by J and 0 in the other positions. A column is then appended to $H_{\tilde{\mathcal{C}}}$ with, in the case of binary codes, 1 only in the position corresponding to $\boldsymbol{h}_p$ (note that this is readily generalized to nonbinary codes). The resulting parity-check matrix $H_{\widehat{\mathcal{C}}}$ describes a degree-$g+1$ generalized extension $\widehat{\mathcal{C}}$. Every row $\boldsymbol{h}_i \neq \boldsymbol{h}_p$ in $H_{\widehat{\mathcal{C}}}$, which contains 1 in all of the positions corresponding to coordinates indexed by J is then replaced by the binary sum of $\boldsymbol{h}_i$ and $\boldsymbol{h}_p$. Suppose that there are $r(J)$ such rows. It is readily verified that the forest-inducing cut size $\widehat{X}_F$ of the GTG that results from this insertion is related to that of the initial GTG, $\tilde{X}_F$, by

$$\Delta X_F = \tilde{X}_F - \widehat{X}_F = (|J| - 1)(r(J) - 1). \tag{66}$$

Algorithm 3 performs a greedy search for a four-cycle-free generalized Tanner graph for $\mathcal{C}$ with the smallest number of inserted partial parity symbols starting with an initial Tanner graph $\mathrm{TG}(H_{\mathcal{C}})$, which corresponds to some binary parity-check matrix $H_{\mathcal{C}}$. Algorithm 3 iteratively finds the symbol subsets that result in the largest forest-inducing cut size reduction and then introduces the partial parity symbol corresponding to one of those subsets. At each step, Algorithm 3 uses Algorithm 2 to generate a candidate list of partial parity symbols to insert and chooses from that list the symbol, which reduces the most short cycles when inserted. This greedy procedure continues until the generalized Tanner graph contains no four cycles.

Algorithm 3 is closely related to the GTG extraction heuristics proposed by Sankaranarayanan and Vasić [62] and Kumar and Milenkovic [63] (henceforth referred to as the SV and KM heuristics, respectively). It is readily shown that Algorithm 3 is guaranteed to terminate using the proof technique of [62]. The SV heuristic considers only the insertion of partial parity symbols corresponding to coordinate index sets of size 2 (i.e., $|J| = 2$). The KM heuristic considers only the insertion of partial parity symbols corresponding to coordinate index sets satisfying $r(J) = 2$. Algorithm 2, however, considers all coordinate index sets satisfying $|J| = 2, 3, 4$ and $r(J) = 2, 3, 4$ and then uses (66) to evaluate which of these coordinate sets results in the largest tree-inducing cut size reduction. Algorithm 3 is thus able to extract GTGs corresponding to generalized extensions of smaller degree than the SV and KM heuristics. To illustrate this observation, the degrees of the generalized code extensions that result when the SV, KM, and proposed (HC) heuristics are applied to parity-check matrices for three codes are provided in Table I. Fig. 6 compares the performance of the three extracted GTG decoding algorithms for the $[31, 21, 5]$ BCH code to illustrate the efficacy of extracting GTGs corresponding to extensions of smallest possible degree. Note that while the decoding algorithm extracted using the HC heuristic outperforms those corresponding to the SV and KM heuristics, respectively, it still loses nearly 1 dB with respect to optimal (trellis) decoding thus motivating the search for more sophisticated graphical models.

Input: Binary generalized parity-check matrix $H_{\tilde{\mathcal{C}}}$.
Output: List $\mathcal{S}$ of best partial parity symbols sets.

$\mathcal{S} \leftarrow \emptyset$; $\Delta X_F^\star \leftarrow 0$;

// Consider pairs of columns of $H_{\tilde{\mathcal{C}}}$.
$J_2 \leftarrow$ coordinate pair that maximizes $r(J_2)$;
Append J_2 to $\mathcal{S}$; $\Delta X_F^\star \leftarrow r(J_2) - 1$;

// Consider 3-tuples of columns of $H_{\tilde{\mathcal{C}}}$.
$J_3 \leftarrow$ coordinate 3-tuple that maximizes $r(J_3)$;
if $2(r(J_3) - 1) > \Delta X_F^\star$ **then**
  $\mathcal{S} \leftarrow \emptyset$; Append J_3 to $\mathcal{S}$; $\Delta X_F^\star \leftarrow 2(r(J_3) - 1)$;
end
else if $2(r(J_3) - 1) = \Delta X_F^\star$ **then** Append J_3 to $\mathcal{S}$;

// Consider 4-tuples of columns of $H_{\tilde{\mathcal{C}}}$.
$J_4 \leftarrow$ coordinate 4-tuple that maximizes $r(J_4)$;
if $3(r(J_4) - 1) > \Delta X_F^\star$ **then**
  $\mathcal{S} \leftarrow \emptyset$; Append J_4 to $\mathcal{S}$; $\Delta X_F^\star \leftarrow 3(r(J_4) - 1)$;
end
else if $3(r(J_4) - 1) = \Delta X_F^\star$ **then** Append J_4 to $\mathcal{S}$;

// Consider pairs of rows of $H_{\tilde{\mathcal{C}}}$.
$J_i \leftarrow$ largest coordinate subset such that $r(J_i) = 2$;
if $|J_i| - 1 > \Delta X_F^\star$ **then**
  $\mathcal{S} \leftarrow \emptyset$; Append J_i to $\mathcal{S}$; $\Delta X_F^\star \leftarrow |J_i| - 1$;
end
else if $|J_i| - 1 = \Delta X_F^\star$ **then** Append J_i to $\mathcal{S}$;

// Consider 3-tuples of rows of $H_{\tilde{\mathcal{C}}}$.
$J_j \leftarrow$ largest coordinate subset such that $r(J_j) = 3$;
if $2(|J_j| - 1) > \Delta X_F^\star$ **then**
  $\mathcal{S} \leftarrow \emptyset$; Append J_j to $\mathcal{S}$; $\Delta X_F^\star \leftarrow 2(|J_j| - 1)$;
end
else if $2(|J_j| - 1) = \Delta X_F^\star$ **then** Append J_j to $\mathcal{S}$;

// Consider 4-tuples of rows of $H_{\tilde{\mathcal{C}}}$.
$J_k \leftarrow$ largest coordinate subset such that $r(J_k) = 4$;
if $3(|J_k| - 1) > \Delta X_F^\star$ **then**
  $\mathcal{S} \leftarrow \emptyset$; Append J_k to $\mathcal{S}$; $\Delta X_F^\star \leftarrow 3(|J_k| - 1)$;
end
else if $3(|J_k| - 1) = \Delta X_F^\star$ **then** Append J_k to $\mathcal{S}$;

return $\mathcal{S}$

Algorithm 2: Heuristic for generating candidate partial parity symbols.

C. A Greedy Heuristic for 2^m-ary Model Extraction

For most codes, the decoding algorithms implied by generalized Tanner graphs exhibit only modest performance gains with respect to those implied by Tanner graphs, if any, thus motivating the search for more complex graphical models. Algorithm 4 iteratively applies the constraint merging operation to obtain a $2^{m^\star}$-ary graphical model from an initial Tanner graph $\mathrm{TG}(H_{\mathcal{C}})$ for some prescribed maximum complexity $m^\star$. At each step, Algorithm 4 determines the pair of local constraints $\mathcal{C}_i$ and $\mathcal{C}_j$, which when merged reduces the most short cycles without violating the maximum complexity constraint $m^\star$. To

Input: Binary parity-check matrix $H_{\mathcal{C}}$.
Output: Binary generalized parity-check matrix $H_{\widetilde{\mathcal{C}}}$.

$H_{\widetilde{\mathcal{C}}} \leftarrow H_{\mathcal{C}}$;
while $\mathrm{GTG}(H_{\widetilde{\mathcal{C}}})$ *contains* 4*-cycles* **do**
  $\mathcal{S} \leftarrow$ set of candidate partial parity symbol subsets from Algorithm 2;
  $J^{\star} \leftarrow$ subset in $\mathcal{S}$ the insertion of which reduces the most 4–cycles in $\mathrm{GTG}(H_{\widetilde{\mathcal{C}}})$;
  Insert symbol corresponding to $J^{\star}$ in $H_{\widetilde{\mathcal{C}}}$;
end
return $H_{\widetilde{\mathcal{C}}}$

Algorithm 3: Greedy heuristic for the removal of 4-cycles in binary generalized Tanner graphs.

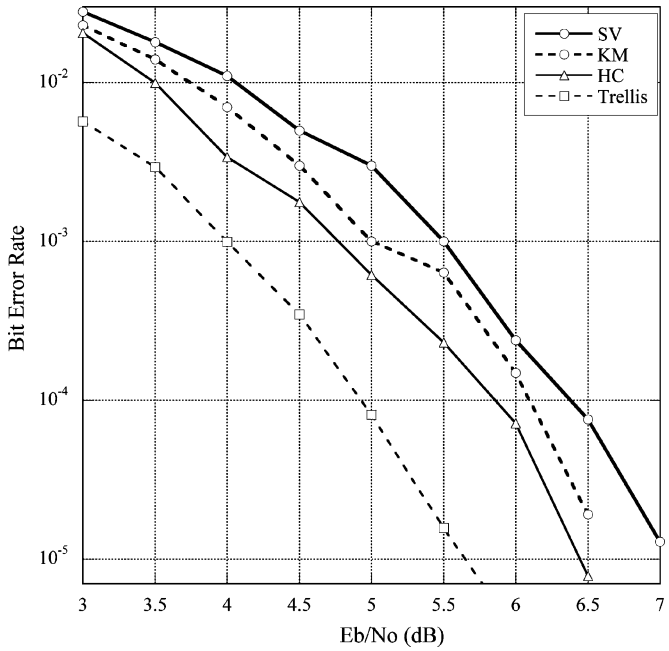

Fig. 6. BER performance of three GTG decoding algorithms for the $[31, 21, 5]$ BCH code. One hundred iterations of a flooding schedule were performed. Binary antipodal signaling over an AWGN channel is assumed.

TABLE I
GENERALIZED CODE EXTENSION DEGREES CORRESPONDING TO THE FOUR-CYCLE-FREE GTGS OBTAINED VIA THE SV, KM, AND HC HEURISTICS

Code	SV	KM	HC
$[23, 12, 7]$ Golay	18	11	10
$[31, 21, 5]$ BCH	47	19	12
$[63, 30, 13]$ BCH	264	121	69

ensure that the efficient cycle counting algorithm of [16] can be utilized, only pairs of constraints that are both internal or both interface are merged at each step. Because the initial Tanner graph is bipartite with vertex classes corresponding to interface (repetition) and internal (single parity-check) constraints, the graphical models that result from every such local constraint merge operations are similarly bipartite.

D. Simulation Results

The proposed extraction heuristics were applied to two extended BCH codes with parameters $[32, 21, 6]$ and $[64, 51, 6]$,

Input: Tanner graph $\mathrm{TG}(H_{\mathcal{C}})$. Max. complexity $m^{\star}$.
Output: $2^{m^{\star}}$-ary graphical model GM for $\mathcal{C}$.

$\mathrm{GM} \leftarrow \mathrm{TG}(H_{\mathcal{C}})$;
repeat
  $(\mathcal{C}_i, \mathcal{C}_j) \leftarrow$ pair of incident or internal constraints the removal of which removes the most 4-cycles from GM while not violating the $2^{m^{\star}}$-ary complexity constraint;
  Merge local constraints $\mathcal{C}_i$ and $\mathcal{C}_j$ in GM;
until *No allowed* 4*-cycle reducing merge operations remain* ;
return GM

Algorithm 4: Greedy heuristic for the extraction of 2^m-ary graphical models.

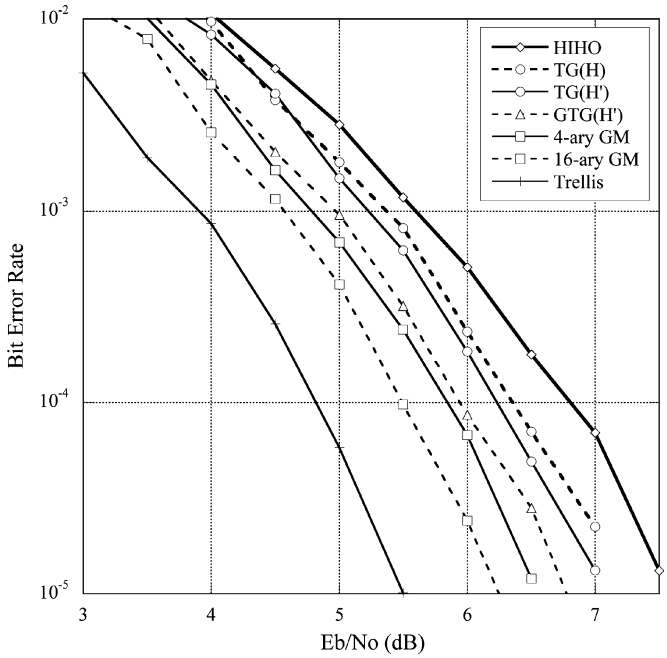

Fig. 7. BER performance of different decoding algorithms for the $[32, 21, 6]$ extended BCH code. Fifty iterations of a flooding schedule were performed for all of the suboptimal SISO decoding algorithms.

respectively. In both Figs. 7 and 8, the performance of a number of suboptimal SISO decoding algorithms for these codes is compared to algebraic hard-in–hard-out (HIHO) decoding (i.e., a classical Berlekamp–Massey-style decoder) and optimal trellis SISO decoding. Binary antipodal signaling over AWGN channels is assumed throughout.

Initial parity-check matrices H were formed by extending cyclic parity-check matrices for the respective $[31, 21, 5]$ and $[63, 51, 5]$ BCH codes (with rows corresponding to cyclic shifts of the generators polynomials of their respective duals) [36]. These initial parity-check matrices were used as inputs to Algorithm 1, yielding the parity-check matrices H', which in turn were used as inputs to Algorithm 3, yielding four-cycle-free generalized Tanner graphs. The suboptimal decoding algorithms implied by these graphical models are labeled $\mathrm{TG}(H), \mathrm{TG}(H')$, and $\mathrm{GTG}(H')$, respectively. The generalized Tanner graphs extracted for the $[32, 21, 6]$ and

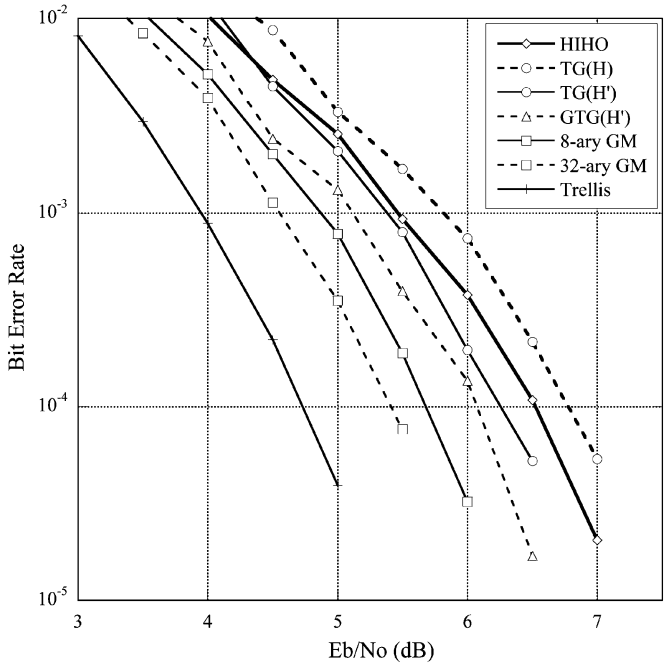

Fig. 8. BER performance of different decoding algorithms for the $[64, 51, 6]$ extended BCH code. Fifty iterations of a flooding schedule were performed for all of the suboptimal SISO decoding algorithms.

TABLE II
SHORT CYCLE STRUCTURE OF THE INITIAL AND EXTRACTED GRAPHICAL MODELS FOR THE $[32, 21, 6]$ EXTENDED BCH CODE

	N_4	N_6	N_8
$\mathrm{TG}(H)$	1128	37404	1126372
$\mathrm{TG}(H')$	453	11152	260170
$\mathrm{GTG}(H')$	0	62	298
4-ary GM	244	3852	50207
16-ary GM	70	340	724

TABLE III
SHORT CYCLE STRUCTURE OF THE INITIAL AND EXTRACTED GRAPHICAL MODELS FOR THE $[64, 51, 6]$ EXTENDED BCH CODE

	N_4	N_6	N_8
$\mathrm{TG}(H)$	9827	1057248	111375740
$\mathrm{TG}(H')$	3797	270554	19374579
$\mathrm{GTG}(H')$	0	163	1229
8-ary GM	847	19590	304416
32-ary GM	201	1384	0

TABLE IV
SHORT CYCLE STRUCTURE OF THE INITIAL AND EXTRACTED GRAPHICAL MODELS FOR THE $[256, 239, 6]$ EXTENDED BCH CODE

	N_4	N_6	N_8
$\mathrm{TG}(H)$	308258	1.99×10^8	1.32×10^{11}
4-ary GM	79060	2.43×10^7	6.76×10^8
16-ary GM	13022	1245115	8.39×10^7
64-ary GM	4462	157520	0

$[64, 51, 6]$ codes correspond to degree-17 and degree-40 generalized extensions, respectively. Finally, the parity-check matrices H' were used as inputs to Algorithm 4 with various values of $m^\star$. The number four-, six-, and eight-cycles (N_4, N_6, N_8) contained in the extracted graphical models for the $[32, 21, 6]$ and $[64, 51, 6]$ codes are given in Tables II and III, respectively.

The utility of Algorithm 1 is illustrated in both Figs. 7 and 8: the $\mathrm{TG}(H')$ algorithms outperform the $\mathrm{TG}(H)$ algorithms by approximately 0.1 and 0.5 dB at a bit error rate (BER) of 10^{-4} for the $[32, 21, 6]$ and $[64, 51, 6]$ codes, respectively. For both codes, the four-cycle-free generalized Tanner graph decoding algorithms outperform Tanner graph decoding by approximately 0.2 dB at a BER of 10^{-4}. Further performance improvements are achieved for both codes by going beyond binary models. Specifically, at a BER of 10^{-5}, the suboptimal SISO decoding algorithm implied by the extracted 16-ary graphical model for the $[32, 21, 6]$ code outperforms algebraic HIHO decoding by approximately 1.5 dB. The minimal trellis for this code is known to contain state variables with alphabet size at least 1024 [39], yet the 16-ary suboptimal SISO decoder performs only 0.7 dB worse at a BER of 10^{-5}. At a BER of 10^{-4}, the suboptimal SISO decoding algorithm implied by the extracted 32-ary graphical model for the $[64, 51, 6]$ code outperforms algebraic HIHO decoding by approximately 1.2 dB. The minimal trellis for this code is known to contain state variables with alphabet size at least 4096 [39]; that a 32-ary suboptimal SISO decoder loses only 0.7 dB with respect to the optimal SISO decoder at a BER of 10^{-4} is notable.

Fig. 9 illustrates the performance of the proposed heuristics when applied to the $[256, 239, 6]$ extended BCH code. The graphical models corresponding to the decoding algorithms illustrated in Fig. 9 were constructed in a manner analogous to those for the $[32, 21, 6]$ and $[64, 51, 6]$ codes. Table IV illustrates the number of four-, six-, and eight-cycles contained in the extracted models. Note that the decoding algorithm implied by the 64-ary graphical model for this code gains less than 1 dB with respect to algebraic decoding. The results of Fig. 9 thus motivate the study of extraction beyond simple greedy searches as well as those that use all of the basic graphical modeling operations (rather than just constraint merging).

VIII. CONCLUSION AND FUTURE WORK

This work studied the space of graphical models for a given code to lay out some of the foundations of the theory of extractive graphical modeling problems. The primary contributions of this work were the introduction of a new bound characterizing the tradeoff between cyclic topology and complexity in graphical models for linear codes and the introduction of a set of basic graphical model transformation operations that were shown to span the space of all graphical models for a given code. It was demonstrated that these operations can be used to extract novel cyclic graphical models—and thus novel suboptimal iterative soft-in–soft-out (SISO) decoding algorithms—for linear block codes.

There are a number of interesting directions for future work motivated by the statement of the FI-CSB and its generalization to proper complexity measures. While the minimal trellis complexity $s(\mathcal{C})$ of linear codes is well understood, less is known about the minimal tree complexity $t(\mathcal{C})$ and characterizing those codes for which $t(\mathcal{C}) < s(\mathcal{C})$ is an open problem. The recent work of Kashyap indicates that tools from matroid theory can be brought to bear on this problem [64]. A study of those codes

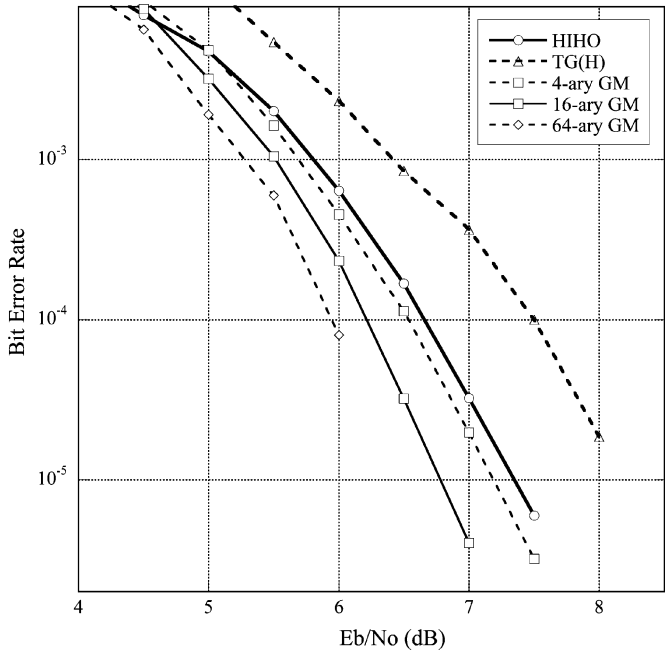

Fig. 9. BER performance of different decoding algorithms for the $[256, 239, 6]$ extended BCH code. Fifty iterations of a flooding schedule were performed for all of the suboptimal SISO decoding algorithms.

that meet or approach the FI-CSB is also an interesting direction for future work, which may provide insight into construction techniques for good codes with short block lengths (e.g., tens to hundreds of bits) defined on graphs with a few cycles (e.g., 3, 6, or 10). The development of statements similar to the FI-CSB for graphical models of more general systems (e.g., group codes, nonlinear codes, and general factor graphs) is also interesting.

There are also a number of interesting directions for future work motivated by the study of graphical model transformation. While the extracted graphical models presented in Section VII-D are notable, ad hoc techniques utilizing massively redundant models and judicious message filtering outperform the models presented in this work [60], [61]. Such massively redundant models contain many more short cycles than the models presented in Section VII-D indicating that short cycle structure alone is not a sufficiently meaningful cost measure for graphical model extraction. It is known that redundancy can be used to remove pseudocodewords (cf., [65]) thus motivating the study of cost measures, which consider both short cycle structure and pseudocodeword spectrum.

Finally, this work has been primarily concerned with the extraction of graphical models for classical linear codes, which are known to have necessarily dense Tanner graphs. A class of extractive graphical modeling problems of particular practical interest concern the extraction of graphical models for *fixed modern codes*. The extraction of graphical models for standard codes (e.g., the DVB-S2, IEEE 802.11n, and IEEE 802.16e LDPC codes [66]), which imply decoding architectures that are particularly amenable to fast hardware implementation, is an important problem currently faced by industry. Furthermore, the extraction of graphical models that imply decoding algorithms that reduce the floors exhibited by Tanner graph decoding of certain codes is also interesting. While the extraction heuristics presented in this work are not suited to such problems (because the Tanner graphs of modern codes tend to avoid four cycles), a number of authors have recently investigated the application of other graphical model transformations—e.g., redundant check insertion [67] and constraint merging [68]—to the Tanner graphs for the length 2640 Margulis code with promising results.

APPENDIX

This Appendix provides detailed definitions of both the q^m-ary graphical model properties described in Section III-B and the basic graphical model operations introduced in Section VI. The proof of Lemma 1 is also further illustrated by example. To elucidate these properties and definitions, a single-cycle graphical model for the extended Hamming code is studied throughout.

A. Single-Cycle Model for the Extended Hamming Code

Fig. 10 illustrates a single-cycle graphical model (i.e., a tail-biting trellis) for the length 8 extended Hamming code $\mathcal{C}_H$. The hidden variables S_1 and S_5 are binary while S_2, S_3, S_4, S_6, S_7, and S_8 are 4-ary. All of the local constraint codes in this model are interface constraints. Equations (67)–(70) define the local constraint codes via generator matrices (where G_i generates $\mathcal{C}_i$)

$$G_1 = \begin{array}{c} \begin{matrix} S_1 & V_1 & S_2 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 10 \\ 0 & 1 & 01 \end{bmatrix} \end{array} \quad G_2 = \begin{array}{c} \begin{matrix} S_2 & V_2 & S_3 \end{matrix} \\ \begin{bmatrix} 10 & 1 & 10 \\ 01 & 1 & 01 \end{bmatrix} \end{array} \tag{67}$$

$$G_3 = \begin{array}{c} \begin{matrix} S_3 & V_3 & S_4 \end{matrix} \\ \begin{bmatrix} 10 & 0 & 01 \\ 01 & 0 & 11 \\ 00 & 1 & 01 \end{bmatrix} \end{array} \quad G_4 = \begin{array}{c} \begin{matrix} S_4 & V_4 & S_5 \end{matrix} \\ \begin{bmatrix} 10 & 1 & 0 \\ 01 & 1 & 1 \end{bmatrix} \end{array} \tag{68}$$

$$G_5 = \begin{array}{c} \begin{matrix} S_5 & V_5 & S_6 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 10 \\ 0 & 1 & 01 \end{bmatrix} \end{array} \quad G_6 = \begin{array}{c} \begin{matrix} S_6 & V_6 & S_7 \end{matrix} \\ \begin{bmatrix} 10 & 1 & 10 \\ 01 & 1 & 01 \end{bmatrix} \end{array} \tag{69}$$

$$G_7 = \begin{array}{c} \begin{matrix} S_7 & V_7 & S_8 \end{matrix} \\ \begin{bmatrix} 10 & 0 & 01 \\ 01 & 0 & 11 \\ 00 & 1 & 01 \end{bmatrix} \end{array} \quad G_8 = \begin{array}{c} \begin{matrix} S_8 & V_8 & S_1 \end{matrix} \\ \begin{bmatrix} 10 & 1 & 0 \\ 01 & 1 & 1 \end{bmatrix} \end{array}. \tag{70}$$

The graphical model for $\mathcal{C}_H$ illustrated in Fig. 10 is 4-ary (i.e., $q = 2, m = 2$): the maximum hidden variable alphabet index set size is 2 and all local constraints satisfy $\min(k(\mathcal{C}_i), n(\mathcal{C}_i) - k(\mathcal{C}_i)) \leq 2$. The behavior $\mathfrak{B}_H$ of this graphical model is generated by (71), shown at the bottom of the next page. The projection of $\mathfrak{B}_H$ onto the visible variable index set I, $\mathfrak{B}_{H|I}$, is thus generated by

$$G_{\mathfrak{B}_{H|I}} = \begin{array}{c} \begin{matrix} V_1 & V_2 & V_3 & V_4 & V_5 & V_6 & V_7 & V_8 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{array} \tag{72}$$

which coincides precisely with a generator matrix for $\mathcal{C}_H$.

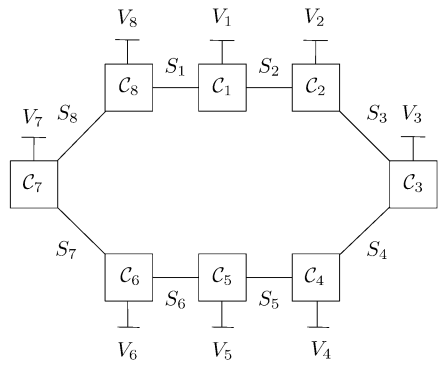

Fig. 10. Tail-biting trellis graphical model for the length 8 extended Hamming code $\mathcal{C}_H$.

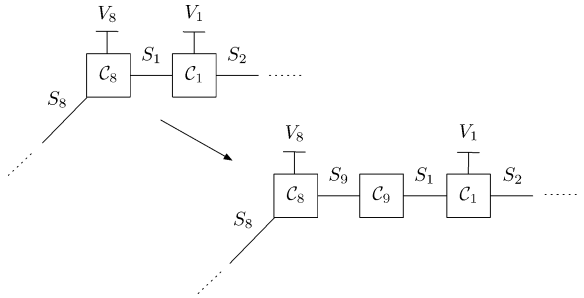

Fig. 11. Insertion of hidden variable S_9 and internal local constraint $\mathcal{C}_9$ into the tail-biting trellis for $\mathcal{C}_H$.

B. q^m-ary Graphical Model Properties

The three properties of q^m-ary graphical models introduced in Section III-B are restated and discussed in detail in the following where it is assumed that a q^m-ary graphical model $\mathcal{G}_\mathcal{C}$ with behavior $\mathfrak{B}$ for a linear code $\mathcal{C}$ over $\mathbb{F}_q$ defined on an index set I is given. Note that the model for the extended Hamming code studied in the previous extension is studied further in this section.

1) Internal Local Constraint Involvement Property: Any hidden variable in a q^m-ary graphical model can be made to be incident (on at least one end) on an *internal* local constraint $\mathcal{C}_i$, which satisfies $n(\mathcal{C}_i) - k(\mathcal{C}_i) \leq m$ without fundamentally altering the complexity or cyclic topology of that graphical model.

Suppose there exists some hidden variable S_j (involved in the local constraints $\mathcal{C}_{j_1}$ and $\mathcal{C}_{j_2}$) that does not satisfy the local constraint involvement property. A new hidden variable S_i that is a copy of S_j is introduced to $\mathcal{G}_\mathcal{C}$ by first redefining $\mathcal{C}_{j_2}$ over S_i and then inserting a local repetition constraint $\mathcal{C}_i$ that enforces $S_j = S_i$. The insertion of S_i and $\mathcal{C}_i$ does not fundamentally alter the complexity of $\mathcal{G}_\mathcal{C}$ because $n(\mathcal{C}_i) - k(\mathcal{C}_i) = |T_j| \leq m$ and because degree-2 repetition constraints are trivial from a decoding complexity viewpoint. Furthermore, the insertion of S_i and $\mathcal{C}_i$ does not fundamentally alter the cyclic topology of $\mathcal{G}_\mathcal{C}$ because no new cycles can be introduced by this procedure.

As an example, consider the binary hidden variable S_1 in Fig. 10, which is incident on the interface constraints $\mathcal{C}_1$ and $\mathcal{C}_8$. By introducing the new binary hidden variable S_9 and binary repetition constraint $\mathcal{C}_9$, as illustrated in Fig. 11, S_1 can be made to be incident on the internal constraint $\mathcal{C}_9$. The insertion of S_9 and $\mathcal{C}_9$ redefines $\mathcal{C}_8$ over S_9 resulting in the generator matrices

$$G_8 = \begin{array}{c} \begin{matrix} S_8 & V_8 & S_9 \end{matrix} \\ \begin{bmatrix} 10 & 1 & 0 \\ 01 & 1 & 1 \end{bmatrix} \end{array} \qquad G_9 = \begin{array}{c} \begin{matrix} S_9 & S_1 \end{matrix} \\ \begin{bmatrix} 1 & 1 \end{bmatrix} \end{array}. \tag{73}$$

Clearly, the modified local constraints $\mathcal{C}_8$ and $\mathcal{C}_9$ satisfy the condition for inclusion in a 4-ary graphical model.

2) Internal Local Constraint Removal Property: The removal of an internal local constraint from a q^m-ary graphical model results in a q^m-ary graphical model for a new code defined on the same index set.

The removal of the internal constraint $\mathcal{C}_r$ from $\mathcal{G}_\mathcal{C}$ in order to define the new code $\mathcal{C}^{\backslash r}$ proceeds as follows. Each hidden variable $S_i, i \in I_S(r)$, is first disconnected from $\mathcal{C}_r$ and connected to a new degree-1 internal constraint $\mathcal{C}_{i'}$, which does not impose any constraint on the value of S_i (because it is degree-1). The local constraint $\mathcal{C}_r$ is then removed from the resulting graphical model yielding $\mathcal{G}_{\mathcal{C}^{\backslash r}}$ with behavior $\mathfrak{B}^{\backslash r}$. The new code $\mathcal{C}^{\backslash r}$ is the projection of $\mathfrak{B}^{\backslash r}$ onto I.

As an example, consider the removal of the internal local constraint $\mathcal{C}_9$ from the graphical model for $\mathcal{C}_H$ described above; the resulting graphical model update is illustrated in Fig. 12. The new codes $\mathcal{C}_{10}$ and $\mathcal{C}_{11}$ are length 1, dimension 1 codes, which thus impose no constraints on S_1 and S_9, respectively. It is readily verified that the code $\mathcal{C}_H^{\backslash 9}$, which results from the removal of $\mathcal{C}_9$ from $\mathcal{C}_H$, has dimension 5 and is generated by

$$G_{H^{\backslash 9}} = \begin{array}{c} \begin{matrix} V_1 & V_2 & V_3 & V_4 & V_5 & V_6 & V_7 & V_8 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \end{array}. \tag{74}$$

$$G_{\mathfrak{B}_H} = \begin{array}{c} \begin{matrix} S_1 & V_1 & S_2 & V_2 & S_3 & V_3 & S_4 & V_4 & S_5 & V_5 & S_6 & V_6 & S_7 & V_7 & S_8 & V_8 \end{matrix} \\ \begin{bmatrix} 0 & 1 & 01 & 1 & 01 & 1 & 10 & 1 & 0 & 0 & 00 & 0 & 00 & 0 & 00 & 0 \\ 0 & 0 & 00 & 0 & 00 & 1 & 01 & 1 & 1 & 0 & 10 & 1 & 10 & 1 & 00 & 0 \\ 0 & 0 & 00 & 0 & 00 & 0 & 00 & 0 & 0 & 1 & 01 & 1 & 01 & 1 & 10 & 1 \\ 1 & 0 & 10 & 1 & 10 & 1 & 00 & 0 & 0 & 0 & 00 & 0 & 00 & 1 & 01 & 1 \end{bmatrix} \end{array}. \tag{71}$$

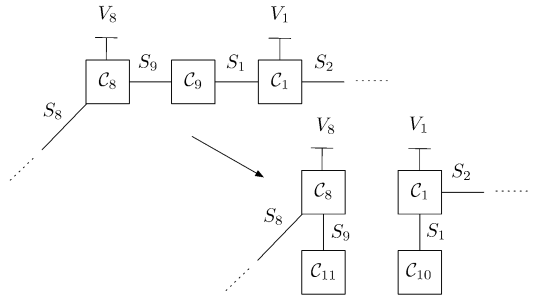

Fig. 12. Removal of internal local constraint $\mathcal{C}_9$ from the tail-biting trellis for $\mathcal{C}_H$.

Note that $\mathcal{C}_H^{\setminus 9}$ corresponds to all paths in the tail-biting trellis representation of $\mathcal{C}_H$, not just those paths that begin and end in the same state.

The removal of an internal local constraint $\mathcal{C}_r$ results in the introduction of $|I_S(r)|$ new degree-1 local constraints. Forney described such constraints as "useless" in [35] and they can indeed be removed from $\mathcal{G}_{\mathcal{C}\setminus r}$ because they impose no constraints on the variables they involve. Specifically, for each hidden variable $S_i, i \in I_S(r)$, involved in the (removed) local constraint $\mathcal{C}_r$, denote by $\mathcal{C}_j$ the other constraint involving S_i in $\mathcal{G}_{\mathcal{C}}$. The constraint $\mathcal{C}_j$ can be redefined as its projection onto $I_V(j) \cup \{I_S(j) \setminus i\}$. It is readily verified that the resulting constraint $\mathcal{C}_j^{\setminus r}$ satisfies the condition for inclusion in a q^m-ary graphical model.

Continuing with the above example, $\mathcal{C}_{10}, \mathcal{C}_{11}, S_1$, and S_9 can be removed from the graphical model illustrated in Fig. 12 by redefining $\mathcal{C}_1$ and $\mathcal{C}_8$ with generator matrices

$$G_1 = \begin{matrix} V_1 & S_2 \end{matrix} \begin{bmatrix} 1 & 01 \\ 0 & 10 \end{bmatrix} \quad G_8 = \begin{matrix} S_8 & V_8 \end{matrix} \begin{bmatrix} 10 & 1 \\ 01 & 1 \end{bmatrix}. \tag{75}$$

3) Internal Local Constraint Redefinition Property: Any internal local constraint $\mathcal{C}_i$ in a q^m-ary graphical model satisfying $n(\mathcal{C}_i) - k(\mathcal{C}_i) = m' \leq m$ can be equivalently represented by $m'q$-ary single parity-check equations over the visible variable index set.

Let $\mathcal{C}_i$ satisfy $n(\mathcal{C}_i) - k(\mathcal{C}_i) = m' \leq m$ and consider a hidden variable S_j involved in $\mathcal{C}_i$ [i.e., $j \in I_S(i)$] with alphabet index set T_j. Each of the $|T_j|$ coordinates of S_j can be redefined as a q-ary sum of some subset of the visible variable set as follows. Consider the behavior $\mathfrak{B}^{\setminus i}$ and corresponding code $\mathcal{C}^{\setminus i}$, which result when $\mathcal{C}_i$ is removed from $\mathcal{G}_{\mathcal{C}}$ (before S_j is discarded). The projection of $\mathfrak{B}^{\setminus i}$ onto $T_j \cup I$, $\mathfrak{B}^{\setminus i}_{|T_j \cup I}$, has length

$$n(\mathcal{C}) + |T_j| \tag{76}$$

and dimension

$$k(\mathcal{C}^{\setminus i}) \geq k(\mathcal{C}) \tag{77}$$

over $\mathbb{F}_q$. There exists a generator matrix for $\mathfrak{B}^{\setminus i}_{|T_j \cup I}$ that is systematic in some size $k(\mathcal{C}^{\setminus i})$ subset of the index set I [36]. A parity-check matrix H_j that is systematic[7] in the $|T_j|$ positions corresponding to the coordinates of S_j can thus be found for this projection; each coordinate of S_j is defined as a q-ary sum of some subset of the visible variables by H_j. Following this procedure, the internal local constraint $\mathcal{C}_i$ is redefined over I by substituting the definitions of S_j implied by H_j for each $j \in I_S(i)$ into each of the $m'q$-ary single parity-check equations, which determine $\mathcal{C}_i$.

Returning to the example of the tail-biting trellis for $\mathcal{C}_H$, the internal local constraint $\mathcal{C}_9$ in Fig. 11 is redefined over the visible variable set as follows. The projection of $\mathfrak{B}_H^{\setminus 9}$ onto $T_1 \cup I$ is generated by

$$G_{\mathfrak{B}^{\setminus 9}_{H|T_1 \cup I}} = \begin{matrix} S_1 & V_1 & V_2 & V_3 & V_4 & V_5 & V_6 & V_7 & V_8 \end{matrix} \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}. \tag{78}$$

A valid parity-check matrix for this projection that is systematic in the position corresponding to S_1 is

$$H_1 = \begin{matrix} S_1 & V_1 & V_2 & V_3 & V_4 & V_5 & V_6 & V_7 & V_8 \end{matrix} \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \tag{79}$$

which defines the binary hidden variable S_1 as

$$S_1 = V_1 + V_2 \tag{80}$$

where addition is over $\mathbb{F}_2$. A similar development defines the binary hidden variable S_9 as

$$S_9 = V_5 + V_8. \tag{81}$$

The local constraint $\mathcal{C}_9$ thus can be redefined to enforce the single parity-check equation

$$V_1 + V_2 + V_5 + V_8 = 0. \tag{82}$$

Finally, to illustrate the use of the q^m-ary graphical model properties in concert, denote by $\mathcal{C}_9^{(1)}$ the single parity-check constraint enforcing (82). It is readily verified that only the first four rows of $G_{H^{\setminus 9}}$ [as defined in (74)] satisfy $\mathcal{C}_9^{(1)}$. It is precisely these four rows that generate $\mathcal{C}_H$ proving that

$$\mathcal{C}_H = \mathcal{C}_H^{\setminus 9} \cap \mathcal{C}_9^{(1)}. \tag{83}$$

C. Illustration of Proof of Lemma 1

In the following, the proof of Lemma 1 is illustrated by updating a cycle-free model for $\mathcal{C}_H^{\setminus 9}$ [as generated by (74)] with the single parity-check constraint defined by (82) in order to obtain a cycle-free graphical model for $\mathcal{C}_H$. A cycle-free binary graphical model for $\mathcal{C}_H^{\setminus 9}$ is illustrated in Fig. 13.[8] All hidden variables in Fig. 13 are binary and the local constraints labeled $\mathcal{C}_{14}, \mathcal{C}_{17}, \mathcal{C}_{20}$, and $\mathcal{C}_{23}$ are binary single parity-check constraints

[7]Let $\mathcal{C}$ be a code defined on the index set I with dual $\mathcal{C}^{\perp}$. A parity-check matrix $H_{\mathcal{C}}$ for the code $\mathcal{C}$ is said to be *systematic* in the coordinates corresponding to $J \subseteq I$ if $H_{\mathcal{C}}$ is a systematic generator matrix for $\mathcal{C}^{\perp}$ in those coordinates.

[8]To emphasize that the code and hidden variable labels in Fig. 13 are in no way related to those labels used previously, the labeling of hidden variables and local constraints begin at S_{12} and $\mathcal{C}_{12}$, respectively.

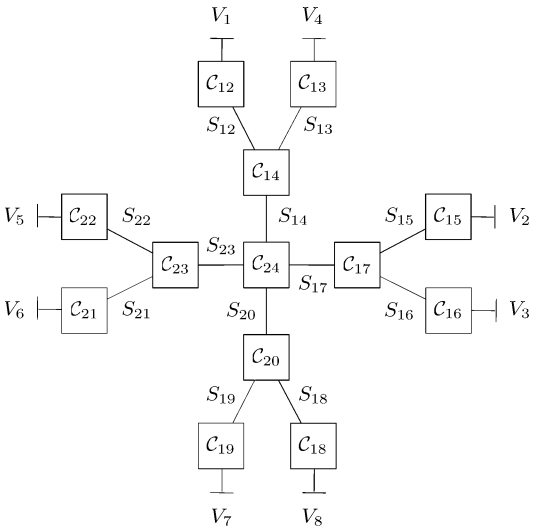

Fig. 13. Cycle-free binary graphical model for $\mathcal{C}_H^{\setminus 9}$. The minimal spanning tree containing the interface constraints that involve V_1, V_2, V_5, and V_8, respectively, is highlighted.

while the remaining local constraints are repetition codes. By construction, it has thus been shown that

$$t\left(\mathcal{C}_H^{\setminus 9}\right) = 1. \tag{84}$$

In light of (82) and (83), a 4-ary graphical model for $\mathcal{C}_H$ can be constructed by updating the graphical model illustrated in Fig. 13 to enforce a single parity-check constraint on V_1, V_2, V_5, and V_8. A natural choice for the root of the minimal spanning tree containing the interface constraints incident on these variables is $\mathcal{C}_{24}$. The updating of the local constraints and hidden variables contained in this spanning tree proceeds as follows. First, note that because $\mathcal{C}_{12}, \mathcal{C}_{15}, \mathcal{C}_{18}$, and $\mathcal{C}_{22}$ simply enforce equality, neither these constraints, nor the hidden variables incident on these constraints, need updating. The hidden variables S_{14}, S_{17}, S_{20}, and S_{23} are updated to be 4-ary so that they send downstream to $\mathcal{C}_{24}$ the values of V_1, V_2, V_8, and V_5, respectively. These hidden variable updates are accomplished by redefining the local constraints $\mathcal{C}_{14}, \mathcal{C}_{17}, \mathcal{C}_{20}$, and $\mathcal{C}_{23}$; the respective generator matrices for the redefined codes are

$$G_{14} = \begin{array}{c} \begin{matrix} S_{12} & S_{13} & S_{14} \end{matrix} \\ \begin{bmatrix} 1 & 0 & 11 \\ 0 & 1 & 10 \end{bmatrix} \end{array} \quad G_{17} = \begin{array}{c} \begin{matrix} S_{15} & S_{16} & S_{17} \end{matrix} \\ \begin{bmatrix} 1 & 0 & 11 \\ 0 & 1 & 10 \end{bmatrix} \end{array} \tag{85}$$

$$G_{20} = \begin{array}{c} \begin{matrix} S_{18} & S_{19} & S_{20} \end{matrix} \\ \begin{bmatrix} 1 & 0 & 11 \\ 0 & 1 & 10 \end{bmatrix} \end{array} \quad G_{23} = \begin{array}{c} \begin{matrix} S_{21} & S_{22} & S_{23} \end{matrix} \\ \begin{bmatrix} 1 & 0 & 10 \\ 0 & 1 & 11 \end{bmatrix} \end{array}. \tag{86}$$

Finally, $\mathcal{C}_{24}$ is updated to enforce both the original repetition constraint on the respective first coordinates of S_{14}, S_{17}, S_{20}, and S_{23} and the additional single parity-check constraint on V_1, V_2, V_5, and V_8 (which correspond to the respective second coordinates of S_{14}, S_{17}, S_{20}, and S_{23}). The generator matrix for the redefined $\mathcal{C}_{24}$ is

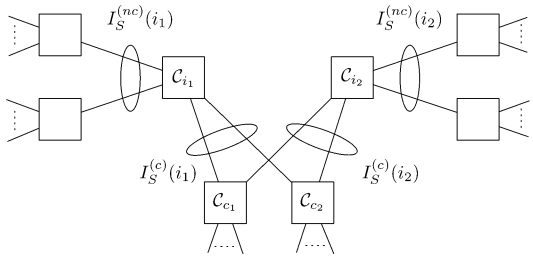

Fig. 14. Local constraint merging notation. The local constraints $\mathcal{C}_{c_1}$ and $\mathcal{C}_{c_2}$ are common.

$$G_{24} = \begin{array}{c} \begin{matrix} S_{14} & S_{17} & S_{20} & S_{23} \end{matrix} \\ \begin{bmatrix} 10 & 10 & 10 & 10 \\ 01 & 00 & 00 & 01 \\ 00 & 01 & 00 & 01 \\ 00 & 00 & 01 & 01 \end{bmatrix} \end{array}. \tag{87}$$

The updated constraints all satisfy the condition for inclusion in a 4-ary graphical model. Specifically, $\mathcal{C}_{24}$ can be decomposed into the Cartesian product of a length 4 binary repetition code and a length 4 binary single parity-check code. The updated graphical model is 4-ary and it has thus been shown by construction that

$$t(\mathcal{C}_H) \leq t\left(\mathcal{C}_H^{\setminus 9}\right) + 1 = 2. \tag{88}$$

D. Graphical Model Transformations

The eight basic graphical model operations introduced in Section VI are discussed in detail in the following where it is assumed that a q^m-ary graphical model $\mathcal{G}_\mathcal{C}$ with behavior $\mathfrak{B}$ for a linear code $\mathcal{C}$ over $\mathbb{F}_q$ defined on an index set I is given.

1) Local Constraint Merging: Suppose that the two local constraints $\mathcal{C}_{i_1}$ and $\mathcal{C}_{i_2}$ shown in Fig. 14 are to be merged. Without loss of generality, assume that there is no hidden variable incident on both $\mathcal{C}_{i_1}$ and $\mathcal{C}_{i_2}$ (because if there is, a degree-2 repetition constraint can be inserted). The hidden variables incident on $\mathcal{C}_{i_1}$ may be partitioned into two sets

$$I_S(i_1) = I_S^{(c)}(i_1) \cup I_S^{(nc)}(i_1) \tag{89}$$

where each $S_j, j \in I_S^{(c)}(i_1)$, is also incident on a constraint $\mathcal{C}_j$ that is adjacent to $\mathcal{C}_{i_2}$. The hidden variables incident on $\mathcal{C}_{i_2}$ may be similarly partitioned. The set of local constraints incident on hidden variables in both $I_S^{(c)}(i_1)$ and $I_S^{(c)}(i_2)$ are denoted *common constraints* and indexed by $I_C^{(c)}(i_1, i_2)$.

The merging of local constraints $\mathcal{C}_{i_1}$ and $\mathcal{C}_{i_2}$ proceeds as follows. For each common local constraint $\mathcal{C}_j, j \in I_C^{(c)}(i_1, i_2)$, denote by $S_{j_1}(S_{j_2})$ the hidden variable incident on $\mathcal{C}_j$ and $\mathcal{C}_{i_1}(\mathcal{C}_{i_2})$. Denote by $\mathcal{C}_{j|\{j_1,j_2\}}$ the projection of $\mathcal{C}_j$ onto the two-variable index set $\{j_1, j_2\}$ and define a new $q^{k(\mathcal{C}_{j|\{j_1,j_2\}})}$-ary hidden variable S_{j_1,j_2}, which encapsulates the possible simultaneous values of S_{j_1} and S_{j_2} (as constrained by $\mathcal{C}_{j|\{j_1,j_2\}}$). After defining such hidden variables for each $\mathcal{C}_j, j \in I_C^{(c)}(i_1, i_2)$, a set of new hidden variables results, which

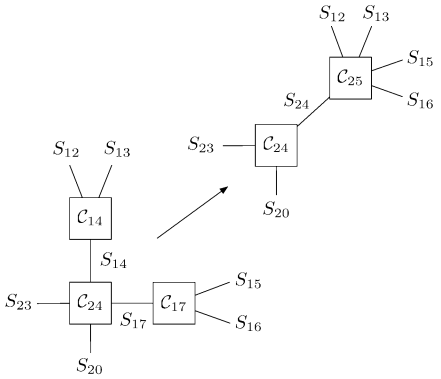

Fig. 15. Merging of constraints $\mathcal{C}_{14}$ and $\mathcal{C}_{17}$ in a 4-ary graphical model for $\mathcal{C}_H$. The resulting graphical model is 8-ary.

is indexed by $I_S^{(c)}(i_1, i_2)$. The local constraints $\mathcal{C}_{i_1}$ and $\mathcal{C}_{i_2}$ are then merged by replacing $\mathcal{C}_{i_1}$ and $\mathcal{C}_{i_2}$ by a code defined over

$$I_V(i_1) \cup I_V(i_2) \cup I_S^{(nc)}(i_1) \cup I_S^{(nc)}(i_2) \cup I_S^{(c)}(i_1, i_2) \quad (90)$$

which is equivalent to $\mathcal{C}_{i_1} \cap \mathcal{C}_{i_2}$ and redefining each local constraint $\mathcal{C}_j, j \in I_C(i_1, i_2)$, over the appropriate hidden variables in $I_S^{(c)}(i_1, i_2)$.

As an example, consider again the 4-ary cycle-free graphical model for $\mathcal{C}_H$ derived in the previous section, a portion of which is reillustrated on the bottom left of Fig. 15, and suppose that the local constraints $\mathcal{C}_{14}$ and $\mathcal{C}_{17}$ are to be merged. The local constraints $\mathcal{C}_{14}, \mathcal{C}_{17}$, and $\mathcal{C}_{24}$ are defined by (85) and (87).

The hidden variables incident on $\mathcal{C}_{14}$ are partitioned into the sets $I_S^{(c)}(14) = \{14\}$ and $I_S^{(nc)}(14) = \{12, 13\}$. Similarly, $I_S^{(c)}(17) = \{17\}$ and $I_S^{(nc)}(17) = \{15, 16\}$. The sole common constraint is thus $\mathcal{C}_{24}$. The projection of $\mathcal{C}_{24}$ onto S_{14} and S_{17} has dimension 3 and the new 8-ary hidden variable S_{24} is defined by the generator matrix

$$G_{14,17} = \begin{array}{c} \begin{matrix} S_{24} & S_{14} & S_{17} \end{matrix} \\ \begin{bmatrix} 100 & 10 & 10 \\ 010 & 01 & 00 \\ 001 & 00 & 01 \end{bmatrix} \end{array}. \quad (91)$$

The local constraints $\mathcal{C}_{14}$ and $\mathcal{C}_{17}$ when defined over S_{24} rather than S_{14} and S_{17}, respectively, are generated by

$$G'_{14} = \begin{array}{c} \begin{matrix} S_{12} & S_{13} & S_{24} \end{matrix} \\ \begin{bmatrix} 1 & 0 & 110 \\ 1 & 0 & 111 \\ 0 & 1 & 100 \end{bmatrix} \end{array} \quad G'_{17} = \begin{array}{c} \begin{matrix} S_{15} & S_{16} & S_{24} \end{matrix} \\ \begin{bmatrix} 1 & 0 & 101 \\ 1 & 0 & 111 \\ 0 & 1 & 100 \end{bmatrix} \end{array}. \quad (92)$$

Finally, $\mathcal{C}_{24}$ is redefined over S_{24} and generated by

$$G_{24} = \begin{array}{c} \begin{matrix} S_{24} & S_{20} & S_{23} \end{matrix} \\ \begin{bmatrix} 100 & 10 & 10 \\ 010 & 00 & 01 \\ 001 & 00 & 01 \\ 000 & 01 & 01 \end{bmatrix} \end{array} \quad (93)$$

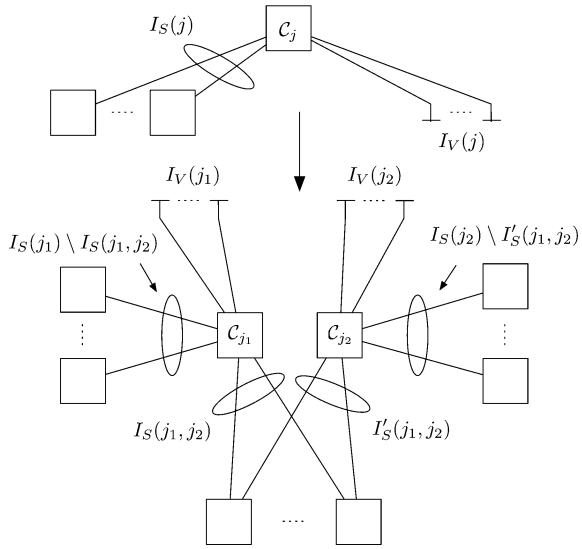

Fig. 16. Splitting of constraint $\mathcal{C}_j$ into $\mathcal{C}_{j_1}$ and $\mathcal{C}_{j_2}$.

while $\mathcal{C}_{14}$ and $\mathcal{C}_{17}$ are replaced by $\mathcal{C}_{25}$, which is equivalent to $\mathcal{C}_{14} \cap \mathcal{C}_{17}$ and is generated by

$$G_{25} = \begin{array}{c} \begin{matrix} S_{24} & S_{12} & S_{13} & S_{15} & S_{16} \end{matrix} \\ \begin{bmatrix} 100 & 0 & 1 & 0 & 1 \\ 010 & 1 & 1 & 0 & 0 \\ 001 & 0 & 0 & 1 & 1 \end{bmatrix} \end{array}. \quad (94)$$

Note that the graphical model that results from the merging of $\mathcal{C}_{14}$ and $\mathcal{C}_{17}$ is 8-ary. Specifically, S_{24} is an 8-ary hidden variable while $n(\mathcal{C}_{24}) - k(\mathcal{C}_{24}) = 3$ and $k(\mathcal{C}_{25}) = 3$.

2) Local Constraint Splitting: Local constraint splitting is simply the inverse operation of local constraint merging. Consider the local constraint $\mathcal{C}_j$ illustrated in Fig. 16, which is defined on the visible and hidden variables indexed by $I_V(j)$ and $I_S(j)$, respectively. Suppose that $\mathcal{C}_j$ is to be split into two local constraints $\mathcal{C}_{j_1}$ and $\mathcal{C}_{j_2}$ defined on the index sets $I_V(j_1) \cup I_S(j_1)$ and $I_V(j_2) \cup I_S(j_2)$, respectively, such that $I_V(j_1)$ and $I_V(j_2)$ partition $I_V(j)$ while $I_S(j_1) \cup I_S(j_2) = I_S(j)$ but $I_S(j_1)$ and $I_S(j_2)$ need not be disjoint. Denote by $I_S(j_1, j_2)$ the intersection of $I_S(j_1)$ and $I_S(j_2)$. Local constraint splitting proceeds as follows. For each $S_i, i \in I_S(j_1, j_2)$, make a copy S'_i of S_i and redefine the local constraint incident on S_i (which is not $\mathcal{C}_j$) over both S_i and S'_i. Denote by $I'_S(j_1, j_2)$ an index set for the copied hidden variables. The local constraint $\mathcal{C}_j$ is then replaced by $\mathcal{C}_{j_1}$ and $\mathcal{C}_{j_2}$ such that $\mathcal{C}_{j_1}$ is defined over $I_V(j_1) \cup I_S(j_1)$ and $\mathcal{C}_{j_2}$ is defined over $I_V(j_1) \cup I_S(j_2)$ where

$$I_S(j_2) = I'_S(j_1, j_2) \cup (I_S(j) \setminus I_S(j_1)). \quad (95)$$

Following this split procedure, some of the hidden variables in $I_S(j_1, j_2)$ and $I'_S(j_1, j_2)$ may have larger alphabets than necessary. Specifically, if the dimension of the projection of $\mathcal{C}_{j_1}(\mathcal{C}_{j_2})$ onto a variable $S_i, i \in I_S(j_1, j_2)(i \in I'_S(j_1, j_2))$, is smaller than the alphabet index set size of S_i, then S_i can be redefined with an alphabet index set size equal to that dimension.

The merged code in the example of the previous section $\mathcal{C}_{25}$ can be split into two codes: $\mathcal{C}_{14}$ defined on S_{12}, S_{13}, and S_{24}, and $\mathcal{C}_{17}$ defined on S_{15}, S_{16}, and S'_{24}. The projection of S_{24} onto

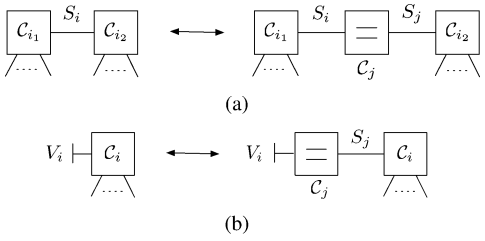

Fig. 17. Insertion and removal of degree-2 repetition constraints.

$\mathcal{C}_{14}$ has dimension 2 and S_{24} can thus be replaced by the 4-ary hidden variable S_{14}. Similarly, the projection of S'_{24} onto $\mathcal{C}_{17}$ has dimension 2 and S'_{24} can be replaced by the 4-ary hidden variable S_{17}.

3) Insertion/Removal of Degree-2 Repetition Constraints: Suppose that S_i is a hidden variable involved in the local constraints $\mathcal{C}_{i_1}$ and $\mathcal{C}_{i_2}$. A degree-2 repetition constraint is inserted by defining a new hidden variable S_j as a copy of S_i, redefining $\mathcal{C}_{i_2}$ over S_j and defining the repetition constraint $\mathcal{C}_j$, which enforces $S_i = S_j$. Degree-2 repetition constraint insertion can be similarly defined for visible variables. Conversely, suppose that $\mathcal{C}_j$ is a degree-2 repetition constraint incident on the hidden variables S_i and S_j. Because $\mathcal{C}_j$ simply enforces $S_i = S_j$, it can be removed and S_j relabeled S_i. Degree-2 repetition constraint removal can be similarly defined for visible variables. The insertion and removal of degree-2 repetition constraints is illustrated in Fig. 17(a) and (b) for hidden and visible variables, respectively.

4) Insertion/Removal of Trivial Constraints: Trivial constraints are those incident on no hidden or visible variables so that their respective block lengths and dimensions are zero. Trivial constraints can obviously be inserted or removed from graphical models.

5) Insertion/Removal of Isolated Partial Parity-Check Constraints: Suppose that $\mathcal{C}_{i_1}, \ldots, \mathcal{C}_{i_j}$ are jq-ary repetition constraints (that is each repetition constraint enforces equality on q-ary variables) and let $\beta_{i_1}, \ldots, \beta_{i_j} \in \mathbb{F}_q$ be nonzero. The insertion of an isolated partial parity-check constraint is defined as follows. Define $j+1$ new q-ary hidden variables $S_{i_1}, \ldots, S_{i_j}$, and S_k, and two new local constraints $\mathcal{C}_p$ and $\mathcal{C}_k$ such that $\mathcal{C}_p$ enforces the q-ary single parity-check equation

$$\sum_{l=1}^{j} \beta_{i_l} S_{i_l} = S_k \tag{96}$$

and $\mathcal{C}_k$ is a degree-1 constraint incident only on S_k with dimension 1. Note that the new hidden variable S_{i_l} is involved in $\mathcal{C}_{i_l}$ and $\mathcal{C}_p$ (for $l = 1, \ldots, j$), while the new hidden variable S_k is involved in $\mathcal{C}_p$ and $\mathcal{C}_k$. The new local constraint $\mathcal{C}_p$ defines the partial parity variable S_k and is denoted *isolated* because it is incident on a hidden variable which is involved in a degree-1, dimension 1 local constraint (i.e., $\mathcal{C}_k$ does not constrain the value of S_k). Because $\mathcal{C}_p$ is isolated, the graphical model that results from its insertion is indeed a valid model for $\mathcal{C}$. Similarly, any such isolated partial parity-check constraint can be removed from a graphical model resulting in a valid model for $\mathcal{C}$.

As an example, Fig. 18 illustrates the insertion and removal of an isolated partial parity-check on the binary sum of V_7 and V_8 in a Tanner graph for $\mathcal{C}_H$ corresponding to (72) [note that $\mathcal{C}_H$ is self-dual so that the generator matrix defined in (72) is also a valid parity-check matrix for $\mathcal{C}_H$].

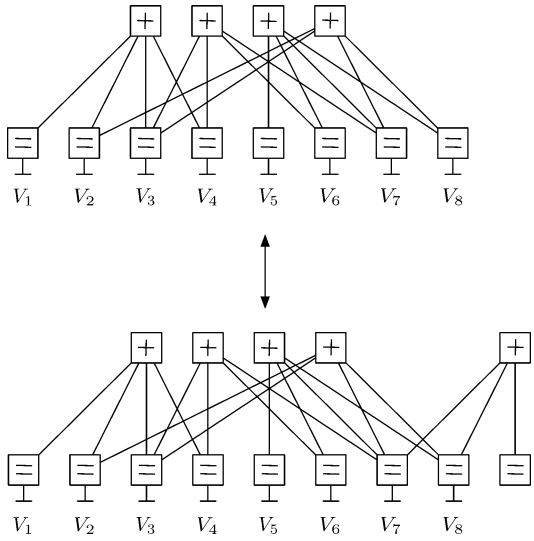

Fig. 18. Insertion/removal of an isolated partial parity-check constraint on V_7 and V_8 in a Tanner graph for $\mathcal{C}_H$.

ACKNOWLEDGMENT

The authors would like to the anonymous reviewers for their help in clarifying the presentation of this work, particularly, the development of Section III-C. They would also like to thank G. D. Forney, Jr. for his helpful comments on early drafts and N. Kashyap for discussions pertaining to Section V-C.

REFERENCES

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," in *Proc. Int. Conf. Commun.*, Geneva, Switzerland, May 1993, pp. 1064–1070.
[2] R. G. Gallager, "Low density parity check codes," *IEEE Trans. Inf. Theory*, vol. 8, no. 1, pp. 21–28, Jan. 1962.
[3] M. Sipser and D. A. Spielman, "Expander codes," *IEEE Trans. Inf. Theory*, vol. 42, no. 6, pp. 1660–1686, Nov. 1996.
[4] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *Inst. Electr. Eng. Electron. Lett.*, vol. 33, pp. 457–458, Mar. 1997.
[5] N. Wiberg, H.-A. Loeliger, and R. Kötter, "Codes and iterative decoding on general graphs," *Eur. Trans. Telecommun.*, vol. 6, no. 5, pp. 513–525, Sept.–Oct. 1995.
[6] N. Wiberg, "Codes and decoding on general graphs," Ph.D. dissertation, Dept. Electr. Eng., Linköping Univ., Linköping, Sweden, 1996.
[7] S. M. Aji and R. J. McEliece, "The generalized distributive law," *IEEE Trans. Inf. Theory*, vol. 46, no. 2, pp. 325–343, Mar. 2000.
[8] K. M. Chugg, A. Anastasopoulos, and X. Chen, *Iterative Detection: Adaptivity, Complexity Reduction, and Applications*. Norwell, MA: Kluwer, 2001.
[9] F. Kschischang, B. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.
[10] G. D. Forney, Jr., "Codes on graphs: Normal realizations," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 520–548, Feb. 2001.
[11] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inf. Theory*, vol. IT-27, no. 5, pp. 533–547, Sep. 1981.
[12] Y. Mao and A. H. Banihashemi, "A heuristic search for good low-density parity-check codes at short block lengths," in *Proc. Int. Conf. Commun.*, Helsinki, Finland, Jun. 2001, vol. 1, pp. 41–44.
[13] D. M. Arnold, E. Eleftheriou, and X. Y. Hu, "Progressive edge-growth Tanner graphs," in *Proc. Globecom Conf.*, San Antonion, TX, Nov. 2001, vol. 2, pp. 995–1001.
[14] X.-P. Ge, D. Eppstein, and P. Smyth, "The distribution of loop lengths in graphical models for turbo decoding," *IEEE Trans. Inf. Theory*, vol. 47, no. 6, pp. 2549–2553, Sep. 2001.
[15] T. Tian, C. R. Jones, J. D. Villasenor, and R. D. Wesel, "Selective avoidance of cycles in irregular LDPC code construction," *IEEE Trans. Inf. Theory*, vol. 52, no. 8, pp. 1242–1247, Aug. 2004.

[16] T. R. Halford and K. M. Chugg, "An algorithm for counting short cycles in bipartite graphs," *IEEE Trans. Inf. Theory*, vol. 52, no. 1, pp. 287–292, Jan. 2006.

[17] Y. Kou, S. Lin, and M. P. C. Fossorier, "Low-density parity-check codes based on finite geometries: A rediscovery and new results," *IEEE Trans. Inf. Theory*, vol. 47, no. 7, pp. 2711–2736, Nov. 2001.

[18] H. Tang, J. Xu, S. Lin, and K. A. S. Abdel-Ghaffar, "Codes on finite geometries," *IEEE Trans. Inf. Theory*, vol. 51, no. 2, pp. 572–596, Feb. 2005.

[19] S. Y. Chung, G. D. Forney, Jr., T. J. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit," *IEEE Commun. Lett.*, vol. 5, no. 2, pp. 58–60, Feb. 2001.

[20] T. Richardson, M. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 619–673, Feb. 2001.

[21] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Efficient erasure correcting codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 569–584, Feb. 2001.

[22] P. Oswald and A. Shokrollahi, "Capacity-achieving sequences for the erasure channel," *IEEE Trans. Inf. Theory*, vol. 48, no. 12, pp. 3017–3028, Dec. 2002.

[23] H. D. Pfister, I. Sason, and R. Urbanke, "Capacity-achieving ensembles for the binary erasure channel with bounded complexity," *IEEE Trans. Inf. Theory*, vol. 51, no. 7, pp. 2352–2379, Jul. 2005.

[24] S. Benedetto and G. Montorsi, "Design of parallel concatenated convolutional codes," *IEEE Trans. Commun.*, vol. 44, no. 5, pp. 591–600, May 1996.

[25] S. Dolinar, D. Divsalar, and F. Pollara, "Code performance as a function of block size," Jet Propulsion Laoratories, Pasadena, CA, Tech. Rep. TDA Progr. Rep. 42-133, May 1998.

[26] C. Berrou, "The ten-year-old turbo codes are entering into service," *IEEE Commun. Mag.*, vol. 41, no. 8, pp. 110–116, Aug. 2003.

[27] S. ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Trans. Commun.*, vol. 49, no. 10, pp. 1727–1737, Oct. 2001.

[28] K. M. Chugg, P. Thiennviboon, G. D. Dimou, P. Gray, and J. Melzer, "A new class of turbo-like codes with universally good performance and high-speed decoding," in *Proc. IEEE Military Commun. Conf.*, Atlantic City, NJ, Oct. 2005, pp. 3117–3126.

[29] T. R. Halford, A. J. Grant, and K. M. Chugg, "Which codes have 4-cycle-free Tanner graphs?," *IEEE Trans. Inf. Theory*, vol. 52, no. 9, pp. 4219–4223, Sep. 2006.

[30] A. Vardy, *Handbook of Coding Theory*, V. S. Pless and W. C. Huffman, Eds. Amsterdam, The Netherlands: Elsevier, 1999, ch. Trellis structure of codes.

[31] A. R. Calderbank, G. D. Forney, Jr., and A. Vardy, "Minimal tail-biting trellises: The Golay code and more," *IEEE Trans. Inf. Theory*, vol. 45, no. 5, pp. 1435–1455, Jul. 1999.

[32] R. Koetter and A. Vardy, "The structure of tail-biting trellises: Minimality and basic principles," *IEEE Trans. Inf. Theory*, vol. 49, no. 9, pp. 2081–2105, Sep. 2003.

[33] R. Koetter, "On the representation of codes in Forney graphs," in *Codes, Graphs, and Systems*, R. E. Blahut and R. Koetter , Eds. Boston, MA: Kluwer, Feb. 2002, pp. 425–450.

[34] R. J. McEliece, "On the BCJR trellis for linear block codes," *IEEE Trans. Inf. Theory*, vol. 42, no. 4, pp. 1072–1092, Jul. 1996.

[35] G. D. Forney, Jr., "Codes on graphs: Constraint complexity of cycle-free realizations of linear codes," *IEEE Trans. Inf. Theory*, vol. 49, no. 7, pp. 1597–1610, Jul. 2003.

[36] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. Amsterdam, The Netherlands: North-Holland, 1978.

[37] D. J. C. MacKay, "Relationships between sparse graph codes," in *Proc. Inf.-Based Induction Sci.*, Izu, Japan, Jul. 2000.

[38] J. S. Yedidia, J. Chen, and M. C. Fossorier, "Generating code representations suitable for belief propagation decoding," in *Proc. Allerton Conf. Commun. Control Comput.*, Monticello, IL, Oct. 2002.

[39] A. Lafourcade and A. Vardy, "Lower bounds on trellis complexity of block codes," *IEEE Trans. Inf. Theory*, vol. 41, no. 6, pp. 1938–1954, Nov. 1995.

[40] D. J. Muder, "Minimal trellises for block codes," *IEEE Trans. Inf. Theory*, vol. 34, no. 5, pp. 1049–1053, Sep. 1988.

[41] Y. Berger and Y. Be'ery, "Bounds on the trellis size of linear block codes," *IEEE Trans. Inf. Theory*, vol. 39, no. 1, pp. 203–209, Jan. 1993.

[42] T. Kasami, T. Takata, T. Fujiwara, and S. Lin, "On the optimum bit orders with respect to the state complexity of trellis diagrams for binary linear codes," *IEEE Trans. Inf. Theory*, vol. 39, no. 1, pp. 242–245, Jan. 1993.

[43] T. Kasami, T. Takata, T. Fujiwara, and S. Lin, "On complexity of trellis structure of linear block codes," *IEEE Trans. Inf. Theory*, vol. 39, no. 3, pp. 1057–1064, May 1993.

[44] G. D. Forney, Jr., "Density/length profiles and trellis complexity of linear block codes," *IEEE Trans. Inf. Theory*, vol. 40, no. 6, pp. 1741–1752, Nov. 1994.

[45] G. D. Forney, Jr., "Coset codes II: Binary lattices and related codes," *IEEE Trans. Inf. Theory*, vol. IT-34, no. 5, pt. 1, pp. 1152–1187, Sep. 1988.

[46] A. Lafourcade and A. Vardy, "Asymptotically good codes have infinite trellis complexity," *IEEE Trans. Inf. Theory*, vol. 41, no. 2, pp. 555–559, Mar. 1994.

[47] T. R. Halford and K. M. Chugg, "The tradeoff between cyclic topology and complexity in graphical models of linear codes," in *Proc. Allerton Conf. Commun. Control Comput.*, Sep. 2006.

[48] T. R. Halford and K. M. Chugg, "The complexity limits of graphical models for linear codes," in *Proc. IEEE Inf. Theory Workshop*, Lake Tahoe, CA, Sep. 2007, pp. 144–149.

[49] R. Diestel, *Graph Theory*, 2nd ed. New York: Springer-Verlag, 2000.

[50] T. R. Halford and K. M. Chugg, "Conditionally cycle-free generalized Tanner graphs: Theory and application to high-rate serially concatenated codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Nice, France, Jun. 2007, pp. 1881–1885.

[51] Y. Wang, R. Ramesh, A. Hassan, and H. Koorapaty, "On MAP decoding for tail-biting convolutional codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Ulm, Germany, Jun. 1997, p. 225.

[52] S. M. Aji, G. B. Horn, and R. J. McEliece, "Iterative decoding on graphs with a single cycle," in *Proc. IEEE Int. Symp. Inf. Theory*, Cambridge, MA, Aug. 1998, p. 276.

[53] J. B. Anderson and S. M. Hladik, "Tailbiting MAP decoders," *IEEE J. Sel. Areas Commun.*, vol. 16, no. 2, pp. 297–302, Feb. 1998.

[54] J. Heo and K. M. Chugg, "Constrained iterative decoding: Performance and convergence analysis," in *Proc. Asilomar Conf. Signals Syst. Comput.*, Pacific Grove, CA, Nov. 2001, pp. 275–279.

[55] Y. Shany and Y. Be'ery, "Linear tail-biting trellises, the square-root bound, and applications for Reed-Muller codes," *IEEE Trans. Inf. Theory*, vol. 46, no. 4, pp. 1514–1523, Jul. 2000.

[56] T. Etzion, A. Trachtenberg, and A. Vardy, "Which codes have cycle-free Tanner graphs?," *IEEE Trans. Inf. Theory*, vol. 45, no. 6, pp. 2173–2181, Sep. 1999.

[57] J. K. Wolf, "Efficient maximum-likelihood decoding of linear block codes using a trellis," *IEEE Trans. Inf. Theory*, vol. IT-24, no. 1, pp. 76–80, Jan. 1978.

[58] N. Kashyap, "On minimal tree realizations of linear codes," *IEEE Trans. Inf. Theory* 2007 [Online]. Available: http://arxiv.org/abs/0711.1383, submitted for publication

[59] S. Lin, T. Kasami, T. Fujiwara, and M. Fossorier, *Trellises and Trellis-Based Decoding Algorithms for Linear Block Codes*. Norwell, MA: Kluwer, 1998.

[60] J. Jiang and K. R. Narayanan, "Iterative soft decision decoding of Reed-Solomon codes," *IEEE Commun. Lett.*, vol. 8, no. 4, pp. 244–246, Apr. 2004.

[61] T. R. Halford and K. M. Chugg, "Random redundant soft-in soft-out decoding of linear block codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Seattle, WA, Jul. 2006, pp. 2230–2234.

[62] S. Sankaranarayanan and B. Vasić, "Iterative decoding of linear block codes: A parity-check orthogonalization approach," *IEEE Trans. Inf. Theory*, vol. 51, no. 9, pp. 3347–3353, Sep. 2005.

[63] V. Kumar and O. Milenkovic, "On graphical representations for algebraic codes suitable for iterative decoding," *IEEE Commun. Lett.*, vol. 9, no. 8, pp. 729–731, Aug. 2005.

[64] N. Kashyap, "A decomposition theory for binary linear codes," *IEEE Trans. Inf. Theory.* 2006 [Online]. Available: http://arxiv.org/abs/cs/0611028, submitted for publication

[65] P. O. Vontobel and R. Koetter, "Graph-cover decoding and finite-length analysis of message-passing iterative decoding of LDPC codes," *IEEE Trans. Inf. Theory*, 2005, submitted for publication.

[66] T. Brack, M. Alles, T. Lehnigk-Emden, F. Kienle, N. Wehn, N. E. L'Insalata, F. Rossi, M. Rovini, and L. Fanucci, "Low complexity LDPC code decoders for next generation standards," in *Proc. Design Autom. Test Eur.*, Nice, France, Apr. 2007, pp. 16–20.

[67] S. Laendner, T. Hehn, O. Milenkovic, and J. B. Huber, "When does one redundant parity-check equation matter?," in *Proc. Globecom Conf.*, San Francisco, CA, Nov. 2006, pp. 1–6.

[68] Y. Han, Y. Zang, and W. E. Ryan, "Toward low floors in LDPC and G-LDPC codes," presented at the IEEE Commun. Theory Workshop, Sedona, AZ, May 2007.