ON MINIMAL TREE REALIZATIONS OF LINEAR CODES

NAVIN KASHYAP

ABSTRACT. A tree decomposition of the coordinates of a code is a mapping from the coordinate set to the set of vertices of a tree. A tree decomposition can be extended to a tree realization, *i.e.*, a cycle-free realization of the code on the underlying tree, by specifying a state space at each edge of the tree, and a local constraint code at each vertex of the tree. The constraint complexity of a tree realization is the maximum dimension of any of its local constraint codes. A measure of the complexity of maximum-likelihood decoding for a code is its treewidth, which is the least constraint complexity of any of its tree realizations.

It is known that among all tree realizations of a linear code that extends a given tree decomposition, there exists a unique minimal realization that minimizes the state space dimension at each vertex of the underlying tree. In this paper, we give two new constructions of these minimal realizations. As a by-product of the first construction, a generalization of the state-merging procedure for trellis realizations, we obtain the fact that the minimal tree realization also minimizes the local constraint code dimension at each vertex of the underlying tree. The second construction relies on certain code decomposition techniques that we develop. We further observe that the treewidth of a code is related to a measure of graph complexity, also called treewidth. We exploit this connection to resolve a conjecture of Forney's regarding the gap between the minimum trellis constraint complexity and the treewidth of a code. We present a family of codes for which this gap can be arbitrarily large.

1. INTRODUCTION

Graphical models of codes and the decoding algorithms associated with them are now a major focus area of research in coding theory. Turbo codes, low-density parity-check (LDPC) codes, and expander codes are all examples of codes defined, in one way or another, on underlying graphs. A unified treatment of graphical models and the associated decoding algorithms began with the work of Wiberg, Loeliger and Koetter [32],[33], and has since been abstracted and refined under the framework of the generalized distributive law [1], factor graphs [21], and normal realizations [7],[8]. The particular case of graphical models in which the underlying graphs are cycle-free has a long and rich history of its own, starting with the study of trellis representations of codes; see *e.g.*, [31] and the references therein.

Briefly, a graphical model consists of a graph, an assignment of symbol variables to the vertices of the graph, an assignment of state variables to the edges of the graph, and a specification of local constraint codes at each vertex of the graph. The full behavior of the model is the set of all configurations of symbol and state variables that satisfy all the local constraints. Such a model is called a realization of a code $\mathcal{C}$ if the restriction of the full behavior to the set of symbol variables is precisely $\mathcal{C}$. The realization is said to be cycle-free if the underlying graph in the model has no cycles. A trellis representation of a code can be viewed as a cycle-free realization in which the underlying graph is a simple path.

A linear code $\mathcal{C}$ has a realization on a graph $\mathcal{G}$ that is not connected if and only if $\mathcal{C}$ can be expressed as the direct sum of the codes that are individually realized on the connected components

Date: February 10, 2009.
This work was supported by a Discovery Grant from the Natural Sciences and Engineering Research Council (NSERC), Canada.
The author is with the Department of Mathematics and Statistics, Queen's University, Kingston, ON K7L 3N6, Canada. Email: nkashyap@mast.queensu.ca.

of $\mathcal{G}$ [7]. Thus, there is no loss of generality in just focusing, as we do, on the case of realizations on connected graphs. In this paper, we will be concerned with tree realizations — cycle-free realizations in which the underlying cycle-free graph is connected, *i.e.*, is a tree.

It is by now well known that the sum-product algorithm on any tree realization provides an exact implementation of maximum-likelihood (ML) decoding [1],[7],[21],[32]. A good initial estimate of the computational complexity of such an implementation is given by the constraint complexity of the realization, which is the maximum dimension of any of the local constraint codes in the realization. Now, distinct tree realizations of the same code have, in general, distinct constraint complexities. The treewidth of a code is defined to be the least constraint complexity of any of its tree realizations. Thus, treewidth may be taken to be a measure of the ML decoding complexity of a code.

Since trellis realizations are instances of tree realizations, the treewidth of a code can be no larger than the minimum constraint complexity[1] of any of its trellis realizations. In the abstract of his paper [8], Forney claimed that "the constraint complexity of a general cycle-free graph realization can be [strictly] less than that of any conventional trellis realization, but not by very much." While he substantiated the first part of his claim by means of an example, he left the "not by very much" part as a conjecture [8, p. 1606, Conjecture 2]. But he also admitted that none of the arguments he gave in support of his conjecture "is very persuasive," and that it is equally plausible that [8, Conjecture 3] there exists no upper bound on the gap between the treewidth of a code and the minimum constraint complexity of any of its trellis realizations.

One of the main contributions of this paper is an example that affirms the validity of Forney's Conjecture 3. We present, in Section 6, a family of codes for which the difference between the minimum trellis constraint complexity and the treewidth grows logarithmically with the length of the code. Our construction of this example is based upon results from the graph theory and matroid theory literatures that connect the notions of treewidth and trellis complexity of a code to certain complexity measures defined for graphs.

This paper makes two other contributions, both relating to minimal tree realizations. A mapping of the set of coordinates of a code $\mathcal{C}$ to the vertices of a tree is called a tree decomposition. A tree decomposition may be viewed as an assignment of symbol variables to the vertices of the tree. It is known that given a code $\mathcal{C}$, among all tree realizations of $\mathcal{C}$ that extend a given tree decomposition, there is one that minimizes the state space dimension at each edge of the underlying tree [7]. This minimal tree realization, an explicit construction of which was also given in [7], is unique up to isomorphism.

We give two new constructions of minimal tree realizations. The first construction involves a generalization of the idea of state merging that can be used to construct minimal trellis realizations [31, Section 4]. We show that any tree realization of a code can be converted to a minimal realization by a sequence of state merging transformations. The state space and constraint code dimensions do not increase at any step of this process. From this, we obtain the fact that a minimal realization also minimizes the constraint code dimension at each vertex of the underlying tree.

Our second construction of minimal tree realizations uses extensions of the code decomposition techniques that were presented in [18]. The main advantage of this construction is its recursive nature, which makes it suitable for mechanical implementation. Also, it is relatively straightforward to estimate the computational complexity of this construction. We show that the complexity is polynomial in the length and dimension of the code, as well as in the size of the underlying tree, but is exponential in the state-complexity of the minimal realization, which is the maximum dimension of any state space in the realization.

[1]In the context of trellis realizations, constraint complexity is usually referred to as "branch complexity" or "edge complexity". We make it a point to avoid this usage, so as not to cause confusion when we define the "branchwidth" of a code later in our paper.

The paper is organized as follows. In Section 2, we provide the necessary background on tree realizations of linear codes. The construction of minimal realizations by means of state merging is presented in Section 3. Code decomposition techniques are developed in Section 4, and used in Section 5 to derive a recursive construction of minimal tree realizations. Proofs of some of the results from Sections 2–5 and an example relevant to Section 3 are deferred to appendices to preserve the flow of the exposition. Treewidth and related complexity measures are defined in Section 6, which also establishes connections between these code complexity measures and certain complexity measures defined for graphs. These connections are used to derive the example of a code family for which the gap between minimum trellis constraint complexity and treewidth is arbitrarily large. We also touch upon the subject of codes of bounded complexity, observing that many hard coding-theoretic problems become polynomial-time solvable when restricted to code families whose treewidth is bounded. Section 7 contains a few concluding remarks.

2. Background on Tree Realizations

Our treatment of the topic of tree realizations in this section is based on the exposition of Forney [7],[8]; see also [9].

We start by establishing some basic notation. We take $\mathbb{F}$ to be an arbitrary finite field. Given a finite index set I, we have the vector space $\mathbb{F}^I = \{\mathbf{x} = (x_i \in \mathbb{F},\ i \in I)\}$. For $\mathbf{x} \in \mathbb{F}^I$ and $J \subseteq I$, the notation $\mathbf{x}|_J$ will denote the *projection* $(x_i,\ i \in J)$. Also, for $J \subseteq I$, we will find it convenient to reserve the use of $\overline{J}$ to denote the set difference $I - J = \{i \in I :\ i \notin J\}$.

2.1. **Codes.** A *linear code* over $\mathbb{F}$, defined on the index set I, is a subspace $\mathcal{C} \subseteq \mathbb{F}^I$. We will only consider linear codes in this paper, so the terms "code" and "linear code" will be used interchangeably. The dimension, over $\mathbb{F}$, of $\mathcal{C}$ will be denoted by $\dim(\mathcal{C})$. An $[n, k]$ code is a code of length n and dimension k. If, additionally, the code has minimum distance d, then the code is an $[n, k, d]$ code. The dual code of $\mathcal{C}$ is denoted by $\mathcal{C}^\perp$, and is defined on the same index set as $\mathcal{C}$.

Let J be a subset of the index set I. The *projection* of $\mathcal{C}$ onto J is the code $\mathcal{C}|_J = \{\mathbf{c}|_J : \mathbf{c} \in \mathcal{C}\}$, which is a subspace of $\mathbb{F}^J$. We will use $\mathcal{C}_J$ to denote the *cross-section* of $\mathcal{C}$ consisting of all projections $\mathbf{c}|_J$ of codewords $\mathbf{c} \in \mathcal{C}$ that satisfy $\mathbf{c}|_{\overline{J}} = \mathbf{0}$. To be precise, $\mathcal{C}_J = \{\mathbf{c}|_J :\ \mathbf{c} \in \mathcal{C}, \mathbf{c}|_{\overline{J}} = \mathbf{0}\}$. Note that $\mathcal{C}_J \subseteq \mathcal{C}|_J$. Also, since $\mathcal{C}_J$ is isomorphic to the kernel of the projection map $\pi : \mathcal{C} \to \mathcal{C}|_{\overline{J}}$ defined by $\pi(\mathbf{c}) = \mathbf{c}|_{\overline{J}}$, we have that $\dim(\mathcal{C}_J) = \dim(\mathcal{C}) - \dim(\mathcal{C}|_{\overline{J}})$. Furthermore, projections and cross-sections are dual notions, in the sense that $(\mathcal{C}|_J)^\perp = (\mathcal{C}^\perp)_J$, and similarly, $(\mathcal{C}_J)^\perp = (\mathcal{C}^\perp)|_J$.

If $\mathcal{C}_1$ and $\mathcal{C}_2$ are codes over $\mathbb{F}$ defined on mutually disjoint index sets I_1 and I_2, respectively, then their *direct sum* is the code $\mathcal{C} = \mathcal{C}_1 \oplus \mathcal{C}_2$ defined on the index set $I_1 \cup I_2$, such that $\mathcal{C}_{I_1} = \mathcal{C}|_{I_1} = \mathcal{C}_1$ and $\mathcal{C}_{I_2} = \mathcal{C}|_{I_2} = \mathcal{C}_2$. This definition naturally extends to multiple codes (or subspaces) $\mathcal{C}_\alpha$, where α is a code identifier that takes values in some set A. Again, it must be assumed that the codes $\mathcal{C}_\alpha$ are defined on mutually disjoint index sets $I_\alpha,\ \alpha \in A$. The direct sum in this situation is denoted by $\bigoplus_{\alpha \in A} \mathcal{C}_\alpha$.

2.2. **Trees.** A tree is a connected graph without cycles. Given a tree T, we will denote its vertex and edge sets by $V(T)$ and $E(T)$, respectively, or simply by V and E if there is no ambiguity. Vertices of degree one are called *leaves*, and all other vertices are called *internal nodes*. Given a $v \in V$, the set of edges incident with v will be denoted by $E(v)$.

Removal of an arbitrary edge e from T produces a disconnected graph $T - e$, which is the disjoint union of two subtrees, which we will denote by T_e and $\overline{T}_e$, of T. Note that $V(T_e)$ and $V(\overline{T}_e)$ form a partition of $V(T)$.

2.3. **Tree Realizations.** Let $\mathcal{C}$ be a code over $\mathbb{F}$, defined on the index set I. To each $i \in I$, we associate a *symbol variable* X_i, which is allowed to take values in $\mathbb{F}$.

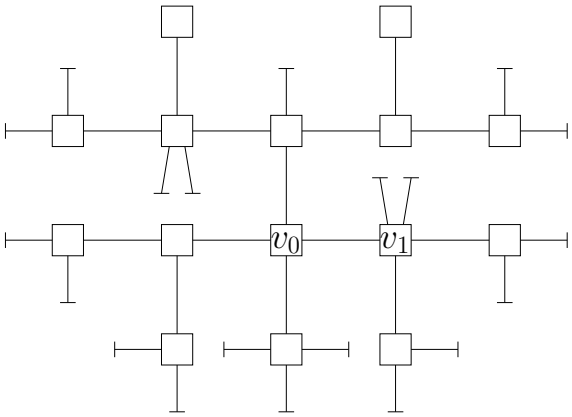

FIGURE 1. A depiction of a tree decomposition (T, ω). Vertices of the tree T are represented by squares. Edges are incident with two vertices, while half-edges are incident with only one vertex. The vertex v_0 has no half-edges incident with it, indicating that $\omega^{-1}(v_0) = \emptyset$, while the vertex v_1 has two half-edges incident with it, which means that $|\omega^{-1}(v_1)| = 2$.

A *tree decomposition* of I is a pair (T, ω), where T is a tree (*i.e.*, a connected, cycle-free graph) and $\omega : I \to V$ is a mapping from I to the vertex set of T. Pictorially, a tree decomposition (T, ω) is depicted as a tree with an additional feature: at each vertex v such that $\omega^{-1}(v)$ is non-empty, we attach special "half-edges", one for each index in $\omega^{-1}(v)$; see Figure 1.

At this point, we introduce some notation that we will consistently use in the rest of the paper. Given a tree decomposition (T, ω) of an index set I, and an edge $e \in E$, we define $J(e) = \omega^{-1}(V(T_e))$ and $\overline{J}(e) = \omega^{-1}(V(\overline{T}_e))$. Thus, $J(e)$ and $\overline{J}(e)$ are the subsets of I that get mapped by ω to vertices in T_e and $\overline{T}_e$, respectively. Clearly, $J(e)$ and $\overline{J}(e)$ form a partition of I.

Recall that $E(v)$, $v \in V$, denotes the set of edges incident with v in T. Consider a tuple of the form $(T, \omega, (\mathcal{S}_e,\ e \in E), (C_v,\ v \in V))$, where

- (T, ω) is a tree decomposition of I;
- for each $e \in E$, $\mathcal{S}_e$ is a vector space over $\mathbb{F}$ called a *state space*;
- for each $v \in V$, C_v is a subspace of $\mathbb{F}^{\omega^{-1}(v)} \oplus \left(\bigoplus_{e \in E(v)} \mathcal{S}_e\right)$, called a *local constraint code*, or simply, a *local constraint*.

Such a tuple will be called a *tree model*. The elements of any state space $\mathcal{S}_e$ are called *states*. The index sets of the state spaces $\mathcal{S}_e$, $e \in E$, are taken to be mutually disjoint, and are also taken to be disjoint from the index set I corresponding to the symbol variables. Finally, to each $e \in E$, we associate a *state variable* S_e that takes values in the corresponding state space $\mathcal{S}_e$.

A *global configuration* of a tree model as above is an assignment of values to each of the symbol and state variables. In other words, it is a vector of the form $((x_i \in \mathbb{F},\ i \in I), (\mathbf{s}_e \in \mathcal{S}_e,\ e \in E))$. A global configuration is said to be *valid* if it satisfies all the local constraints. Thus, $((x_i \in \mathbb{F},\ i \in I), (\mathbf{s}_e \in \mathcal{S}_e,\ e \in E))$ is a valid global configuration if for each $v \in V$, $((x_i,\ i \in \omega^{-1}(v)), (\mathbf{s}_e,\ e \in E(v))) \in C_v$. The set of all valid global configurations of a tree model is called the *full behavior* of the model.

Note that the full behavior is a subspace $\mathfrak{B} \subseteq \mathbb{F}^I \oplus \left(\bigoplus_{e \in E} \mathcal{S}_e\right)$. As usual, $\mathfrak{B}|_I$ denotes the projection of $\mathfrak{B}$ onto the index set I. If $\mathfrak{B}|_I = \mathcal{C}$, then the model $(T, \omega, (\mathcal{S}_e,\ e \in E), (C_v,\ v \in V))$ is called a *(linear) tree realization* of $\mathcal{C}$. A tree realization $(T, \omega, (\mathcal{S}_e,\ e \in E), (C_v,\ v \in V))$ of $\mathcal{C}$ is said to *extend* (or be an extension of) the tree decomposition (T, ω) of the index set of $\mathcal{C}$. Any tree decomposition of the index set of a code can always be extended to a tree realization of the code, as explained in the following example.

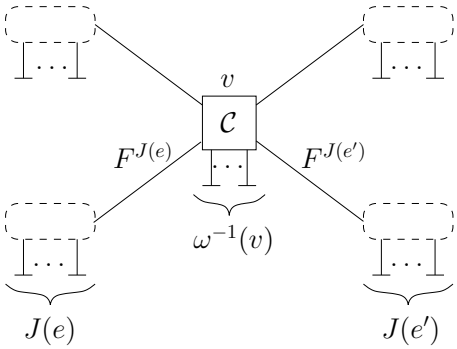

FIGURE 2. A trivial extension of a tree decomposition (T, ω) of the index set of a code $\mathcal{C}$. At the vertex v, we have $C_v = \mathcal{C}$. The state variables at the edges $e \in E(v)$ are copies of the symbol variables indexed by $J(e)$. Dashed ovals represent subtrees.

Example 2.1. *Let $\mathcal{C}$ be a code defined on index set I, and let (T, ω) be a tree decomposition of I. Pick an arbitrary $v \in V$, and define $C_v = \mathcal{C}$. Now, consider the set, $E(v)$, of edges incident with v. Removal of any $e \in E(v)$ produces the two subtrees T_e and $\overline{T}_e$. We specify T_e to be the subtree that does* not *contain the vertex v, and as usual, $J(e) = \omega^{-1}(V(T_e))$. For each $e \in E(v)$, the state space $\mathcal{S}_e$ is taken to be a copy of $\mathbb{F}^{J(e)}$. The remaining state spaces and local constraints are chosen so that, for each $e \in E(v)$, the symbol variables indexed by $J(e)$ simply get relayed (unchanged) to the state variable S_e; see Figure 2. It should be clear that the resulting tree model is a tree realization of the code $\mathcal{C}$. This will be called a* trivial extension[2] *of (T, ω). We will present constructions of non-trivial extensions of tree decompositions a little later.*

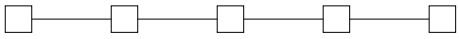

FIGURE 3. A simple path on five vertices.

Example 2.2. *A* simple path *is a tree with exactly two leaves (the end-points of the path), in which all internal nodes have degree two; see Figure 3. Let $\mathcal{C}$ be a code defined on index set I, and let (T, ω) be a tree decomposition of I, in which T is a simple path, and ω is a surjective map $\omega : I \to V(T)$. Any tree realization of $\mathcal{C}$ that extends (T, ω) is called a* trellis realization *of $\mathcal{C}$. When ω is a bijection, then any trellis realization extending (T, ω) is called a* conventional *trellis realization. When ω is not a bijection (but still a surjection), a trellis realization that extends (T, ω) is called a* sectionalized *trellis realization. In trellis terminology, the local constraint codes in a trellis realization are called* branch spaces. *The theory of trellis realizations is well established; we refer the reader to* [31] *for an excellent survey of this theory.*

Let $\mathfrak{B}$ be the full behavior of a tree model $(T, \omega, (\mathcal{S}_e,\ e \in E), (C_v,\ v \in V))$. We will find it useful to define certain projections of $\mathfrak{B}$, other than $\mathfrak{B}|_J$ for $J \subseteq I$. Let $\mathbf{b} = ((x_i,\ i \in I),\ (\mathbf{s}_e,\ e \in E))$ be a global configuration in $\mathfrak{B}$. At any given $v \in V$, the *local configuration* of $\mathbf{b}$ at v is defined as

$$\mathbf{b}|_v = ((x_i,\ i \in \omega^{-1}(v)),\ (\mathbf{s}_e,\ e \in E(v))).$$

The set of all local configurations of $\mathfrak{B}$ at v is then defined as $\mathfrak{B}|_v = \{\mathbf{b}|_v : \mathbf{b} \in \mathfrak{B}\}$. Recall that, by definition, $\mathbf{b}$ is a valid global configuration if, for each $v \in V$, we have $\mathbf{b}|_v \in C_v$. Thus, $\mathfrak{B}|_v \subseteq C_v$ for each $v \in V$. Similarly, for $F \subseteq E$, and $\mathbf{b}$ as above, we define the projections $\mathbf{b}|_F = (\mathbf{s}_e,\ e \in F)$ and $\mathfrak{B}|_F = \{\mathbf{b}|_F : \mathbf{b} \in \mathfrak{B}\}$. Clearly, $\mathfrak{B}|_F$ is a subspace of $\bigoplus_{e \in F} \mathcal{S}_e$. If F

[2]This generalizes the notion of a "trivial trellis" in [23],[24].

consists of a single edge e, then we simply denote the corresponding projections by $\mathbf{b}|_e$ and $\mathfrak{B}|_e$. The following elementary property of the projections $\mathbf{b}|_e$ will be useful later; a proof for it is given in Appendix A.

Lemma 2.1. *Let $\mathfrak{B}$ be the full behavior of some tree realization of a code $\mathcal{C}$, defined on the index set I, that extends the tree decomposition (T, ω). Suppose that $\mathbf{b} \in \mathfrak{B}$ and $e \in E$ are such that $\mathbf{b}|_e = \mathbf{0}$. Then, $\mathbf{b}|_I \in \mathcal{C}_{J(e)} \oplus \mathcal{C}_{\overline{J}(e)}$.*

A tree model (or realization) $(T, \omega, (\mathcal{S}_e,\ e \in E), (C_v,\ v \in V))$, with full behavior $\mathfrak{B}$, is said to be *essential* if $\mathfrak{B}|_e = \mathcal{S}_e$ for all $e \in E$. This definition actually implies something more.

Lemma 2.2. *If the tree model $(T, \omega, (\mathcal{S}_e,\ e \in E), (C_v,\ v \in V))$, with full behavior $\mathfrak{B}$, is essential, then $\mathfrak{B}|_v = C_v$ for all $v \in V$.*

A proof of the lemma can be found in Appendix A.

An arbitrary tree model can always be "essentialized". To see this, let $\Gamma = (T, \omega, (\mathcal{S}_e,\ e \in E), (C_v,\ v \in V))$ be a tree model with full behavior $\mathfrak{B}$. Recall that $\mathfrak{B}|_e$ is a subspace of $\mathcal{S}_e$, and $\mathfrak{B}|_v$ is a subspace of C_v. Define the *essentialization* of Γ to be the tree model $\mathsf{ess}(\Gamma) = (T, \omega, (\mathfrak{B}|_e,\ e \in E), (\mathfrak{B}|_v,\ v \in V))$. It is readily verified that $\mathsf{ess}(\Gamma)$ has the same full behavior as Γ.

2.4. **Minimal Tree Realizations.** Given a code $\mathcal{C}$ and a tree decomposition (T, ω) of its index set I, there exists an essential tree realization, $(T, \omega, (\mathcal{S}^*_e,\ e \in E), (C^*_v,\ v \in V))$, of $\mathcal{C}$ with the following property [7],[8]:

> if $(T, \omega, (\mathcal{S}_e,\ e \in E), (C_v,\ v \in V))$ is a tree realization of $\mathcal{C}$ that extends (T, ω),
> then for all $e \in E$, $\dim(\mathcal{S}^*_e) \leq \dim(\mathcal{S}_e)$.

This *minimal* tree realization, which we henceforth denote by $\mathcal{M}(\mathcal{C}; T, \omega)$, is unique up to isomorphism. More precisely, if $(T, \omega, (\mathcal{S}^{**}_e,\ e \in E), (C^{**}_v,\ v \in V))$ is also a tree realization of $\mathcal{C}$ with the above property (except that $\mathcal{S}^*_e$ is replaced by $\mathcal{S}^{**}_e$), then $\mathcal{S}^*_e \cong \mathcal{S}^{**}_e$ for each $e \in E$, and $C^*_v \cong C^{**}_v$ for each $v \in V$. We will not distinguish between isomorphic tree realizations.

We outline a construction, due to Forney [8], of $\mathcal{M}(\mathcal{C}; T, \omega)$. For any edge $e \in E$, the sets $J(e)$ and $\overline{J}(e)$ form a partition of the index set I. Set

$$\mathcal{S}^*_e = \mathcal{C}/(\mathcal{C}_{J(e)} \oplus \mathcal{C}_{\overline{J}(e)}), \tag{1}$$

and let

$$\mathbf{s}^*_e :\ \mathcal{C} \to \mathcal{C}/(\mathcal{C}_{J(e)} \oplus \mathcal{C}_{\overline{J}(e)}) \tag{2}$$

be the canonical projection map. In other words, for $\mathbf{c} \in \mathcal{C}$, $\mathbf{s}^*_e(\mathbf{c})$ is the coset $\mathbf{c} + (\mathcal{C}_{J(e)} \oplus \mathcal{C}_{\overline{J}(e)})$.

Now, let $\mathfrak{B}$ be the vector space consisting of all global configurations $(\mathbf{c}, \mathbf{s}^*(\mathbf{c}))$ corresponding to codewords $\mathbf{c} \in \mathcal{C}$, where $\mathbf{s}^*(\mathbf{c}) = (\mathbf{s}^*_e(\mathbf{c}),\ e \in E)$. It is worth noting that $\mathfrak{B}|_I = \mathcal{C}$, and furthermore, $\mathfrak{B} \cong \mathcal{C}$, since $\mathbf{c} = \mathbf{0}$ implies that $\mathbf{s}^*(\mathbf{c}) = \mathbf{0}$.

We can now define for each $v \in V$, the local constraint

$$C^*_v = \mathfrak{B}|_v = \left\{ \left(\mathbf{c}|_{\omega^{-1}(v)},\ (\mathbf{s}^*_e(\mathbf{c}),\ e \in E(v)) \right) \ :\ \mathbf{c} \in \mathcal{C} \right\}. \tag{3}$$

The minimal realization $\mathcal{M}(\mathcal{C}; T, \omega)$ is the tuple $(T, \omega, (\mathcal{S}^*_e,\ e \in E), (C^*_v,\ v \in V))$. It may be verified that $\mathfrak{B}$ is the full behavior of $\mathcal{M}(\mathcal{C}; T, \omega)$, so that $\mathcal{M}(\mathcal{C}; T, \omega)$ is indeed an essential tree realization of $\mathcal{C}$.

From the definition of $\mathcal{S}^*_e$ in (1), it is clear that for each $e \in E$,

$$\dim(\mathcal{S}^*_e) = \dim(\mathcal{C}) - \dim(\mathcal{C}_{J(e)}) - \dim(\mathcal{C}_{\overline{J}(e)}). \tag{4}$$

It is useful to point out that $\dim(\mathcal{S}^*_e)$ may also be expressed as

$$\dim(\mathcal{S}^*_e) = \dim(\mathcal{C}|_{J(e)}) + \dim(\mathcal{C}|_{\overline{J}(e)}) - \dim(\mathcal{C}), \tag{5}$$

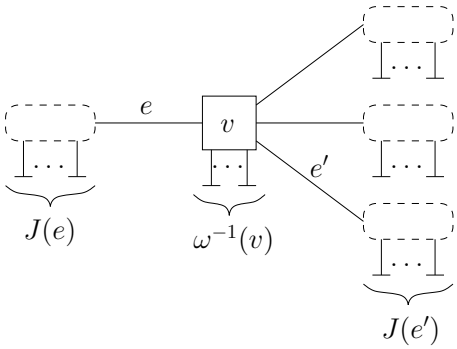

FIGURE 4. Figure depicting the perspective about a vertex v in a tree decomposition. Dashed ovals represent subtrees.

a consequence of the fact that for any $J \subseteq I$, $\dim(\mathcal{C}_J) = \dim(\mathcal{C}) - \dim(\mathcal{C}|_{\overline{J}})$. Thus, by the uniqueness of minimal tree realizations, if $\Gamma^{**} = (T, \omega, (\mathcal{S}_e^{**},\ e \in E), (C_v^{**},\ v \in V))$ is a tree realization of $\mathcal{C}$ with the property that for all $e \in E$, $\dim(\mathcal{S}_e^{**})$ equals one of the expressions in (4) or (5), then Γ^{**} is in fact $\mathcal{M}(\mathcal{C}; T, \omega)$.

Forney [8] also derived an expression for the dimension of the local constraints C_v^*. Consider any $v \in V$. For each $e \in E(v)$, we specify T_e to be the component of $T - e$ that does *not* contain v. As usual, $J(e) = \omega^{-1}(V(T_e))$. Then [8, Theorem 1],

$$\dim(C_v^*) = \dim(\mathcal{C}) - \sum_{e \in E(v)} \dim(\mathcal{C}_{J(e)}). \tag{6}$$

Forney gave the following bound for $\dim(C_v^*)$ [8, Theorem 5]: for any $e \in E(v)$, $\dim(\mathcal{S}_e^*) \leq \dim(C_v^*) \leq \text{len}(C_v^*) - \dim(\mathcal{S}_e^*)$, where $\text{len}(C_v^*)$ denotes the length of the code C_v^*. The upper bound can be improved slightly.

Lemma 2.3. *In the minimal tree realization $\mathcal{M}(\mathcal{C}; T, \omega)$, we have, for $v \in V$ and $e \in E(v)$,*

$$\dim(\mathcal{S}_e^*) \leq \dim(C_v^*) \leq \dim(\mathcal{C}|_{\omega^{-1}(v)}) + \sum_{e' \in E(v) - \{e\}} \dim(\mathcal{S}_{e'}^*).$$

Proof. The upper bound may be proved as follows. Since $\dim(\mathcal{C}_J) = \dim(\mathcal{C}) - \dim(\mathcal{C}|_{\overline{J}})$ for any $J \subseteq I$, we may write (6) as

$$\dim(C_v^*) = \sum_{e \in E(v)} \dim(\mathcal{C}|_{\overline{J}(e)}) - (|E(v)| - 1)\dim(\mathcal{C}).$$

Now, let $e \in E(v)$ be fixed. We have

$$\dim(C_v^*) = \dim(\mathcal{C}|_{\overline{J}(e)}) + \sum_{e' \in E(v) - \{e\}} \left(\dim(\mathcal{C}|_{\overline{J}(e')}) - \dim(\mathcal{C})\right).$$

However, as can be seen from Figure 4, $\overline{J}(e)$ is the disjoint union of $\omega^{-1}(v)$ and the sets $J(e')$, $e' \in E(v) - \{e\}$. Therefore,

$$\dim(\mathcal{C}|_{\overline{J}(e)}) \leq \dim(\mathcal{C}|_{\omega^{-1}(v)}) + \sum_{e' \in E(v) - \{e\}} \dim(\mathcal{C}|_{J(e')}),$$

and hence,

$$\dim(C_v^*) \leq \dim(\mathcal{C}|_{\omega^{-1}(v)}) + \sum_{e' \in E(v) - \{e\}} \left(\dim(\mathcal{C}|_{J(e')}) + \dim(\mathcal{C}|_{\overline{J}(e')}) - \dim(\mathcal{C})\right).$$

The lemma now follows from (5). □

To justify our claim that the upper bound of the previous lemma is an improvement upon Forney's bound $\dim(C_v^*) \leq \text{len}(C_v^*) - \dim(\mathcal{S}_e^*)$ [8, Theorem 5], we note that

$$\begin{aligned}\dim(\mathcal{C}|_{\omega^{-1}(v)}) + \sum_{e' \in E(v)-\{e\}} \dim(\mathcal{S}_{e'}^*) &= \dim(\mathcal{C}|_{\omega^{-1}(v)}) + \sum_{e' \in E(v)} \dim(\mathcal{S}_{e'}^*) \; - \; \dim(\mathcal{S}_e^*) \\ &\leq |\omega^{-1}(v)| + \sum_{e' \in E(v)} \text{len}(\mathcal{S}_{e'}^*) \; - \; \dim(\mathcal{S}_e^*) \\ &= \text{len}(C_v^*) - \dim(\mathcal{S}_e^*),\end{aligned}$$

the last equality being a consequence of the fact that C_v^* is, by definition, a subspace of $\mathbb{F}^{\omega^{-1}(v)} \oplus \left(\bigoplus_{e' \in E(v)} \mathcal{S}_{e'}^*\right)$, Thus, for any $\mathcal{C}$, ω and v such that $\dim(\mathcal{C}|_{\omega^{-1}(v)}) < |\omega^{-1}(v)|$, we see that the bound of Lemma 2.3 is strictly better than Forney's bound.

As already mentioned, among all tree realizations of $\mathcal{C}$ extending (T, ω), the minimal realization $\mathcal{M}(\mathcal{C}; T, \omega)$ minimizes state space dimension at each edge of the tree T. It is natural to ask whether $\mathcal{M}(\mathcal{C}; T, \omega)$ also minimizes local constraint code dimension at each vertex of T. We will show in the next section that $\mathcal{M}(\mathcal{C}; T, \omega)$ does in fact have the following property:

> if $(T, \omega, (\mathcal{S}_e, \; e \in E), (C_v, \; v \in V))$ is a tree realization of $\mathcal{C}$ that extends (T, ω), then for all $v \in V$, $\dim(C_v^*) \leq \dim(C_v)$.

We will deduce this fact from an alternative construction of $\mathcal{M}(\mathcal{C}; T, \omega)$ that we present next.

3. A Construction of $\mathcal{M}(\mathcal{C}; T, \omega)$ via State Merging

The construction we describe in this section takes an arbitrary tree realization Γ that extends the tree decomposition (T, ω) — for example, the trivial extension given in Example 2.1 — and via a sequence of transformations, converts Γ to $\mathcal{M}(\mathcal{C}; T, \omega)$. These transformations constitute a natural generalization of the state-merging process in the context of minimal trellis realizations; see, for example, [23] or [31, Section 4]. It would be useful to keep this special case in mind while going through the details of the description that follows. For the reader's convenience, we have also included (in Appendix C) a specific example meant to serve as an aid to understanding the description of state merging in the context of tree realizations.

Let $\Gamma = (T, \omega, (\mathcal{S}_e, \; e \in E), (C_v, \; v \in V))$ be an essential[3] tree realization of a code $\mathcal{C}$ with index set I, and let $\mathfrak{B}$ be the full behavior of Γ. As Γ is essential, we have that $\mathfrak{B}|_e = \mathcal{S}_e$ for all $e \in E$ (by definition), and $\mathfrak{B}|_v = C_v$ for all $v \in V$ (by Lemma 2.2).

Pick an arbitrary edge $\hat{e} \in E$, and for ease of notation, set $J = J(\hat{e})$ and $\overline{J} = \overline{J}(\hat{e})$. Let W be the subspace of $\mathcal{S}_{\hat{e}}$ defined by

$$W = \{\mathbf{s} \in \mathcal{S}_{\hat{e}} : \; \exists \, \mathbf{b} \in \mathfrak{B} \text{ such that } \mathbf{b}|_I \in \mathcal{C}_J \oplus \mathcal{C}_{\overline{J}} \text{ and } \mathbf{b}|_{\hat{e}} = \mathbf{s}\}. \tag{7}$$

We will define a new tree model $\overline{\Gamma} = (T, \omega, (\overline{\mathcal{S}}_e, \, e \in E), (\overline{C}_v, \, v \in V))$, such that states in the same coset of W in $\mathcal{S}_{\hat{e}}$ are represented by a single "merged" state in $\overline{\mathcal{S}}_{\hat{e}}$.

Let

$$\Phi : \mathbb{F}^I \oplus \left(\bigoplus_{e \neq \hat{e}} \mathcal{S}_e\right) \oplus \mathcal{S}_{\hat{e}} \longrightarrow \mathbb{F}^I \oplus \left(\bigoplus_{e \neq \hat{e}} \mathcal{S}_e\right) \oplus \mathcal{S}_{\hat{e}}/W$$

be the mapping defined by

$$\Phi((x_i, \, i \in I), (\mathbf{s}_e, \, e \neq \hat{e}), \mathbf{s}) \;=\; ((x_i, \, i \in I), (\mathbf{s}_e, \, e \neq \hat{e}), \mathbf{s} + W). \tag{8}$$

Define $\overline{\mathfrak{B}} = \Phi(\mathfrak{B})$. It is clear from the definitions that $\overline{\mathfrak{B}}|_I = \mathfrak{B}|_I = \mathcal{C}$, and that $\dim(\overline{\mathfrak{B}}) \leq \dim(\mathfrak{B})$.

[3]This restriction can be dropped by considering $\mathsf{ess}(\Gamma)$ instead; see Theorem 3.4.

Consider now the tree model $\overline{\Gamma} = (T, \omega, (\overline{\mathcal{S}}_e,\, e \in E), (\overline{C}_v,\, v \in V))$, where $\overline{\mathcal{S}}_e = \overline{\mathfrak{B}}|_e$ for each $e \in E$, and $\overline{C}_v = \overline{\mathfrak{B}}|_v$ for each $v \in V$. Note that $\overline{\mathcal{S}}_{\hat{e}} = \mathcal{S}_{\hat{e}}/W$, and for $e \neq \hat{e}$, we have $\overline{\mathcal{S}}_e = \overline{\mathfrak{B}}|_e = \mathfrak{B}|_e = \mathcal{S}_e$. All states in $\mathcal{S}_{\hat{e}}$ belonging to the same coset of W, say, $\mathbf{s} + W$, are mapped to (or merged into) the single state $\mathbf{s} + W$ in $\overline{\mathcal{S}}_{\hat{e}}$. Further note that if v is not one of the two vertices incident with $\hat{e}$, then $\overline{C}_v = \overline{\mathfrak{B}}|_v = \mathfrak{B}|_v = C_v$. At the two vertices that are incident with $\hat{e}$, the local constraints are appropriately modified to take into account the state-merging at edge $\hat{e}$. In any case, we have

$$\dim(\overline{\mathcal{S}}_e) = \dim(\overline{\mathfrak{B}}|_e) \leq \dim(\mathfrak{B}|_e) = \dim(\mathcal{S}_e), \quad \text{for each } e \in E, \tag{9}$$

and

$$\dim(\overline{C}_v) = \dim(\overline{\mathfrak{B}}|_v) \leq \dim(\mathfrak{B}|_v) = \dim(C_v), \quad \text{for each } v \in V. \tag{10}$$

We claim that $\overline{\Gamma}$ is an essential tree realization of $\mathcal{C}$. To prove this claim, we must show that $\mathfrak{B}(\overline{\Gamma})|_e = \overline{\mathfrak{B}}|_e$ for all $e \in E$, and that $\mathfrak{B}(\overline{\Gamma})|_I = \mathcal{C}$, where $\mathfrak{B}(\overline{\Gamma})$ denotes the full behavior of $\overline{\Gamma}$. Note that we do *not* claim that $\mathfrak{B}(\overline{\Gamma}) = \overline{\mathfrak{B}}$; indeed, this may not be true.

It is easy to see that the inclusion $\mathcal{C} \subseteq \mathfrak{B}(\overline{\Gamma})|_I$ holds. Indeed, since $\overline{\Gamma} = (T, \omega, (\overline{\mathfrak{B}}|_e,\, e \in E), (\overline{\mathfrak{B}}|_v,\, v \in V))$, it is evident that any $\overline{\mathbf{b}} \in \overline{\mathfrak{B}}$ satisfies all the local constraints of $\overline{\Gamma}$, and hence is in $\mathfrak{B}(\overline{\Gamma})$. Therefore, $\overline{\mathfrak{B}} \subseteq \mathfrak{B}(\overline{\Gamma})$, and in particular, $\mathcal{C} = \overline{\mathfrak{B}}|_I \subseteq \mathfrak{B}(\overline{\Gamma})|_I$.

The reverse inclusion, $\mathfrak{B}(\overline{\Gamma})|_I \subseteq \mathcal{C}$, follows from part (a) of the lemma below.

Lemma 3.1. *Let $\overline{\mathbf{b}}$ be a global configuration in $\mathfrak{B}(\overline{\Gamma})$. Then,*

(a) $\overline{\mathbf{b}}|_I \in \mathcal{C}$*; and*
(b) $\overline{\mathbf{b}}|_{\hat{e}} = \mathbf{0}$ *if and only if* $\overline{\mathbf{b}}|_I \in \mathcal{C}_J \oplus \mathcal{C}_{\overline{J}}$.

We defer the proof of the lemma to Appendix B. Lemma 3.1(a) shows that $\mathfrak{B}(\overline{\Gamma})|_I \subseteq \mathcal{C}$, thus proving that $\overline{\Gamma}$ is a tree realization of $\mathcal{C}$. It remains to show that Γ is essential, *i.e.*, that $\mathfrak{B}(\overline{\Gamma})|_e = \overline{\mathfrak{B}}|_e$ for all $e \in E$. This is shown by the following simple argument. We have already seen that $\overline{\mathfrak{B}} \subseteq \mathfrak{B}(\overline{\Gamma})$, and hence, $\overline{\mathfrak{B}}|_e \subseteq \mathfrak{B}(\overline{\Gamma})|_e$ for all $e \in E$. On the other hand, at any $e \in E$, $\mathfrak{B}(\overline{\Gamma})|_e$ is, by definition, a subspace of $\overline{\mathcal{S}}_e = \overline{\mathfrak{B}}|_e$. Hence, $\overline{\Gamma}$ is essential, thus proving our original claim, which we record in the following proposition.

Proposition 3.2. *The tree model $\overline{\Gamma}$ is an essential tree realization of $\mathcal{C}$.*

Let us call the process described above of obtaining $\overline{\Gamma}$ from Γ as the *state-merging process at edge $\hat{e}$*. We use the notation $\overline{\Gamma} = \mathsf{merge}_{\hat{e}}(\Gamma)$ to denote this transformation. Our goal now is to show that, starting from an essential tree realization, if we apply the state-merging process at each edge of the underlying tree, then we always end up with a minimal realization. A proof of this assertion requires the following technical lemma, whose proof we also defer to Appendix B.

Lemma 3.3. *Suppose that there exists $e' \in E - \{\hat{e}\}$ such that the full behavior, $\mathfrak{B}$, of Γ satisfies the following property: for $\mathbf{b} \in \mathfrak{B}$, we have $\mathbf{b}|_{e'} = \mathbf{0}$ if and only if $\mathbf{b}|_I \in \mathcal{C}_{J(e')} \oplus \mathcal{C}_{\overline{J}(e')}$. Then, for any $\overline{\mathbf{b}} \in \mathfrak{B}(\overline{\Gamma})$, we also have $\overline{\mathbf{b}}|_{e'} = \mathbf{0}$ if and only if $\overline{\mathbf{b}}|_I \in \mathcal{C}_{J(e')} \oplus \mathcal{C}_{\overline{J}(e')}$.*

We are now in a position to prove the main result of this section, which provides a construction of $\mathcal{M}(\mathcal{C}; T, \omega)$ via state merging. The reader is referred to Appendix C for an example that illustrates this construction.

Theorem 3.4. *Let Γ be a tree realization of $\mathcal{C}$ that extends the tree decomposition (T, ω). Let $e_1, e_2, \ldots, e_{|E|}$ be a listing of the edges of T. Set $\Gamma_0 = \mathsf{ess}(\Gamma)$, and for $i = 1, 2, \ldots, |E|$, set $\Gamma_i = \mathsf{merge}_{e_i}(\Gamma_{i-1})$. Then, $\Gamma_{|E|}$ is the minimal tree realization $\mathcal{M}(\mathcal{C}; T, \omega)$.*

Proof. Let $\mathfrak{B}$ denote the full behavior of Γ (and hence, also of $\mathsf{ess}(\Gamma)$), and for $i = 1, 2, \ldots, |E|$, let $\mathfrak{B}(\Gamma_i)$ denote the full behavior of Γ_i. By Proposition 3.2, each Γ_i is an essential tree realization of $\mathcal{C}$.

By Lemma 3.1(b), for any $\mathbf{b} \in \mathfrak{B}(\Gamma_i)$, we have $\mathbf{b}|_{e_i} = \mathbf{0}$ if and only if $\mathbf{b}|_I \in \mathcal{C}_{J(e_i)} \oplus \mathcal{C}_{\overline{J}(e_i)}$. Furthermore, by Lemma 3.3, for any $j \geq i$, $\mathfrak{B}(\Gamma_j)$ satisfies the following property:

for any $\mathbf{b} \in \mathfrak{B}(\Gamma_j)$, we have $\mathbf{b}|_{e_i} = \mathbf{0}$ if and only if $\mathbf{b}|_I \in \mathcal{C}_{J(e_i)} \oplus \mathcal{C}_{\overline{J}(e_i)}$.

In particular, $\mathfrak{B}^* \stackrel{\text{def}}{=} \mathfrak{B}(\Gamma_{|E|})$ satisfies the following property for $i = 1, 2, \ldots, |E|$:

for any $\mathbf{b} \in \mathfrak{B}^*$, we have $\mathbf{b}|_{e_i} = \mathbf{0}$ if and only if $\mathbf{b}|_I \in \mathcal{C}_{J(e_i)} \oplus \mathcal{C}_{\overline{J}(e_i)}$.

Let us call the above property (P). Property (P) has two important consequences. Firstly, it implies that if $\mathbf{b} \in \mathfrak{B}^*$ is such that $\mathbf{b}|_I = \mathbf{0}$, then $\mathbf{b}|_e = \mathbf{0}$ for all $e \in E$. This means that the projection $\pi : \mathfrak{B}^* \to \mathcal{C}$ defined by $\pi(\mathbf{b}) = \mathbf{b}|_I$ is in fact an isomorphism.

For the second consequence of (P), consider, for any $e \in E$, the homomorphism $\beta_e : \mathcal{C} \to \mathfrak{B}^*|_e$ defined by $\beta_e(\mathbf{c}) = (\pi^{-1}(\mathbf{c}))|_e$. This map is well-defined since π is an isomorphism. Property (P) is equivalent to the assertion that, for any $e \in E$, the kernel of β_e is precisely $\mathcal{C}_{J(e)} \oplus \mathcal{C}_{\overline{J}(e)}$. Therefore, $\mathfrak{B}^*|_e \cong \mathcal{C}/(\mathcal{C}_{J(e)} \oplus \mathcal{C}_{\overline{J}(e)})$.

Thus, for each $e \in E$, state space $\mathfrak{B}^*|_e$ is isomorphic to $\mathcal{S}_e^*$ defined in (1), and the map β_e is the canonical projection map $\mathbf{s}_e^*$ given by (2). It easily follows that for each $v \in V$, $\mathfrak{B}^*|_v$ is isomorphic to C_v^* defined in (3). Hence, $\Gamma_{|E|} = (T, \omega, (\mathfrak{B}^*|_e,\ e \in E), (\mathfrak{B}^*|_v,\ v \in V))$ is the minimal realization $\mathcal{M}(\mathcal{C}; T, \omega)$. □

Observe that at each step of the procedure outlined in Theorem 3.4, the dimensions of the state spaces and the local constraints do not increase. To make this precise, given tree models $\Gamma' = (T, \omega, (\mathcal{S}'_e,\ e \in E), (C'_v,\ v \in V))$ and $\Gamma'' = (T, \omega, (\mathcal{S}''_e,\ e \in E), (C''_v,\ v \in V))$, let us say that $\Gamma' \preccurlyeq \Gamma''$ if $\dim(\mathcal{S}'_e) \leq \dim(\mathcal{S}''_e)$ for all $e \in E$, and $\dim(C'_v) \leq \dim(C''_v)$ for all $v \in V$. Then, for Γ and Γ_i, $i = 0, 1, 2 \ldots, |E|$, as in the statement of Theorem 3.4, we have by virtue of (9) and (10),

$$\Gamma_{|E|} \preccurlyeq \Gamma_{|E|-1} \preccurlyeq \ldots \preccurlyeq \Gamma_1 \preccurlyeq \Gamma_0 = \mathsf{ess}(\Gamma) \preccurlyeq \Gamma.$$

Thus, we have that if Γ is any tree realization of $\mathcal{C}$ that extends the tree decomposition (T, ω), then $\mathcal{M}(\mathcal{C}; T, \omega) \preccurlyeq \Gamma$. We record this strong property of minimal realizations as a corollary to Theorem 3.4.

Corollary 3.5. *Let (T, ω) be a tree decomposition of the index set of a code $\mathcal{C}$, and let $\mathcal{M}(\mathcal{C}; T, \omega) = (T, \omega, (\mathcal{S}_e^*,\ e \in E), (C_v^*,\ v \in V))$ be the corresponding minimal tree realization of $\mathcal{C}$. Then, for any tree realization, $(T, \omega, (\mathcal{S}_e,\ e \in E), (C_v,\ v \in V))$, of $\mathcal{C}$ that extends (T, ω), we have $\dim(\mathcal{S}_e^*) \leq \dim(\mathcal{S}_e)$ for all $e \in E$, and $\dim(C_v^*) \leq \dim(C_v)$ for all $v \in V$.*

The procedure outlined in Theorem 3.4 does not translate to an efficient algorithm for the construction of $\mathcal{M}(\mathcal{C}; T, \omega)$. This is because the state-merging procedure that creates Γ_i from Γ_{i-1} requires knowledge of the full behavior of Γ_{i-1}, which may not be easily determined. So, as a practical method for constructing $\mathcal{M}(\mathcal{C}; T, \omega)$, given $\mathcal{C}$ and (T, ω), we propose a novel construction that relies upon the code decomposition techniques of the next section.

4. Code Decompositions

In previous work [18], it was demonstrated that techniques from the decomposition theory of matroids [28],[29] could be put to good use in a coding-theoretic setting. The decomposition theory in that work was presented in the context of binary linear codes. As we will now show, the basic elements of that theory can be easily extended to cover the case of nonbinary codes as well. The object of this exercise is not just to create a more general code decomposition theory, but as we will see in the next section, this decomposition theory ties in very nicely with the theory of tree realizations.

Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be linear codes over the finite field[4] $\mathbb{F}_q = GF(q)$, defined on the index sets I_1 and I_2, respectively. Let $I_1 \Delta I_2$ denote the symmetric difference, $(I_1 \cup I_2) - (I_1 \cap I_2)$, of the index sets. We will construct a code $\mathsf{S}(\mathcal{C}_1, \mathcal{C}_2)$ with $I_1 \Delta I_2$ as its index set. For $\mathbf{x} = (x_i,\, i \in I_1) \in \mathcal{C}_1$ and $\mathbf{y} = (y_i,\, i \in I_2) \in \mathcal{C}_2$, let $\mathbf{x} \star \mathbf{y} = (c_i,\, i \in I_1 \cup I_2)$ be defined by

$$c_i = \begin{cases} x_i & \text{for } i \in I_1 - I_2 \\ y_i & \text{for } i \in I_2 - I_1 \\ x_i - y_i & \text{for } i \in I_1 \cap I_2. \end{cases}$$

Setting $\mathcal{C}_1 \star \mathcal{C}_2 = \{\mathbf{x} \star \mathbf{y} :\ \mathbf{x} \in \mathcal{C}_1,\, \mathbf{y} \in \mathcal{C}_2\}$, we see that $\mathcal{C}_1 \star \mathcal{C}_2$ has $I_1 \cup I_2$ as its index set. We take $\mathsf{S}(\mathcal{C}_1, \mathcal{C}_2)$ to be the cross-section $(\mathcal{C}_1 \star \mathcal{C}_2)_{I_1 \Delta I_2}$. Note that when $I_1 \cap I_2 = \emptyset$, we have $\mathsf{S}(\mathcal{C}_1, \mathcal{C}_2) = \mathcal{C}_1 \star \mathcal{C}_2 = \mathcal{C}_1 \oplus \mathcal{C}_2$.

For $i = 1, 2$, let $\mathcal{C}_i^{(p)}$ and $\mathcal{C}_i^{(s)}$ denote the projection $\mathcal{C}_i|_{I_1 \cap I_2}$ and the cross-section $(\mathcal{C}_i)_{I_1 \cap I_2}$, respectively. The codes $\mathcal{C}_i^{(p)}$ and $\mathcal{C}_i^{(s)}$, for $i = 1, 2$, all have $I_1 \cap I_2$ as their index set. The dimension of $\mathsf{S}(\mathcal{C}_1, \mathcal{C}_2)$ can be expressed in terms of the codes $\mathcal{C}_i$, $\mathcal{C}_i^{(p)}$ and $\mathcal{C}_i^{(s)}$, $i = 1, 2$, as stated in the following lemma.

Proposition 4.1. *For codes $\mathcal{C}_1, \mathcal{C}_2$, we have*

$$\dim(\mathsf{S}(\mathcal{C}_1, \mathcal{C}_2)) = \dim(\mathcal{C}_1) + \dim(\mathcal{C}_2) - \dim(\mathcal{C}_1^{(s)} \cap \mathcal{C}_2^{(s)}) - \dim(\mathcal{C}_1^{(p)} + \mathcal{C}_2^{(p)}),$$

where $\mathcal{C}_1^{(p)} + \mathcal{C}_2^{(p)} = \{\mathbf{x} + \mathbf{y} :\ \mathbf{x} \in \mathcal{C}_1^{(p)}, \mathbf{y} \in \mathcal{C}_2^{(p)}\}$.

Proof. For a code $\mathcal{C}$, and a subset J of its index set, the kernel of the projection map $\pi : \mathcal{C} \to \mathcal{C}|_{\overline{J}}$ is isomorphic to $\mathcal{C}_J$, and hence, $\dim(\mathcal{C}_J) = \dim(\mathcal{C}) - \dim(\mathcal{C}|_{\overline{J}})$. Thus, taking $\mathcal{C} = \mathcal{C}_1 \star \mathcal{C}_2$, and $J = I_1 \Delta I_2$, we find that

$$\dim(\mathsf{S}(\mathcal{C}_1, \mathcal{C}_2)) = \dim(\mathcal{C}_1 \star \mathcal{C}_2) - \dim((\mathcal{C}_1 \star \mathcal{C}_2)|_{I_1 \cap I_2}) = \dim(\mathcal{C}_1 \star \mathcal{C}_2) - \dim(\mathcal{C}_1^{(p)} + \mathcal{C}_2^{(p)}),$$

since $(\mathcal{C}_1 \star \mathcal{C}_2)|_{I_1 \cap I_2} = \mathcal{C}_1^{(p)} + \mathcal{C}_2^{(p)}$. So, we must show that $\dim(\mathcal{C}_1 \star \mathcal{C}_2) = \dim(\mathcal{C}_1) + \dim(\mathcal{C}_2) - \dim(\mathcal{C}_1^{(s)} \cap \mathcal{C}_2^{(s)})$.

Let $\widetilde{\mathcal{C}}_2$ be a copy of $\mathcal{C}_2$ defined on an index set that is disjoint from I_1. For each $\mathbf{y} \in \mathcal{C}_2$, denote by $\widetilde{\mathbf{y}}$ its copy in $\widetilde{\mathcal{C}}_2$. Consider the homomorphism $\phi : \mathcal{C}_1 \oplus \widetilde{\mathcal{C}}_2 \to \mathcal{C}_1 \star \mathcal{C}_2$ defined by $\phi(\mathbf{x}, \widetilde{\mathbf{y}}) = \mathbf{x} \star \mathbf{y}$. Note that $\mathbf{x} \star \mathbf{y} = \mathbf{0}$ iff $\mathbf{x}|_{I_1 - I_2} = \mathbf{y}|_{I_2 - I_1} = \mathbf{0}$ and $\mathbf{x}|_{I_1 \cap I_2} - \mathbf{y}|_{I_1 \cap I_2} = \mathbf{0}$. Equivalently, $\mathbf{x} \star \mathbf{y} = \mathbf{0}$ iff $\mathbf{x}|_{I_1 \cap I_2} \in \mathcal{C}_1^{(s)}$, $\mathbf{y}|_{I_1 \cap I_2} \in \mathcal{C}_2^{(s)}$, and $\mathbf{x}|_{I_1 \cap I_2} = \mathbf{y}|_{I_1 \cap I_2}$. It follows that the kernel of ϕ is isomorphic to

$$\{\mathbf{z} :\ \mathbf{z} \in \mathcal{C}_1^{(s)},\, \mathbf{z} \in \mathcal{C}_2^{(s)}\}.$$

which is simply $\mathcal{C}_1^{(s)} \cap \mathcal{C}_2^{(s)}$.

Hence, $\dim(\mathcal{C}_1 \star \mathcal{C}_2) = \dim(\mathcal{C}_1 \oplus \widetilde{\mathcal{C}}_2) - \dim(\ker(\phi)) = \dim(\mathcal{C}_1) + \dim(\mathcal{C}_2) - \dim(\mathcal{C}_1^{(s)} \cap \mathcal{C}_2^{(s)})$, as desired. □

We will restrict our attention to a particular instance of the $\mathsf{S}(\mathcal{C}_1, \mathcal{C}_2)$ construction, in which we require that the codes $\mathcal{C}_i^{(p)}$ and $\mathcal{C}_i^{(s)}$, $i = 1, 2$, take on a specific form. We need to introduce some notation first. For each positive integer r, set $m_r = (q^r - 1)/(q - 1)$, and fix an $r \times m_r$ matrix, which we denote by D_r, over $\mathbb{F}_q$, with the property that each pair of columns of D_r is linearly independent over $\mathbb{F}_q$. Note that D_r is a parity-check matrix for an $[m_r, m_r - r]$ Hamming code over $\mathbb{F}_q$ (cf. [30, § 3.3]). Let Δ_r denote the dual of this Hamming code, *i.e.*, Δ_r is the $[m_r, r]$ code over $\mathbb{F}_q$ *generated* by D_r. The code Δ_r is sometimes referred to as a *simplex code*.

Given an $r > 0$, suppose that the codes $\mathcal{C}_1$ and $\mathcal{C}_2$, defined on the index sets I_1 and I_2, respectively, are such that $|I_1 \cap I_2| = m_r$, and for $i = 1, 2$, we have $\mathcal{C}_i^{(p)} = \Delta_r$ and $\mathcal{C}_i^{(s)} = \{\mathbf{0}\}$. In such

[4]Up to this point, we did not need to specify the number of elements in the finite field over which we were working, but from now on, it will be useful for us to do so.

a case, $\mathsf{S}(\mathcal{C}_1, \mathcal{C}_2)$ is called the *r-sum* of $\mathcal{C}_1$ and $\mathcal{C}_2$, and is denoted by $\mathcal{C}_1 \oplus_r \mathcal{C}_2$. It is convenient to extend this definition to the case of $r = 0$ as well: when $|I_1 \cap I_2| = 0$, the 0-sum $\mathcal{C}_1 \oplus_0 \mathcal{C}_2$ is defined to be the direct sum $\mathcal{C}_1 \oplus \mathcal{C}_2$.

Example 4.1. *Consider the case of codes defined over the binary field $\mathbb{F}_2$. Note that $\Delta_1 = \{0, 1\}$. Suppose that $|I_1 \cap I_2| = 1$, and that the coordinates of $\mathcal{C}_1$ are $\mathcal{C}_2$ are ordered so that the index common to I_1 and I_2 corresponds to the last coordinate of $\mathcal{C}_1$ and the first coordinate of $\mathcal{C}_2$. The conditions necessary for the 1-sum $\mathcal{C}_1 \oplus_1 \mathcal{C}_2$ to be defined can then be stated as*

(P1) $0 \ldots 01$ *is not a codeword of $\mathcal{C}_1$, and the last coordinate of $\mathcal{C}_1$ is not identically zero;*

(P2) $10 \ldots 0$ *is not a codeword of $\mathcal{C}_2$, and the first coordinate of $\mathcal{C}_2$ is not identically zero.*

The composite code $\mathsf{S}(\mathcal{C}_1, \mathcal{C}_2)$ resulting from $\mathcal{C}_1, \mathcal{C}_2$ that satisfy (P1), (P2) above was studied in [18]*, where it was actually called a "2-sum".*

We would also like to point out that the specialization of our r-sum operation to the case $r = 2$ was called "$\overline{3}$-sum" in [18].[5] *To add to the confusion, there was in fact an operation called "3-sum" defined in* [18]*, but that, in a certain sense, dualizes the 2-sum operation we have given in this paper.*

For $r > 0$, note that if $\mathcal{C}_i^{(p)}$ and $\mathcal{C}_i^{(s)}$ ($i = 1, 2$) are in the form needed to define an r-sum, then $\mathcal{C}_1^{(p)} + \mathcal{C}_2^{(p)} = \Delta_r$, and $\mathcal{C}_1^{(s)} \cap \mathcal{C}_2^{(s)} = \{\mathbf{0}\}$. Therefore, as a corollary to Proposition 4.1, we have the following result (which also applies trivially to the $r = 0$ case).

Corollary 4.2. *For $r \geq 0$, if $\mathcal{C}_1, \mathcal{C}_2$ are such that $\mathcal{C}_1 \oplus_r \mathcal{C}_2$ can be defined, then*

$$\dim(\mathcal{C}_1 \oplus_r \mathcal{C}_2) = \dim(\mathcal{C}_1) + \dim(\mathcal{C}_2) - r.$$

An elementary property of direct sums (*i.e.*, 0-sums) is that a code $\mathcal{C}$ is expressible as a direct sum of smaller codes if and only if there exists a partition $(J, \overline{J})$ of the index set of $\mathcal{C}$ such that $\dim(\mathcal{C}|_J) + \dim(\mathcal{C}|_{\overline{J}}) - \dim(\mathcal{C}) = 0$. This property extends beautifully to r-sums in general.

Theorem 4.3. *Let $\mathcal{C}$ be a linear code over $\mathbb{F}_q$, defined on the index set I, and let r be a positive integer. Then, the following statements are equivalent.*

(a) *$\mathcal{C} = \mathcal{C}_1 \oplus_r \mathcal{C}_2$ for some codes $\mathcal{C}_1, \mathcal{C}_2$.*

(b) *There exists a partition $(J, \overline{J})$ of I, with $\min\{|J|, |\overline{J}|\} \geq r$, such that*

$$\dim(\mathcal{C}|_J) + \dim(\mathcal{C}|_{\overline{J}}) - \dim(\mathcal{C}) = r.$$

Proof. (a) $\Rightarrow$ (b): See Appendix D.

(b) $\Rightarrow$ (a): We give here a complete proof of this direction of the theorem, as it gives an explicit construction of codes $\mathcal{C}_1$, $\mathcal{C}_2$ such that $\mathcal{C} = \mathcal{C}_1 \oplus_r \mathcal{C}_2$, given a partition $(J, \overline{J})$ as in (b). The proof generalizes ideas from similar constructions presented in [18].

Let $(J, \overline{J})$ be a partition of I such that $\dim(\mathcal{C}|_J) + \dim(\mathcal{C}|_{\overline{J}}) - \dim(\mathcal{C}) = r$. Set $n = |I|$ and $k = \dim(\mathcal{C})$, and let G be a $k \times n$ generator matrix for $\mathcal{C}$. Without loss of generality, we may assume that the columns of G are ordered so that the first $|J|$ columns are indexed by the elements of J, and the rest by the elements of $\overline{J}$. In the following exposition, we will often permute the columns of G to bring the matrix into some desired form. Whenever this is the case, it will be tacitly assumed that column indices migrate with the columns.

Let $G|_J$ and $G|_{\overline{J}}$ denote the restrictions of G to the columns indexed by the elements of J and $\overline{J}$, respectively; thus, $G = \begin{bmatrix} G|_J & G|_{\overline{J}} \end{bmatrix}$. Let $\mathsf{rank}(G|_J) = k_1$ and $\mathsf{rank}(G|_{\overline{J}}) = k_2$; by our assumption on $(J, \overline{J})$, we have we have $k_1 + k_2 = k + r$.

[5] The 2-sum and $\overline{3}$-sum operations defined in [18] imposed additional conditions on the lengths of the codes involved in the sum, which we have dropped here.

Bring G into reduced row-echelon form (rref) over $\mathbb{F}_q$. Permuting within the columns of $G|_J$ and within those of $G|_{\overline{J}}$ if necessary, $\mathsf{rref}(G)$ may be assumed to be of the form

$$\overline{G} = \begin{bmatrix} I_{k_1} & A & \mathbf{O} & B \\ \mathbf{O} & \mathbf{O} & I_{k-k_1} & C \end{bmatrix}, \tag{11}$$

where I_j, for $j = k_1, k_2 - r$, denotes the $j \times j$ identity matrix, A is a $k_1 \times (|J| - k_1)$ matrix, B is a $k_1 \times (|\overline{J}| - k + k_1)$ matrix, C is a $(k - k_1) \times (|\overline{J}| - k + k_1)$ matrix, and the $\mathbf{O}$'s denote all-zeros matrices of appropriate sizes.

The fact that the submatrix $\begin{bmatrix} \mathbf{O} & B \\ I_{k-k_1} & C \end{bmatrix}$ must have rank equal to $\mathsf{rank}(G|_{\overline{J}}) = k_2$ implies that B must have rank $k_2 - (k - k_1) = r$. Hence, B has r linearly independent rows, call them $\mathbf{b}_1, \ldots, \mathbf{b}_r$, which form a basis of the row-space of B. Permuting the first k_1 rows of $\overline{G}$ if necessary, we may assume that $\mathbf{b}_1, \ldots, \mathbf{b}_r$ constitute the first r rows of B. (Permuting these rows of $\overline{G}$ will also permute the rows of the I_{k_1} matrix, but the effects of this can be negated by appropriately permuting the first k_1 columns of $\overline{G}$.) Any row of B is uniquely expressible as a linear combination (over $\mathbb{F}_q$) of $\mathbf{b}_1, \ldots, \mathbf{b}_r$. In particular, for $i = 1, 2, \ldots, k_1$, the ith row of B can be uniquely expressed as $\sum_{j=1}^{r} \alpha_{i,j}\mathbf{b}_j$ for some $\alpha_{i,j} \in \mathbb{F}_q$.

Let us denote by $\mathbf{d}_1, \ldots, \mathbf{d}_r$, the rows of the $r \times m_r$ generator matrix, D_r, of the code Δ_r. Let X be the $k_1 \times m_r$ matrix such that for $i = 1, 2, \ldots, k_1$, the ith row of X equals $\sum_{j=1}^{r} \alpha_{i,j}\mathbf{d}_j$, where the $\alpha_{i,j}$'s are such that the ith row of B is $\sum_{j=1}^{r} \alpha_{i,j}\mathbf{b}_j$. Thus, the row-space of X is the span of $\mathbf{d}_1, \ldots, \mathbf{d}_r$, *i.e.*, it is the code Δ_r. To the columns of X, we assign indices from some set I_X disjoint from I.

Now, define the $k_1 \times (|J| + m_r)$ matrix

$$G_1 = \begin{bmatrix} I_{k_1} & A & X \end{bmatrix}, \tag{12}$$

allowing the submatrix $[I_{k_1} \;\; A]$ to retain its column indices from $\overline{G}$. Also, define the $k \times (|\overline{J}| + m_r)$ matrix

$$G_2 = \begin{bmatrix} X & \mathbf{O} & B \\ \mathbf{O} & I_{k-k_1} & C \end{bmatrix}, \tag{13}$$

again allowing the submatrix $\begin{bmatrix} \mathbf{O} & B \\ I_{k-k_1} & C \end{bmatrix}$ to retain its column indices from $\overline{G}$. Thus, the index set of the columns of G_1 is $I_1 \stackrel{\text{def}}{=} J \,\dot\cup\, I_X$, while that of the columns of G_2 is $I_2 \stackrel{\text{def}}{=} I_X \,\dot\cup\, \overline{J}$.

Finally, for $i = 1, 2$, let $\mathcal{C}_i$ denote the code over $\mathbb{F}_q$ generated by G_i. The following facts about $\mathcal{C}_1$ and $\mathcal{C}_2$ may be verified:

(i) $\dim(\mathcal{C}_i) = \mathsf{rank}(G_i) = k_i$, $i = 1, 2$.

(ii) $\mathcal{C}_1 \oplus_r \mathcal{C}_2$ can be defined, so that by Corollary 4.2, $\dim(\mathcal{C}_1 \oplus_r \mathcal{C}_2) = k_1 + k_2 - r = k = \dim(\mathcal{C})$.

(iii) All rows of $\overline{G}$ are in $\mathcal{C}_1 \oplus_r \mathcal{C}_2$. Since $\overline{G}$ generates the same code as G (recall that column indices get permuted along with columns), we see that $\mathcal{C}_1 \oplus_r \mathcal{C}_2$ contains all the codewords of $\mathcal{C}$.

We leave the details of the routine verification of the above facts to the reader. It only remains to point out that facts (ii) and (iii) above show that $\mathcal{C}_1 \oplus_r \mathcal{C}_2 = \mathcal{C}$, thus completing the proof of the implication (b) $\Rightarrow$ (a). $\square$

The procedure described in the above proof can be formalized into an algorithm that takes as input a $k \times n$ generator matrix G (over $\mathbb{F}_q$) for $\mathcal{C}$, and a partition $(J, \overline{J})$ of the index set of $\mathcal{C}$, and produces as output generator matrices of two codes $\mathcal{C}_1$ and $\mathcal{C}_2$ (and their associated index sets) such that $\mathcal{C} = \mathcal{C}_1 \oplus_r \mathcal{C}_2$, where $r = \dim(\mathcal{C}|_J) + \dim(\mathcal{C}|_{\overline{J}}) - \dim(\mathcal{C})$. The run-time complexity of this procedure is determined by the following:

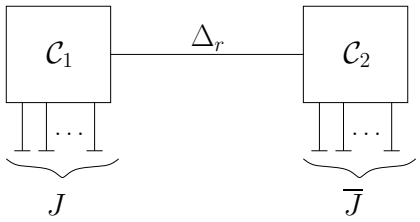

FIGURE 5. A tree realization of an r-sum decomposition.

- an rref computation to find $\overline{G}$ as in (11); this can be carried out in $O(k^2n)$ time, which is the run-time complexity of bringing a $k \times n$ matrix to reduced row-echelon form via elementary row operations;
- the computations required to identify a basis $(\mathbf{b}_1, \ldots, \mathbf{b}_r)$ of the row-space of the matrix B, and correspondingly the coefficients $\alpha_{i,j}$; this could be done by computing the rref of B, which would also take $O(k^2n)$ time;
- the computations needed to determine the $k_1 \times m_r$ matrix X; each row of the matrix requires $O(rm_r)$ computations, and there are $k_1 = O(|J|)$ rows, so the computation of X takes $O(|J|rm_r) = O(|J|rq^r)$ time.

Therefore, the entire procedure can be carried out in $O(k^2n + |J|rq^r)$ time. It is worth noting that the run-time complexity of the procedure is polynomial in n, k and q, but exponential in r.

We take a moment here to clarify an important point in our definition of the r-sum operation in this section. Recall that in order to define $\mathcal{C}_1 \oplus_r \mathcal{C}_2$, we required that $\mathcal{C}_1^{(p)} = \mathcal{C}_2^{(p)} = \Delta_r$. The code Δ_r acted as a "glue" in the r-sum composition of $\mathcal{C}$ from $\mathcal{C}_1$ and $\mathcal{C}_2$. An astute reader may have noted that if, instead of Δ_r, *any* fixed r-dimensional code over $\mathbb{F}_q$ were to be used as the "glue", the results stated in this section (Corollary 4.2 and Theorem 4.3) would hold continue to hold true. (The construction of the matrix X in the proof given for Theorem 4.3(a) would have to be appropriately modified, but the rest of the proof would go through unchanged.) So, the question naturally arises as to why we picked the code Δ_r for our definition. One reason for our choice was simply to make a specific r-sum definition compatible with the definitions in [18] (cf. Example 4.1). A second, and much more compelling, reason was the following: it is the choice of the code Δ_r as the "glue" that allows for the new recursive construction of minimal tree realizations described next.

5. A CONSTRUCTION OF $\mathcal{M}(\mathcal{C}; T, \omega)$ VIA CODE DECOMPOSITIONS

The procedure given in the previous section for determining an r-rum decomposition of a given code forms the basis of a new construction of minimal tree realizations that we present here. The key observation behind this construction is that if a code $\mathcal{C}$ has a partition $(J, \overline{J})$ of its index set such that $\dim(\mathcal{C}|_J) + \dim(\mathcal{C}|_{\overline{J}}) - \dim(\mathcal{C}) = r$, then $\mathcal{C}$ has an essential tree realization of the form depicted in Figure 5. The tree in the figure consists of a single edge $e = \{v_1, v_2\}$, the state space $\mathcal{S}_e$ is the code Δ_r, and the local constraint codes at the two vertices are the codes $\mathcal{C}_1$ and $\mathcal{C}_2$ such that $\mathcal{C}_1 \oplus_r \mathcal{C}_2 = \mathcal{C}$. In fact, this is the minimal realization $\mathcal{M}(\mathcal{C}; T, \omega)$, for the tree T consisting of the single edge $e = \{v_1, v_2\}$, and the index map ω such that $\omega^{-1}(v_1) = J$ and $\omega^{-1}(v_2) = \overline{J}$. This is simply because $\dim(\mathcal{S}_e) = \dim(\Delta_r) = r$, so by virtue of (5), $\mathcal{S}_e$ has the same dimension as the state space $\mathcal{S}_e^*$ in the minimal realization $\mathcal{M}(\mathcal{C}; T, \omega)$. So, by the uniqueness of minimal tree realizations, the tree realization depicted in Figure 5 is $\mathcal{M}(\mathcal{C}; T, \omega)$.

To summarize, if $\mathcal{C}$ is a code defined on the index set I, and (T, ω) is a tree decomposition of I such that T consists of the single edge $e = \{v_1, v_2\}$, then we may construct $\mathcal{M}(\mathcal{C}; T, \omega)$ as follows. Set $J = \omega^{-1}(v_1)$ and $\overline{J} = \omega^{-1}(v_2)$, and compute $r = \dim(\mathcal{C}|_J) + \dim(\mathcal{C}|_{\overline{J}}) - \dim(\mathcal{C})$. Assign an index set I_Δ that is disjoint from I to the code Δ_r. Use the procedure in the proof of Theorem 4.3

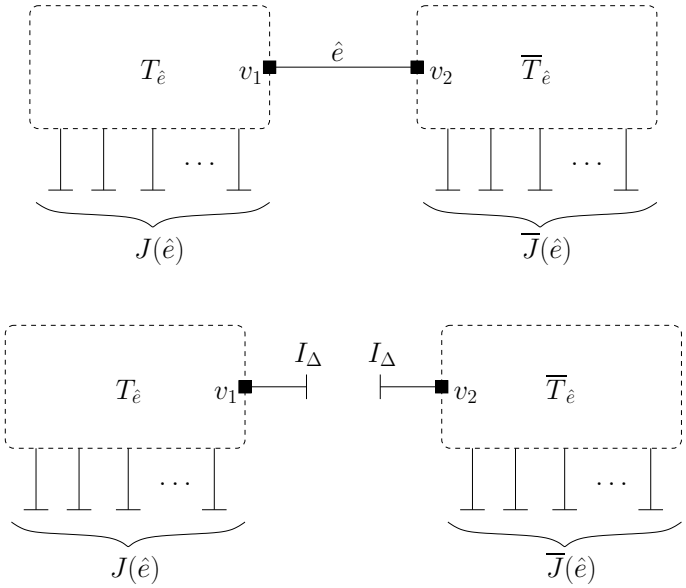

FIGURE 6. Depiction of the manner in which the tree decompositions $(T_{\hat{e}}, \omega_1)$ and $(\overline{T}_{\hat{e}}, \omega_2)$ are obtained from (T, ω).

to determine codes $\mathcal{C}_1$ and $\mathcal{C}_2$, defined on the respective index sets $I_1 = J \dot{\cup} I_\Delta$ and $I_2 = \overline{J} \dot{\cup} I_\Delta$, such that $\mathcal{C} = \mathcal{C}_1 \oplus_r \mathcal{C}_2$. For $i = 1, 2,$ assign $\mathcal{C}_i$ to be the local constraint code at vertex v_i, and assign Δ_r to be the state space at edge e. The resulting tree model $(T, \omega, \Delta_r, C_1, C_2)$ is the minimal tree realization $\mathcal{M}(\mathcal{C}; T, \omega)$.

Before describing how the construction may be extended to the case of trees with more than one edge, we deal with the trivial case of trees without any edges. If T is a tree consisting of a single vertex v, and no edges, then given any code $\mathcal{C}$ defined on some index set I, there is only one way of realizing $\mathcal{C}$ on T. This is the realization (T, ω, C_v), where ω is the unique mapping $\omega : I \to \{v\}$, and C_v is the code $\mathcal{C}$ itself. Of course, this is also the minimal realization $\mathcal{M}(\mathcal{C}; T, \omega)$.

At this point, we know how to construct $\mathcal{M}(\mathcal{C}; T, \omega)$, for any code $\mathcal{C}$, and any tree decomposition (T, ω) such that T has at most one edge. From this, we can recursively construct $\mathcal{M}(\mathcal{C}; T, \omega)$ for any $\mathcal{C}$ and any (T, ω), as we now describe.

Suppose that we know how to construct $\mathcal{M}(\mathcal{C}; T, \omega)$ for any $\mathcal{C}$, and any (T, ω) such that T has at most $\eta - 1$ edges, for some integer $\eta \geq 2$. Let $\mathcal{C}$ be a code defined on the index set I, and let (T, ω) be a tree decomposition such that $|E(T)| = \eta$. Pick any $\hat{e} = \{v_1, v_2\} \in E(T)$, and as usual, let $T_{\hat{e}}$ and $\overline{T}_{\hat{e}}$ be the two components of $T - \hat{e}$. We will assume that $v_1 \in V(T_{\hat{e}})$ and $v_2 \in V(\overline{T}_{\hat{e}})$. Let $J(\hat{e}) = \omega^{-1}(V(T_{\hat{e}}))$ and $\overline{J}(\hat{e}) = \omega^{-1}(V(\overline{T}_{\hat{e}}))$. Compute

$$r = \dim(\mathcal{C}|_{J(\hat{e})}) + \dim(\mathcal{C}|_{\overline{J}(\hat{e})}) - \dim(\mathcal{C}), \tag{14}$$

which determines the code Δ_r. Assign Δ_r an index set I_Δ that is disjoint from I. Use the procedure in the proof of Theorem 4.3 to determine codes $\mathcal{C}_1$ and $\mathcal{C}_2$, defined on the respective index sets $I_1 = J(\hat{e}) \dot{\cup} I_\Delta$ and $I_2 = \overline{J}(\hat{e}) \dot{\cup} I_\Delta$, such that $\mathcal{C} = \mathcal{C}_1 \oplus_r \mathcal{C}_2$.

Now, define the index maps $\omega_1 : I_1 \to V(T_{\hat{e}})$ and $\omega_2 : I_2 \to V(\overline{T}_{\hat{e}})$ as follows (see Figure 6):

$$\omega_1(i) = \begin{cases} \omega(i), & \text{if } i \in J(\hat{e}) \\ v_1, & \text{if } i \in I_\Delta \end{cases} \tag{15}$$

$$\omega_2(i) = \begin{cases} \omega(i), & \text{if } i \in \overline{J}(\hat{e}) \\ v_2, & \text{if } i \in I_\Delta \end{cases} \tag{16}$$

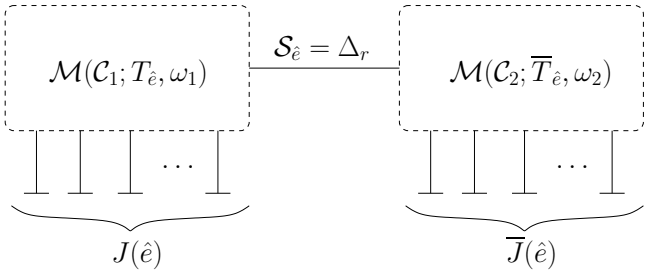

FIGURE 7. A depiction of the construction of Γ^* from $\mathcal{M}(\mathcal{C}_1; T_{\hat{e}}; \omega_1)$ and $\mathcal{M}(\mathcal{C}_2; \overline{T}_{\hat{e}}, \omega_2)$.

Thus, $(T_{\hat{e}}, \omega_1)$ and $(\overline{T}_{\hat{e}}, \omega_2)$ are tree decompositions of the index sets of $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively. As neither $E(T_{\hat{e}})$ nor $E(\overline{T}_{\hat{e}})$ contains the edge $\hat{e}$, we have $|E(T_{\hat{e}})| \leq \eta - 1$ and $|E(\overline{T}_{\hat{e}})| \leq \eta - 1$. Therefore, by our assumption, we know how to construct $\mathcal{M}(\mathcal{C}_1; T_{\hat{e}}, \omega_1)$ and $\mathcal{M}(\mathcal{C}_2; \overline{T}_{\hat{e}}, \omega_2)$. Let

$$\mathcal{M}(\mathcal{C}_1; T_{\hat{e}}, \omega_1) = \left(T_{\hat{e}}, \omega_1, (\mathcal{S}_e^{(1)}, e \in E(T_{\hat{e}})), (C_v^{(1)}, v \in V(T_{\hat{e}}))\right), \tag{17}$$

$$\mathcal{M}(\mathcal{C}_2; \overline{T}_{\hat{e}}, \omega_2) = \left(\overline{T}_{\hat{e}}, \omega_2, (\mathcal{S}_e^{(2)}, e \in E(\overline{T}_{\hat{e}})), (C_v^{(2)}, v \in V(\overline{T}_{\hat{e}}))\right). \tag{18}$$

Finally, set $\Gamma^* = (T, \omega, (\mathcal{S}_e, e \in E(T)), (C_v, v \in V(T)))$, where

$$\mathcal{S}_e = \begin{cases} \mathcal{S}_e^{(1)}, & \text{if } e \in E(T_{\hat{e}}) \\ \Delta_r, & \text{if } e = \hat{e} \\ \mathcal{S}_e^{(2)}, & \text{if } e \in E(\overline{T}_{\hat{e}}), \end{cases} \tag{19}$$

and

$$C_v = \begin{cases} C_v^{(1)}, & \text{if } v \in V(T_{\hat{e}}) \\ C_v^{(2)}, & \text{if } v \in V(\overline{T}_{\hat{e}}). \end{cases} \tag{20}$$

Figure 7 contains a depiction of Γ^*. It is easy to see that Γ^* is a tree realization of $\mathcal{C}$. Indeed, $\mathcal{M}(\mathcal{C}_1; T_{\hat{e}}, \omega_1)$ is a realization of $\mathcal{C}_1$, and $\mathcal{M}(\mathcal{C}_2; \overline{T}_{\hat{e}}, \omega_2)$ is a realization of $\mathcal{C}_2$, and hence (as should be clear from Figure 7), Γ^* is a realization of $\mathcal{C}_1 \oplus_r \mathcal{C}_2 = \mathcal{C}$. It is not immediately obvious that Γ^* is actually $\mathcal{M}(\mathcal{C}; T, \omega)$, but this is in fact true, as stated in the following proposition, a proof of which is given in Appendix E. It should be pointed out that the proof is strongly reliant on the choice of the code Δ_r as the "glue" in the r-sum decomposition of $\mathcal{C}$ into $\mathcal{C}_1$ and $\mathcal{C}_2$.

Proposition 5.1. *Γ^* is the minimal tree realization $\mathcal{M}(\mathcal{C}; T, \omega)$.*

In summary, we have the following recursive procedure for constructing $\mathcal{M}(\mathcal{C}; T, \omega)$, given a code $\mathcal{C}$ and a tree decomposition (T, ω).

Procedure `MIN_REALZN`$(\mathcal{C}, T, \omega)$

Input: A $k \times n$ generator matrix for a code $\mathcal{C}$, and a tree decomposition (T, ω) of the index set of $\mathcal{C}$.

Output: A specification of the state spaces and the local constraints in the minimal realization $\mathcal{M}(\mathcal{C}; T, \omega)$.

Step M1. If T consists of a single vertex, then return $\mathcal{M}(\mathcal{C}; T, \omega) = (T, \omega, \mathcal{C})$.

Step M2. If T contains at least one edge, then choose an $\hat{e} \in E(T)$. Let v_1 be the vertex of $T_{\hat{e}}$ incident with $\hat{e}$, and let v_2 be the vertex of $\overline{T}_{\hat{e}}$ incident with $\hat{e}$.
(M2.1) Compute $r = \dim(\mathcal{C}|_{J(\hat{e})}) + \dim(\mathcal{C}|_{\overline{J}(\hat{e})}) - \dim(\mathcal{C})$.
(M2.2) Determine Δ_r, and assign it an index set I_Δ disjoint from $J(\hat{e}) \cup \overline{J}(\hat{e})$.
(M2.3) Determine codes $\mathcal{C}_1$ and $\mathcal{C}_2$, with index sets $I_1 = J(\hat{e}) \dot{\cup} I_\Delta$ and $I_2 =$

$\overline{J}(\hat{e}) \dot{\cup} I_\Delta$, respectively, such that $\mathcal{C} = \mathcal{C}_1 \oplus_r \mathcal{C}_2$.
(M2.4) Determine the index maps ω_1 and ω_2 as in (15) and (16).

Step M3. Determine $\mathcal{M}(\mathcal{C}_1; T_{\hat{e}}, \omega_1)$ by calling `MIN_REALZN`$(\mathcal{C}_1, T_{\hat{e}}, \omega_1)$; determine $\mathcal{M}(\mathcal{C}_2; \overline{T}_{\hat{e}}, \omega_2)$ by calling `MIN_REALZN`$(\mathcal{C}_2, \overline{T}_{\hat{e}}, \omega_2)$. We may assume that $\mathcal{M}(\mathcal{C}_1; T_{\hat{e}}, \omega_1)$ and $\mathcal{M}(\mathcal{C}_2; \overline{T}_{\hat{e}}, \omega_2)$ are in the form given in (17) and (18).

Step M4. Return $\mathcal{M}(\mathcal{C}; T, \omega) = (T, \omega, (\mathcal{S}_e,\, e \in E(T)),\, (C_v,\, v \in V(T)))$, where $\mathcal{S}_e$ and C_v are as defined in (19) and (20).

A simplified version of the above procedure may be obtained by choosing, in Step M2, the edge $\hat{e}$ to be an edge incident with a leaf of T. Then, one of the two components of $T - \hat{e}$, say, $\overline{T}_{\hat{e}}$, consists of a single vertex, so that the call to `MIN_REALZN`$(\mathcal{C}_2; \overline{T}_{\hat{e}}, \omega_2)$ may be avoided, as it would simply return $(\overline{T}_{\hat{e}}, \omega_2, \mathcal{C}_2)$. We will use this modification of the procedure to give an estimate of its run-time complexity.

Let n denote the length of $\mathcal{C}$, let $k = \dim(\mathcal{C})$, and let $E = E(T)$. Also, define

$$r_{\max} = \max_{e \in E} \dim(\mathcal{C}|_{J(e)}) + \dim(\mathcal{C}|_{\overline{J}(e)}) - \dim(\mathcal{C}). \tag{21}$$

Observe that, as a result of the modification suggested above, in the determination of $\mathcal{M}(\mathcal{C}; T, \omega)$, the procedure `MIN_REALZN` gets called $|E|$ times, once for each edge $e \in E$. The run-time complexity of any particular run of `MIN_REALZN` is determined by the computations in Step M2. In the ith run, the procedure acts upon some code $\mathcal{C}^{(i)}$ of length n_i and dimension k_i, and in Step M2, it computes an r_i, a code Δ_{r_i} with index set $I_\Delta^{(i)}$, and a code $\mathcal{C}_1^{(i)}$. Via Lemma E.1, we have that $r_i \le r_{\max}$. We bound k_i and n_i as follows. Note that $\mathcal{C}_1^{(i)}$ is the code $\mathcal{C}^{(i+1)}$ that the $(i+1)$th run of the procedure takes as input. Thus, we have $k_{i+1} \le k_i$, and $n_{i+1} \le n_i + |I_\Delta^{(i)}|$. Since $k_1 = k$, $n_1 = n$, and $|I_\Delta^{(i)}| = (q^{r_i} - 1)/(q-1) \le q^{r_{\max}}$, we have, for $i = 1, 2, \ldots, |E|$, $k_i \le k$ and $n_i \le n + (i-1)q^{r_{\max}}$. Now, by the estimate given in Section 4 of the run-time complexity of the r-sum decomposition procedure, we see that the ith run of Step M2 of `MIN_REALZN` takes $O(k_i^2 n_i + n_i r_i q^{r_i})$ time. Hence the overall run-time complexity of `MIN_REALZN` may be estimated to be $\sum_{i=1}^{|E|} O(k_i^2 n_i + n_i r_i q^{r_i})$. This expression can be simplified by observing that

$$\begin{aligned}
\sum_{i=1}^{|E|} (k_i^2 n_i + n_i r_i q^{r_i}) &\le (k^2 + r_{\max} q^{r_{\max}}) \sum_{i=1}^{|E|} n_i \\
&\le (k^2 + r_{\max} q^{r_{\max}}) \sum_{i=1}^{|E|} [n + (i-1) q^{r_{\max}}] \\
&= (k^2 + r_{\max} q^{r_{\max}})\, [n|E| + (1/2)|E|(|E|-1) q^{r_{\max}}]\,.
\end{aligned}$$

It follows that `MIN_REALZN` runs in $O((k^2 + r_{\max} q^{r_{\max}})(n|E| + |E|^2 r_{\max} q^{r_{\max}}))$ time. Note that this is polynomial in n, k, q and $|E|$, but exponential in $r_{\max}$.

6. Complexity Measures

6.1. **Complexity Measures for Codes.** As observed in [7], any graphical realization of a code specifies an associated decoding algorithm, namely, the sum-product algorithm. The sum-product algorithm specified by a tree realization, $\Gamma = (T, \omega, (\mathcal{S}_e,\, e \in E),\, (C_v,\, v \in V))$, of a code $\mathcal{C}$ provides an exact implementation of ML decoding for $\mathcal{C}$. A reasonable initial estimate of the computational complexity of the sum-product algorithm on Γ is provided by the *constraint complexity* of Γ, which is defined as $\max_{v \in V} \dim(C_v)$. As implied by Corollary 3.5, given a tree decomposition (T, ω) of the index set of $\mathcal{C}$, the minimal realization $\mathcal{M}(\mathcal{C}; T, \omega)$ has the least constraint

complexity among all tree realizations of $\mathcal{C}$ that extend (T, ω). Let $\kappa(\mathcal{C}; T, \omega)$ denote the constraint complexity of $\mathcal{M}(\mathcal{C}; T, \omega)$. Note that, by (6),

$$\kappa(\mathcal{C}; T, \omega) = \max_{v \in V} \left(\dim(\mathcal{C}) - \sum_{e \in E(v)} \dim(\mathcal{C}_{J(e)}) \right). \tag{22}$$

Thus, $\kappa(\mathcal{C}; T, \omega)$ is a measure of the complexity of implementing ML decoding for $\mathcal{C}$ as a sum-product algorithm on $\mathcal{M}(\mathcal{C}; T, \omega)$.

Let us now define the *treewidth* of the code $\mathcal{C}$ to be

$$\kappa(\mathcal{C}) = \min_{(T,\omega)} \kappa(\mathcal{C}; T, \omega), \tag{23}$$

where the minimum is taken over all tree decompositions (T, ω) of the index set of $\mathcal{C}$. The treewidth of a code is an indicator of how small the computational complexity of an ML decoding algorithm for $\mathcal{C}$ can be. The notion of treewidth (*i.e.*, minimal constraint complexity) of a code was first introduced by Forney [8]. A related notion, called minimal tree complexity, was defined and studied by Halford and Chugg [9]. Treewidth, as defined in (23), is an upper bound on the minimal tree complexity measure of Halford and Chugg.

A tree is called *cubic* if all its internal nodes have degree 3. Forney [8] showed that the minimum in (23) is always achieved by a tree decomposition (T, ω) in which T is a cubic tree, and ω is a bijection[6] between the index set of $\mathcal{C}$ and the set of leaves of T. Let $\mathcal{Q}(\mathcal{C})$ denote the set of all tree decompositions (T, ω) in which T is cubic and ω maps the index set of $\mathcal{C}$ bijectively onto the set of leaves of T. We may then re-write (23) as

$$\kappa(\mathcal{C}) = \min_{(T,\omega) \in \mathcal{Q}(\mathcal{C})} \kappa(\mathcal{C}; T, \omega). \tag{24}$$

An alternate measure of code complexity may be obtained from the notion of *state complexity* of a tree realization Γ, which is the largest dimension of a state space in Γ. Thus, by virtue of (4) and (5), the state complexity of a minimal realization $\mathcal{M}(\mathcal{C}; T, \omega)$ is given by

$$\begin{aligned} \sigma(\mathcal{C}; T, \omega) &= \max_{e \in E} \dim(\mathcal{C}) - \dim(\mathcal{C}_{J(e)}) - \dim(\mathcal{C}_{\overline{J}(e)}) \\ &= \max_{e \in E} \dim(\mathcal{C}|_{J(e)}) + \dim(\mathcal{C}|_{\overline{J}(e)}) - \dim(\mathcal{C}). \end{aligned} \tag{25}$$

We then define, in analogy with (24),

$$\sigma(\mathcal{C}) = \min_{(T,\omega) \in \mathcal{Q}(\mathcal{C})} \sigma(\mathcal{C}; T, \omega). \tag{26}$$

Note that the minimum in the above definition is taken over tree decompositions in $\mathcal{Q}(\mathcal{C})$ only. It must be emphasized that $\sigma(\mathcal{C})$, as defined in (26), need *not* be the same as the least $\sigma(\mathcal{C}; T, \omega)$ over all tree decompositions (T, ω) of the index set of $\mathcal{C}$.

A notion analogous to $\sigma(\mathcal{C})$ is known as branchwidth in the matroid theory literature; see *e.g.*, [15]. In keeping with that nomenclature, we will call $\sigma(\mathcal{C})$ the *branchwidth* of the code $\mathcal{C}$. Branchwidth and treewidth are very closely related, as shown by the following result, which can be obtained in a straightforward manner from the bounds in Lemma 2.3.

Proposition 6.1 ([16], Theorem 4.2)**.** *Given a code $\mathcal{C}$, if $(T, \omega) \in \mathcal{Q}(\mathcal{C})$, then*

$$\sigma(\mathcal{C}; T, \omega) \leq \kappa(\mathcal{C}; T, \omega) \leq 2\sigma(\mathcal{C}; T, \omega).$$

Hence, $\sigma(\mathcal{C}) \leq \kappa(\mathcal{C}) \leq 2\sigma(\mathcal{C})$.

[6]Forney [8] only explicitly states that the minimizing (T, ω) may be taken to be such that T is a cubic tree and ω is a surjective map onto the leaves of T. However, the symbol-splitting argument in Section V.F of his paper actually implies that ω in the minimizing tree decomposition may be taken to be one-to-one as well.

The notions of state and constraint complexity have been studied extensively in the context of conventional trellis realizations of a code; see *e.g.*, [31]. Recall from Example 2.2 that a conventional trellis realization of a code is a tree realization that extends a tree decomposition (T, ω) in which T is a simple path and ω is a bijection between the index set of $\mathcal{C}$ and the vertices of T. This special case of a tree decomposition is referred to as a *path decomposition*. Specifically, a path decomposition of a code $\mathcal{C}$ defined on the index set I is a pair (T, ω), where T is a simple path on $|I|$ vertices, and $\omega : I \to V(T)$ is a bijection. Let $\mathcal{P}(\mathcal{C})$ denote the set of all path decompositions of $\mathcal{C}$. We then define

$$\kappa_{\text{trellis}}(\mathcal{C}) = \min_{(T,\omega)\in\mathcal{P}(\mathcal{C})} \kappa(\mathcal{C}; T, \omega) \tag{27}$$

and

$$\sigma_{\text{trellis}}(\mathcal{C}) = \min_{(T,\omega)\in\mathcal{P}(\mathcal{C})} \sigma(\mathcal{C}; T, \omega). \tag{28}$$

It is well-known, and indeed readily follows from Lemma 2.3, that $\sigma_{\text{trellis}}(\mathcal{C}) \leq \kappa_{\text{trellis}}(\mathcal{C}) \leq \sigma_{\text{trellis}}(\mathcal{C}) + 1$.

It is clear from (23) and (27) that $\kappa(\mathcal{C}) \leq \kappa_{\text{trellis}}(\mathcal{C})$. Forney [8] asked the question of whether $\kappa(\mathcal{C})$ could be significantly smaller than $\kappa_{\text{trellis}}(\mathcal{C})$. He conjectured that either $\kappa_{\text{trellis}}(\mathcal{C}) - \kappa(\mathcal{C}) \leq 1$ for all codes $\mathcal{C}$, or $\kappa_{\text{trellis}}(\mathcal{C}) - \kappa(\mathcal{C})$ is unbounded. We show here that it is in fact the latter that is true. To do so, we need to introduce some new concepts.

6.2. **Complexity Measures for Graphs.** In their fundamental work on graph minors [27], Robertson and Seymour introduced two notions of complexity of graphs, namely, treewidth and pathwidth. These notions have proved to be invaluable tools with many applications in graph theory and theoretical computer science. An overview of such applications can be found, for example, in [6]. We will define the notions of treewidth and pathwidth of a graph in this subsection, and subsequently, relate them to the complexity measures $\kappa(\mathcal{C})$ and $\kappa_{\text{trellis}}(\mathcal{C})$ defined above for codes.

Let $\mathcal{G}$ be a graph with vertex set $V(\mathcal{G})$ and edge set $E(\mathcal{G})$. The graph may contain self-loops and parallel edges. A *tree decomposition* of $\mathcal{G}$ is a pair (T, β), where T is a tree, and $\beta : V(T) \to 2^{V(\mathcal{G})}$ is a mapping that satisfies the following:

- (T1) $\bigcup_{x\in V(T)} \beta(x) = V(\mathcal{G})$;
- (T2) for each pair of adjacent vertices $u, v \in V(\mathcal{G})$, we have $\{u, v\} \subseteq \beta(x)$ for some $x \in V(T)$; and
- (T3) for each pair of vertices $x, z \in V(T)$, if $y \in V(T)$ is any vertex on the unique path between x and z, then $\beta(x) \cap \beta(z) \subseteq \beta(y)$.

It may be helpful to point out that (T3) above is equivalent to the following:

- (T3′) for each $v \in V(\mathcal{G})$, the subgraph of T induced by $\{x \in V(T) : v \in \beta(x)\}$ is a (connected) subtree of T.

A reader familiar with the notion of "junction trees" (see *e.g.*, [1]) will recognize a tree decomposition of $\mathcal{G}$ to be a junction tree.

The *width* of a tree decomposition (T, β) as above is defined to be $\max_{x\in V(T)} |\beta(x)| - 1$. The *treewidth* of $\mathcal{G}$, which we denote by $\kappa(\mathcal{G})$, is the minimum among the widths of all its tree decompositions. Note that if $\mathcal{G}$ has at least one edge, then, because of (T2), any tree decomposition of $\mathcal{G}$ must have width at least one. Thus, for any graph $\mathcal{G}$ with $|E(\mathcal{G})| \geq 1$, we have $\kappa(\mathcal{G}) \geq 1$.

Example 6.1. *For any tree T with at least two vertices, we have $\kappa(T) = 1$. This can be seen as follows. Fix a vertex $r \in V(T)$. Define a mapping $\beta : V(T) \to 2^{V(T)}$ as follows: $\beta(r) = \{r\}$, and for $x \neq r$, $\beta(x) = \{x, y\}$, where $\{x, y\}$ is the first edge on the unique path from x to r. It is easily verified that (T, β) is a tree decomposition of T. Since this tree decomposition has width one, it follows that $\kappa(T) = 1$.*

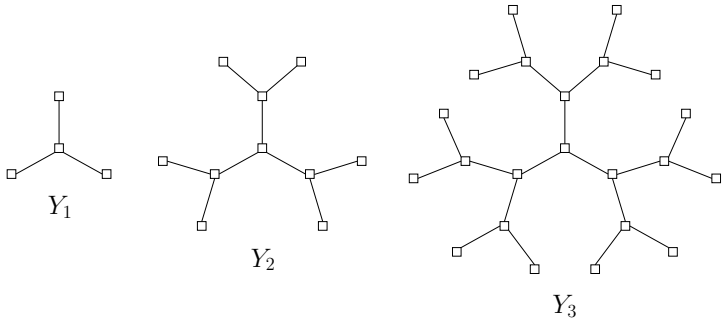

FIGURE 8. The trees Y_1, Y_2 and Y_3.

If (T, β) is a tree decomposition in which T is a simple path, then (T, β) is called a *path decomposition*. The minimum among the widths of all the path decompositions of $\mathcal{G}$ is called the *pathwidth* of $\mathcal{G}$, which we denote by $\kappa_{\text{path}}(\mathcal{G})$. It is evident that $\kappa(\mathcal{G}) \leq \kappa_{\text{path}}(\mathcal{G})$.

Analogous to the situation of Example 6.1, a simple path has pathwidth one. However, trees may have arbitrarily large pathwidth. The following example is due to Robertson and Seymour [26].

Example 6.2. *Let Y_1 be the complete bipartite graph $K_{1,3}$. For $i \geq 2$, we inductively define Y_i by taking a copy of Y_{i-1}, and to each leaf v of this graph, adding two new vertices adjacent to v. Figure 8 shows the trees Y_1, Y_2 and Y_3. The pathwidth of Y_i, $i \geq 1$, is $\lceil \frac{1}{2}(i+1) \rceil$* [26].

Thus, for trees T, the difference $\kappa_{\text{path}}(T) - \kappa(T)$ can be arbitrarily large. We will use this fact to construct codes $\mathcal{C}$ for which $\kappa_{\text{trellis}}(\mathcal{C}) - \kappa(\mathcal{C})$ is arbitrarily large.

We remark that the problem of determining the treewidth or pathwidth of a graph is known to be NP-hard [2],[6]. As we will see a little later, this implies that the problem of determining the treewidth of a code, or its trellis counterpart, is also NP-hard.

6.3. **Relating the Complexity Measures for Codes and Graphs.** Let $\mathbb{F}$ be an arbitrary finite field. To any given graph $\mathcal{G}$, we will associate a code $\mathcal{C}[\mathcal{G}]$ over $\mathbb{F}$ as follows. Let $D(\mathcal{G})$ be any directed graph obtained by arbitrarily assigning orientations to the edges of $\mathcal{G}$, and let $A_{D(\mathcal{G})}$ be the vertex-edge incidence matrix of $D(\mathcal{G})$. This is the $|V(\mathcal{G})| \times |E(\mathcal{G})|$ matrix whose rows and columns are indexed by the vertices and directed edges, respectively, of $D(\mathcal{G})$, and whose (i,j)th entry, $a_{i,j}$, is determined as follows:

$$a_{i,j} = \begin{cases} 1 & \text{if vertex } i \text{ is the tail of non-loop edge } j \\ -1 & \text{if vertex } i \text{ is the head of non-loop edge } j \\ 0 & \text{otherwise.} \end{cases}$$

The code $\mathcal{C}[\mathcal{G}]$ is defined to be the linear code over $\mathbb{F}$ generated by the matrix $A_{D(\mathcal{G})}$. When $\mathbb{F}$ is the binary field, the code $\mathcal{C}[\mathcal{G}]$ is the *cut-set code* of $\mathcal{G}$, *i.e.*, the dual of the cycle code of $\mathcal{G}$ [10].

The following fundamental result that relates the treewidths of the graph $\mathcal{G}$ and the code $\mathcal{C}[\mathcal{G}]$ is due to Hliněný and Whittle[7][16],[17].

Theorem 6.2 ([16], Theorem 3.2)**.** *If $\mathcal{G}$ is a graph with at least one edge, then $\kappa(\mathcal{G}) = \kappa(\mathcal{C}[\mathcal{G}])$.*

Since determining the treewidth of a graph is NP-hard, it immediately follows from the above theorem that the problem of determining the treewidth of a code (over any fixed finite field) is also NP-hard. We remark that the problem of determining the branchwidth of a code is also NP-hard. This follows from a result [11] that relates the branchwidth of the code $\mathcal{C}[\mathcal{G}]$ to the branchwidth of the graph $\mathcal{G}$, the latter being a notion we have not defined in this paper.

[7]The results in [16],[17] are stated in matroid-theoretic language. The vocabulary necessary to translate the language of matroid theory into that of coding theory can be found, for example, in [19].

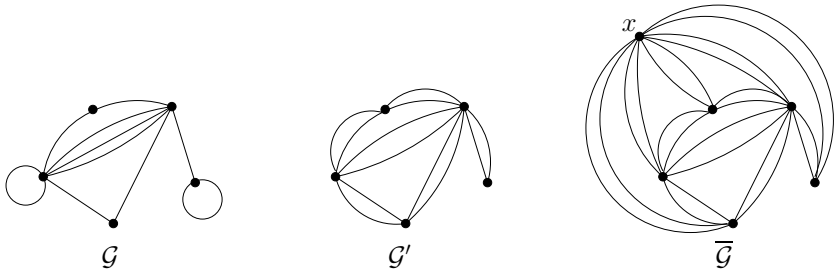

FIGURE 9. Construction of $\mathcal{G}'$ and $\overline{\mathcal{G}}$ from $\mathcal{G}$.

Unfortunately, it is not true that $\kappa_{\text{path}}(\mathcal{G}) = \kappa_{\text{trellis}}(\mathcal{C}[\mathcal{G}])$. As an example, consider the code $\mathcal{C}[T]$ over the binary field, for an arbitrary tree T. It is not hard to see that $\mathcal{C}[T] = \{0,1\}^{|E(T)|}$ which, being the direct sum of multiple copies of $\{0,1\}$, has $\kappa_{\text{trellis}}(\mathcal{C}[T]) = 1$. But as we have already noted, trees can have arbitrarily large pathwidth.

We get around this problem by means of a suitable transformation of graphs. Given a graph $\mathcal{G}$, let $\mathcal{G}'$ be a graph defined on the same vertex set as $\mathcal{G}$, having the following properties (see Figure 9):

- $\mathcal{G}'$ is loopless;
- a pair of distinct vertices is adjacent in $\mathcal{G}'$ iff it is adjacent in $\mathcal{G}$; and
- in $\mathcal{G}'$, there are exactly two edges between each pair of adjacent vertices.

Define $\overline{\mathcal{G}}$ to be the graph obtained by adding an extra vertex, x, to $\mathcal{G}'$, along with a pair of parallel edges from x to each $v \in V(\mathcal{G}')$ (see Figure 9). It is easy to see that $\overline{\mathcal{G}}$ is constructible directly from $\mathcal{G}$ in $O(|V(\mathcal{G})|^2)$ time.

The following result was used in [19] to show that the problem of determining $\sigma_{\text{trellis}}(\mathcal{C})$ for an arbitrary code $\mathcal{C}$ (over any fixed finite field) is NP-hard.

Theorem 6.3 ([19], Proposition 3.1)**.** *If $\overline{\mathcal{G}}$ is the graph constructed from a given graph $\mathcal{G}$ as described above, then $\sigma_{trellis}(\mathcal{C}[\overline{\mathcal{G}}]) = \kappa_{path}(\mathcal{G}) + 1$.*

Since $\sigma_{\text{trellis}}(\mathcal{C})$ is always within one of $\kappa_{\text{trellis}}(\mathcal{C})$, the above theorem implies that

$$\kappa_{\text{path}}(\mathcal{G}) + 1 \leq \kappa_{\text{trellis}}(\mathcal{C}[\overline{\mathcal{G}}]) \leq \kappa_{\text{path}}(\mathcal{G}) + 2. \tag{29}$$

While this falls short of establishing the NP-hardness of computing $\kappa_{\text{trellis}}(\mathcal{C})$ for an arbitrary code $\mathcal{C}$, it is certainly enough to provide us with the desired example of codes $\mathcal{C}$ for which $\kappa_{\text{trellis}}(\mathcal{C}) - \kappa(\mathcal{C})$ is arbitrarily large. We just need to make one more observation: $\kappa(\overline{\mathcal{G}}) = \kappa(\mathcal{G}) + 1$. The proof of this fact, which is along the lines of the proof of Lemma 3.5 in [19], is left to the reader as a straightforward exercise. We can now prove the following corollary to Theorems 6.2 and 6.3.

Corollary 6.4. *Over any finite field $\mathbb{F}$, there exists a family of codes $\mathcal{C}_i$, $i \in 1, 2, \ldots$, such that*

$$\lim_i \ \kappa_{trellis}(\mathcal{C}_i) - \kappa(\mathcal{C}_i) = \infty.$$

Proof. Let Y_i, $i = 1, 2, \ldots$, be the family of trees defined in Example 6.2. Define $\mathcal{C}_i = \mathcal{C}[\overline{Y_i}]$, where $\overline{Y_i}$ refers to the graph obtained from Y_i by the transformation depicted in Figure 9. Note that $\kappa(\overline{Y_i}) = \kappa(Y_i) + 1 = 2$, since $\kappa(Y_i) = 1$, as shown in Example 6.1. Thus, on the one hand, from Theorem 6.2, we have $\kappa(\mathcal{C}_i) = \kappa(\overline{Y_i}) = 2$. And on the other hand, from (29) and Example 6.2, we have $\kappa_{\text{trellis}}(\mathcal{C}_i) \geq \kappa_{\text{path}}(Y_i) + 1 = \lceil \frac{1}{2}(i+3) \rceil$. □

Using standard facts known about the incidence matrix $A_{D(\mathcal{G})}$ for a graph $\mathcal{G}$ (see, for example, [25, Chapter 5]), it may be verified that the codes $\mathcal{C}_i$, $i \geq 1$, constructed in the above proof are

$[n_i, k_i, d_i]$ codes, where

$$\begin{aligned} n_i &= |E(\overline{Y_i})| = 12(2^i - 1) + 2, \\ k_i &= |V(\overline{Y_i})| - 1 = 3(2^i - 1) + 1, \\ d_i &= \text{size of the smallest cut-set in } \overline{Y_i} = 4. \end{aligned}$$

Note that $\kappa_{\text{trellis}}(\mathcal{C}_i) - \kappa(\mathcal{C}_i)$ grows as $O(\log n_i)$. It is tempting to conjecture that this is in fact the maximal rate of growth of the difference $\kappa_{\text{trellis}}(\mathcal{C}_i) - \kappa(\mathcal{C}_i)$ for any code family $\mathcal{C}_i$, but this is unlikely to be true. What is true, however, is that the *ratio* $\kappa_{\text{trellis}}(\mathcal{C}_i)/\kappa(\mathcal{C}_i)$ can grow at most logarithmically with the length of the code. Indeed, it has been shown in [20] that for any code $\mathcal{C}$ of length $n > 1$,

$$\frac{\kappa_{\text{trellis}}(\mathcal{C})}{\kappa(\mathcal{C})} \leq 2\log_2(n-1) + 3. \tag{30}$$

The above bound on the ratio of $\kappa_{\text{trellis}}(\mathcal{C})$ to $\kappa(\mathcal{C})$ is the best possible, up to the constants involved. Indeed, for the codes $\mathcal{C}_i$ constructed in the proof of Corollary 6.4, we have that

$$\frac{\kappa_{\text{trellis}}(\mathcal{C}_i)}{\kappa(\mathcal{C}_i)} \geq \frac{i+3}{4} = \frac{1}{4}\left[\log_2(n_i + 10) - \log_2(3/2)\right].$$

The codes $\mathcal{C}_i$ above constitute an example of a family of codes whose treewidth is bounded by a constant. Issues related to such code families are discussed next.

6.4. **Codes of Bounded Complexity.** Many NP-hard combinatorial problems on graphs are known to be solvable in polynomial (often, linear) time when restricted to graphs of bounded treewidth [3],[5]. In this subsection, we will see that the same general principle applies to problems pertaining to codes as well.

Let $\mathbb{F}_q = GF(q)$ be a fixed finite field. Given an integer $t \geq 0$, denote by $\mathsf{TW}(t)$ (resp. $\mathsf{BW}(t)$) the family of all codes over $\mathbb{F}_q$ of treewidth (resp. branchwidth) at most t. Thus, a family $\mathfrak{C}$ of codes over $\mathbb{F}_q$ is said to have bounded treewidth (resp. branchwidth) if $\mathfrak{C} \subseteq \mathsf{TW}(t)$ (resp. $\mathfrak{C} \subseteq \mathsf{BW}(t)$) for some integer t. Note that by Proposition 6.1, $\mathsf{BW}(\lfloor t/2 \rfloor) \subseteq \mathsf{TW}(t) \subseteq \mathsf{BW}(t)$, and so, a code family $\mathfrak{C}$ has bounded treewidth if and only if it has bounded branchwidth.

A fundamental result of coding theory [4] states that the problem of ML decoding is NP-hard for an arbitrary family of codes. However, we will now show that this problem becomes solvable in linear time for any code family of bounded treewidth. So, consider a code family $\mathfrak{C} \subseteq \mathsf{TW}(t)$, where t is a *fixed* integer, and pick an arbitrary $\mathcal{C} \in \mathfrak{C}$. Let n denote the length of $\mathcal{C}$. By definition, $\mathcal{C}$ has a minimal realization $\mathcal{M}(\mathcal{C}; T, \omega)$ with constraint complexity at most t. Moreover, by (24), (T, ω) can be chosen to be in $\mathcal{Q}(\mathcal{C})$, *i.e.*, it may be chosen so that T is a cubic tree, and ω maps the index set of $\mathcal{C}$ bijectively onto the leaves of T. In particular, the number of leaves of T equals the cardinality, n, of the index set of $\mathcal{C}$.

Now, recall that ML decoding of $\mathcal{C}$ may be implemented as a sum-product algorithm on any tree realization of $\mathcal{C}$, and in particular, on $\mathcal{M}(\mathcal{C}; T, \omega)$. The computational complexity of the sum-product algorithm on $\mathcal{M}(\mathcal{C}; T, \omega)$ is determined by the computations that take place at the internal nodes of T. By an estimate of Forney [7, Theorem 5.2], the number of computations at the internal node $v \in V(T)$ is of the order of $\delta_v(\delta_v - 2)q^{\dim(C_v^*)}$, where δ_v is the degree of v in T. Since T is cubic, $\delta_v = 3$, and since the constraint complexity of $\mathcal{M}(\mathcal{C}; T, \omega)$ is at most t, we have $\dim(C_v^*) \leq t$. Hence, the number of computations performed by the sum-product algorithm at any internal node of T is bounded by $3q^t$, which is a constant. Now, T is a cubic tree on n leaves, so it has at most $n - 2$ internal nodes. It follows that the computational complexity of the sum-product algorithm on $\mathcal{M}(\mathcal{C}; T, \omega)$ is $O(n)$, the constant in the O-notation being proportional to $3q^t$. Thus, there is a linear-time implementation of ML decoding for any $\mathcal{C} \in \mathfrak{C}$.

A question that naturally arises in this context is that of how hard it is to explicitly determine the minimal tree realization required for linear-time implementation of ML decoding. Note that this is

not exactly a decoding complexity issue, since the determination of a suitable $\mathcal{M}(\mathcal{C};T,\omega)$ may be done "off-line" for each $\mathcal{C} \in \mathfrak{C}$.

An explicit determination of $\mathcal{M}(\mathcal{C};T,\omega)$ involves finding a tree decomposition $(T,\omega) \in \mathcal{Q}(\mathcal{C})$ such that $\kappa(\mathcal{C};T,\omega) \leq t$, and the specification of the state spaces and the local constraint codes of $\mathcal{M}(\mathcal{C};T,\omega)$. Given a $(T,\omega) \in \mathcal{Q}(\mathcal{C})$ satisfying $\kappa(\mathcal{C};T,\omega) \leq t$, the state spaces and local constraint codes of $\mathcal{M}(\mathcal{C};T,\omega)$ may be determined by the `MIN_REALZN` procedure of Section 5. An estimate of the computational complexity of this procedure was given in that section, in terms of the length and dimension of $\mathcal{C}$, the number of edges in T, and the quantity $r_{\max}$ defined in (21). Comparing (21) with (25), we see that $r_{\max}$ is simply $\sigma(\mathcal{C};T,\omega)$, which by Proposition 6.1, is bounded from above by t. The number of edges of T is $|V(T)|-1$, and $V(T)$ consists of n leaves and at most $n-2$ internal nodes, n being the length of $\mathcal{C}$. Therefore, by the estimate of the computational complexity of `MIN_REALZN` given in Section 5, for an $[n,k]$ code $\mathcal{C} \in \mathfrak{C}$, and a $(T,\omega) \in \mathcal{Q}(\mathcal{C})$ such that $\kappa(\mathcal{C};T,\omega) \leq t$, the minimal realization $\mathcal{M}(\mathcal{C};T,\omega)$ may be constructed in $O(k^2n^2)$ time. Note that t appears in the exponent of the constant implicit in the O-notation.

This leaves us with the problem of finding, for a given code $\mathcal{C} \in \mathsf{TW}(t)$, a tree decomposition $(T,\omega) \in \mathcal{Q}(\mathcal{C})$ such that $\kappa(\mathcal{C};T,\omega) \leq t$. Unfortunately, there appears to be no efficient algorithm known for solving this problem. However, reasonably good algorithms do exist for solving a closely related problem: given a code $\mathcal{C} \in \mathsf{BW}(t)$, find a tree decomposition $(T,\omega) \in \mathcal{Q}(\mathcal{C})$ such that $\sigma(\mathcal{C};T,\omega) \leq t$. Several polynomial-time algorithms for solving this problem are given in [14], the most efficient of these being an algorithm that runs in $O(n^3)$ time[8], n being the length of $\mathcal{C}$. Now, by Proposition 6.1, any $\mathcal{C} \in \mathsf{TW}(t)$ is also in $\mathsf{BW}(t)$, and furthermore, $\kappa(\mathcal{C};T,\omega) \leq 2\sigma(\mathcal{C};T,\omega)$. Therefore, the algorithms of [14] find, for a given code $\mathcal{C} \in \mathsf{TW}(t)$, a tree decomposition $(T,\omega) \in \mathcal{Q}(\mathcal{C})$ such that $\kappa(\mathcal{C};T,\omega) \leq 2t$. This is sufficient for our purposes, as the computational complexity of the sum-product algorithm on the resulting $\mathcal{M}(\mathcal{C};T,\omega)$ would still be $O(n)$, except that the constant in the O-notation would now be proportional to $3q^{2t}$.

While code families of bounded treewidth have the desirable property of having linear decoding complexity, they tend to have poor error-correcting properties. We give here an argument to support this statement. Recall from coding theory that a code family $\mathfrak{C}$ is called *asymptotically good* if there exists a sequence of $[n_i,k_i,d_i]$ codes $\mathcal{C}_i \in \mathfrak{C}$, with $\lim_i n_i = \infty$, such that $\liminf_i k_i/n_i$ and $\liminf_i d_i/n_i$ are both strictly positive. The code family $\mathcal{C}_i$, $i \geq 1$, from the proof of Corollary 6.4 has bounded treewidth, but is not asymptotically good: $k_i/n_i \to 1/4$, but $d_i/n_i \to 0$, as $i \to \infty$. More generally, it has recently been shown that for any $t > 0$, the code family $\mathsf{TW}(t)$ is not asymptotically good [20].

We wrap up our discussion on complexity measures for codes by elaborating on a comment we made at the beginning of this subsection, in which we implied that hard coding-theoretic problems often become polynomial-time solvable when restricted to codes of bounded complexity. We saw earlier several examples of algorithms that, given a code $\mathcal{C} \in \mathsf{TW}(t)$, solve some problem in time polynomial in the length of $\mathcal{C}$. In each of these cases, the computational complexity of the algorithm displayed an exponential dependence on the parameter t. But since t was a fixed constant, this exponential dependence could be absorbed into the constant hidden in the "big-O" estimate of the complexity. Thus, fixing the parameter t allowed a potentially intractable coding-theoretic problem to become tractable. Problems that may be hard in general, but which become solvable in polynomial time when one of the parameters of the problem is fixed, are called *fixed-parameter tractable*. We noted previously that the problems of computing the treewidth and branchwidth of a code are NP-hard. It should come as no surprise that these problems are in fact fixed-parameter tractable. Hliněný [12] gives an $O(n^3)$ algorithm that, for a fixed integer t, determines whether or not a given length-n code is in $\mathsf{BW}(t)$. From this, one can also prove the existence of an $O(n^3)$ algorithm for deciding membership of a given length-n code in $\mathsf{TW}(t)$ [13].

[8]As usual, the constant hidden in the O-notation depends exponentially on t.

7. Concluding Remarks

As we have pointed out in this paper, the problem of computing the treewidth $\kappa(\mathcal{C})$ of a code $\mathcal{C}$ is NP-hard. This means that there can be (modulo the unlikely scenario that $P = NP$) no efficient — meaning polynomial-time — algorithm for computing the treewidth of an arbitrary code. However, it may still be possible to determine exactly, or give efficient algorithms for computing, the treewidth of standard algebraic codes such as Hamming codes, Reed-Muller codes, Golay codes etc. To the best of our knowledge, there has been no prior work done on this subject. In fact, except for the pioneering work of Forney [7],[8], no serious attempt seems to have been made on the problem of finding general constructions of tree realizations for such codes, having constraint complexity equal to, or close to, the treewidth of the code. This, perhaps, constitutes the most important theoretical problem in the context of tree realizations of linear codes. Note that explicit constructions of such low-complexity tree realizations would be useful to a code designer who wishes to employ the sum-product algorithm for decoding algebraic codes.

However, an open problem of far greater significance is the development of a general theory of minimal realizations of codes on graphs with cycles. At present, such a theory only exists for the case of realizations of codes on graphs consisting of a single cycle, *i.e.*, tail-biting trellis realizations [22]. This simplest case of graphs with cycles is already more difficult to study than the cycle-free case — for example, there can be several non-equivalent definitions of minimality in the context of tail-biting trellis realizations. The challenge posed by graphs with more complex cycle structures can only be greater.

Appendix A. Proofs of Lemmas 2.1 and 2.2

Proof of Lemma 2.1. Consider an arbitrary $e \in E$. An arbitrary global configuration $\mathbf{b}$ may be written in the form $(\mathbf{b}|_{J(e)}, \mathbf{b}|_{E(T_e)}, \mathbf{b}|_e, \mathbf{b}|_{E(\overline{T}_e)}, \mathbf{b}|_{\overline{J}(e)})$. Now, suppose that $\mathbf{b}$ is such that $\mathbf{b}|_e = \mathbf{0}$, *i.e.*, $\mathbf{b} = (\mathbf{b}|_{J(e)}, \mathbf{b}|_{E(T_e)}, \mathbf{0}, \mathbf{b}|_{E(\overline{T}_e)}, \mathbf{b}|_{\overline{J}(e)})$. Observe that the global configurations

$$\mathbf{b}' = (\mathbf{b}|_{J(e)}, \mathbf{b}|_{E(T_e)}, \mathbf{0}, \mathbf{0}, \mathbf{0}) \quad \text{and} \quad \mathbf{b}'' = (\mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{b}|_{E(\overline{T}_e)}, \mathbf{b}|_{\overline{J}(e)})$$

also satisfy all local constraints (since $\mathbf{0} \in \mathfrak{B}|_v$ for each $v \in V$), and hence are in $\mathfrak{B}$. Therefore, $(\mathbf{b}|_{J(e)}, \mathbf{0}) = \mathbf{b}'|_I \in \mathcal{C}$, and so by definition of $\mathcal{C}_{J(e)}$, we have $\mathbf{b}|_{J(e)} \in \mathcal{C}_{J(e)}$. Similarly, $(\mathbf{0}, \mathbf{b}|_{\overline{J}(e)}) = \mathbf{b}''|_I \in \mathcal{C}$, so that $\mathbf{b}|_{\overline{J}(e)} \in \mathcal{C}_{\overline{J}(e)}$. Hence, $\mathbf{b}|_I = (\mathbf{b}|_{J(e)}, \mathbf{b}|_{\overline{J}(e)}) \in \mathcal{C}_{J(e)} \oplus \mathcal{C}_{\overline{J}(e)}$. $\square$

Proof of Lemma 2.2. For any tree model (essential or not), we have, by definition, $\mathfrak{B}|_v \subseteq C_v$ for all $v \in V$. So we need only show the reverse inclusion in the case when $\Gamma = (T, \omega, (\mathcal{S}_e,\, e \in E),\, (C_v,\, v \in V))$ is an essential tree model.

Pick an arbitrary $v \in V$. Let $e_1, e_2, \ldots, e_\delta$ be the edges of T incident with V. For $i = 1, 2, \ldots, \delta$, let T_i denote the component of $T - e_i$ that does not include v. Set $F_i = E(T_i)$, and $J_i = \omega^{-1}(V(T_i))$. We will write an arbitrary configuration $\mathbf{b} \in \mathfrak{B}$ as

$$\left(\mathbf{b}|_{\omega^{-1}(v)},\, (\mathbf{b}|_{e_1}, \mathbf{b}|_{F_1}, \mathbf{b}|_{J_1}), \ldots, (\mathbf{b}|_{e_\delta}, \mathbf{b}|_{F_\delta}, \mathbf{b}|_{J_\delta})\right).$$

Consider any $(\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_\delta) \in C_v$, where $\mathbf{c}_0 \in \mathbb{F}^{\omega^{-1}(v)}$, and $\mathbf{c}_i \in \mathcal{S}_{e_i}$ for $i = 1, \ldots, \delta$. As the tree model Γ is essential, we have $\mathcal{S}_{e_i} = \mathfrak{B}|_{e_i}$ for all i. In particular, $\mathbf{c}_i \in \mathfrak{B}|_{e_i}$, so that there exists $\mathbf{b}^{(i)} \in \mathfrak{B}$ such that $\mathbf{b}^{(i)}|_{e_i} = \mathbf{c}_i$. As $\mathbf{b}^{(i)}$ is in $\mathfrak{B}$, its "sub-configuration" $(\mathbf{c}_i, \mathbf{b}^{(i)}|_{F_i}, \mathbf{b}^{(i)}|_{J_i})$ satisfies the local constraints of Γ at all vertices in $V(T_i)$. Hence,

$$\overline{\mathbf{b}} \;=\; \left(\mathbf{c}_0,\, (\mathbf{c}_1, \mathbf{b}^{(1)}|_{F_1}, \mathbf{b}^{(1)}|_{J_1}), \ldots, (\mathbf{c}_\delta, \mathbf{b}^{(\delta)}|_{F_\delta}, \mathbf{b}^{(\delta)}|_{J_\delta})\right)$$

satisfies the local constraints of Γ at all vertices in $\bigcup_{i=1}^{\delta} V(T_i)$. Now, v is the only vertex of T that is not in $\bigcup_{i=1}^{\delta} V(T_i)$. But, by construction, $\overline{\mathbf{b}}|_v = (\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_\delta) \in C_v$, and so, $\overline{\mathbf{b}}$ also

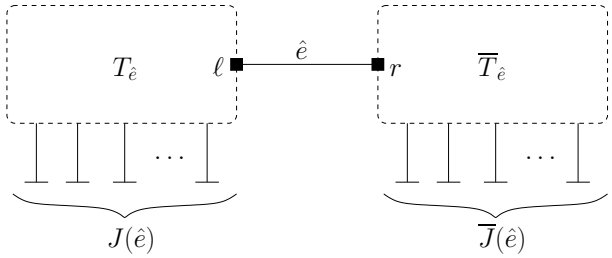

FIGURE 10. A depiction of the two subtrees $T_{\hat{e}}$ and $\overline{T}_{\hat{e}}$ connected by the edge $\hat{e}$.

satisfies the local constraint at v. Thus, $\overline{\mathbf{b}}$ satisfies all local constraints of Γ, so that $\overline{\mathbf{b}} \in \mathfrak{B}$. Hence, $(\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_\delta) = \overline{\mathbf{b}}|_v$ is in $\mathfrak{B}|_v$, which proves the lemma. □

APPENDIX B. PROOFS OF LEMMAS 3.1 AND 3.3

Proof of Lemma 3.1. For simplicity of notation, let F denote the edge set of the subtree $T_{\hat{e}}$, and let $\overline{F}$ denote that of the subtree $\overline{T}_{\hat{e}}$. Note that $F \cup \overline{F} = E(T) - \hat{e}$. Throughout this proof, we will write an arbitrary global configuration $\mathbf{b}$, belonging to $\mathfrak{B}$ or $\mathfrak{B}(\overline{\Gamma})$, in the form $(\mathbf{b}|_J, \mathbf{b}|_F, \mathbf{b}|_{\hat{e}}, \mathbf{b}|_{\overline{F}}, \mathbf{b}|_{\overline{J}})$.

Consider any $\overline{\mathbf{b}} = (\overline{\mathbf{b}}|_J, \overline{\mathbf{b}}|_F, \overline{\mathbf{b}}|_{\hat{e}}, \overline{\mathbf{b}}|_{\overline{F}}, \overline{\mathbf{b}}|_{\overline{J}}) \in \mathfrak{B}(\overline{\Gamma})$. Let ℓ and r be the two vertices incident with the edge $\hat{e}$ in T. We assume that $\ell \in V(T_{\hat{e}})$ and $r \in V(\overline{T}_{\hat{e}})$, as depicted in Figure 10. We write the local configuration $\overline{\mathbf{b}}|_\ell$ as $(\overline{\mathbf{b}}|_{E(\ell)-\hat{e}}, \overline{\mathbf{b}}|_{\omega^{-1}(\ell)}, \overline{\mathbf{b}}|_{\hat{e}})$, and $\overline{\mathbf{b}}|_r$ as $(\overline{\mathbf{b}}|_{\hat{e}}, \overline{\mathbf{b}}|_{\omega^{-1}(r)}, \overline{\mathbf{b}}|_{E(r)-\hat{e}})$.

Suppose first that $\overline{\mathbf{b}}|_{\hat{e}} = \mathbf{0}$; note that the zero element of $\overline{\mathcal{S}}_{\hat{e}}$ $(= \mathcal{S}_{\hat{e}}/W)$ is W. By definition of $\overline{\Gamma}$, $\overline{\mathbf{b}}|_\ell \in \overline{\mathfrak{B}}|_\ell = \Phi(\mathfrak{B})|_\ell$. Hence, there exists $(\overline{\mathbf{b}}|_{E(\ell)-\hat{e}}, \overline{\mathbf{b}}|_{\omega^{-1}(\ell)}, \mathbf{w}) \in \mathfrak{B}|_\ell$, for some $\mathbf{w} \in W$. Now, $(\overline{\mathbf{b}}|_J, \overline{\mathbf{b}}|_F)$ (being a "sub-configuration" of $\overline{\mathbf{b}}$) satisfies the local constraints of $\overline{\Gamma}$ at all vertices in $V(T_{\hat{e}}) - \{\ell\}$. But these local constraints are of the form $\overline{\mathfrak{B}}|_v$ which, for $v \in V(T_{\hat{e}}) - \{\ell\}$, is identical to $\mathfrak{B}_v$. Therefore, the sub-configuration $(\overline{\mathbf{b}}|_J, \overline{\mathbf{b}}|_F)$ satisfies the local constraints of Γ at all vertices in $V(T_{\hat{e}}) - \{\ell\}$. It follows that $(\overline{\mathbf{b}}|_J, \overline{\mathbf{b}}|_F, \mathbf{w})$ satisfies the local constraints of Γ at all vertices in $V(T_{\hat{e}})$, including ℓ. By a similar argument, there exists a $\mathbf{w}' \in W$ such that $(\mathbf{w}', \overline{\mathbf{b}}|_{\overline{F}}, \overline{\mathbf{b}}|_{\overline{J}})$ satisfies the local constraints of Γ at all vertices in $V(\overline{T}_{\hat{e}})$.

Now, by definition of W, there exist $\mathbf{b}$ and $\mathbf{b}'$ in $\mathfrak{B}$, such that $\mathbf{b} = (\mathbf{b}|_J, \mathbf{b}|_F, \mathbf{w}, \mathbf{b}|_{\overline{F}}, \mathbf{b}|_{\overline{J}})$, $\mathbf{b}' = (\mathbf{b}'|_J, \mathbf{b}'|_F, \mathbf{w}', \mathbf{b}'|_{\overline{F}}, \mathbf{b}'|_{\overline{J}})$, and $(\mathbf{b}|_J, \mathbf{b}|_{\overline{J}}), (\mathbf{b}'|_J, \mathbf{b}'|_{\overline{J}}) \in \mathcal{C}_J \oplus \mathcal{C}_{\overline{J}}$. Note, in particular, that the sub-configuration $(\mathbf{w}, \mathbf{b}|_{\overline{F}}, \mathbf{b}|_{\overline{J}})$ of $\mathbf{b}$ satisfies the local constraints of Γ at all vertices in $V(\overline{T}_{\hat{e}})$. Therefore, the global configuration $\mathbf{g} = (\overline{\mathbf{b}}|_J, \overline{\mathbf{b}}|_F, \mathbf{w}, \mathbf{b}|_{\overline{F}}, \mathbf{b}|_{\overline{J}})$ satisfies the local constraints of Γ at all vertices in T, and hence is in the full behavior, $\mathfrak{B}$, of Γ. A similar argument shows that $\mathbf{g}' = (\mathbf{b}'|_J, \mathbf{b}'|_F, \mathbf{w}', \overline{\mathbf{b}}|_{\overline{F}}, \overline{\mathbf{b}}|_{\overline{J}})$ is also in $\mathfrak{B}$.

As $\mathfrak{B}$ is a vector space, it must also contain

$$\mathbf{b} - \mathbf{g} = (\mathbf{b}|_J - \overline{\mathbf{b}}|_J, \mathbf{b}|_F - \overline{\mathbf{b}}|_F, \mathbf{0}, \mathbf{0}, \mathbf{0})$$

and

$$\mathbf{b}' - \mathbf{g}' = (\mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{b}'|_{\overline{F}} - \overline{\mathbf{b}}|_{\overline{F}}, \mathbf{b}'|_{\overline{J}} - \overline{\mathbf{b}}|_{\overline{J}}).$$

Since Γ is a tree realization of $\mathcal{C}$, we have $\mathfrak{B}|_I = \mathcal{C}$. In particular, $(\mathbf{b}|_J - \overline{\mathbf{b}}|_J, \mathbf{0}) = (\mathbf{b} - \mathbf{g})|_I \in \mathcal{C}$, and similarly, $(\mathbf{0}, \mathbf{b}'|_{\overline{J}} - \overline{\mathbf{b}}|_{\overline{J}}) = (\mathbf{b}' - \mathbf{g}')|_I \in \mathcal{C}$. Hence, $\mathbf{b}|_J - \overline{\mathbf{b}}|_J \in \mathcal{C}_J$ and $\mathbf{b}'|_{\overline{J}} - \overline{\mathbf{b}}|_{\overline{J}} \in \mathcal{C}_{\overline{J}}$. However, $\mathbf{b}$ and $\mathbf{b}'$ were chosen so that $\mathbf{b}|_J \in \mathcal{C}_J$ and $\mathbf{b}'|_{\overline{J}} \in \mathcal{C}_{\overline{J}}$. Thus, we also have $\overline{\mathbf{b}}|_J \in \mathcal{C}_J$ and $\overline{\mathbf{b}}|_{\overline{J}} \in \mathcal{C}_{\overline{J}}$. This finally yields $\overline{\mathbf{b}}|_I = (\overline{\mathbf{b}}|_J, \overline{\mathbf{b}}|_{\overline{J}}) \in \mathcal{C}_J \oplus \mathcal{C}_{\overline{J}}$, thus proving one direction of part (b) of the lemma.

We will next show that if $\overline{\mathbf{b}}|_{\hat{e}} \neq \mathbf{0}$, then $\overline{\mathbf{b}}_I \in \mathcal{C}$ but $\overline{\mathbf{b}}_I \notin \mathcal{C}_J \oplus \mathcal{C}_{\overline{J}}$. This will prove both part (a) and the reverse direction of part (b).

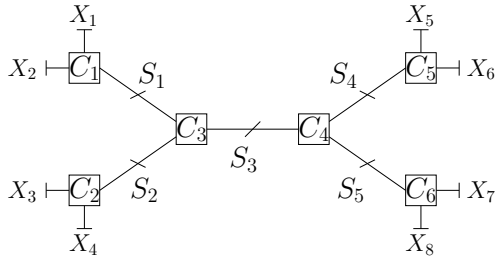

FIGURE 11. Template for tree realizations of the $[8, 4]$ Reed-Muller code.

So, suppose that $\overline{\mathbf{b}}|_{\hat{e}} = \bar{\mathbf{s}} \neq \mathbf{0}$. Thus, $\bar{\mathbf{s}}$ is some coset of W in $\mathcal{S}_{\hat{e}}$, but is not W itself. Pick some $\mathbf{s} \in \bar{\mathbf{s}}$. As $\mathcal{S}_{\hat{e}} = \mathfrak{B}|_{\hat{e}}$, there exists some $\mathbf{b} \in \mathfrak{B}$ such that $\mathbf{b}|_{\hat{e}} = \mathbf{s}$. Observe that $\mathbf{b}|_I \in \mathfrak{B}|_I = \mathcal{C}$, but since $\mathbf{s} \in \bar{\mathbf{s}} \neq W$, $\mathbf{b}|_I \notin \mathcal{C}_J \oplus \mathcal{C}_{\overline{J}}$.

Define $\widetilde{\mathbf{b}} = \Phi(\mathbf{b})$, so that $\widetilde{\mathbf{b}} \in \overline{\mathfrak{B}}$. Furthermore, $\widetilde{\mathbf{b}}|_{\hat{e}} = \bar{\mathbf{s}}$, and $\widetilde{\mathbf{b}}|_I$ $(= \mathbf{b}|_I)$ is in $\mathcal{C}$ but not in $\mathcal{C}_J \oplus \mathcal{C}_{\overline{J}}$. We have already noted (prior to the statement of Lemma 3.1) that $\overline{\mathfrak{B}} \subseteq \mathfrak{B}(\overline{\Gamma})$. Therefore, $\widetilde{\mathbf{b}} \in \mathfrak{B}(\overline{\Gamma})$, and since $\mathfrak{B}(\overline{\Gamma})$ is a vector space, $\overline{\mathbf{b}} - \widetilde{\mathbf{b}} \in \mathfrak{B}(\overline{\Gamma})$.

However, $(\overline{\mathbf{b}} - \widetilde{\mathbf{b}})|_{\hat{e}} = \bar{\mathbf{s}} - \bar{\mathbf{s}} = \mathbf{0}$, and as we showed above, this implies that $(\overline{\mathbf{b}} - \widetilde{\mathbf{b}})|_I \in \mathcal{C}_J \oplus \mathcal{C}_{\overline{J}}$. Since $\widetilde{\mathbf{b}}|_I$ is in $\mathcal{C}$ but not in $\mathcal{C}_J \oplus \mathcal{C}_{\overline{J}}$, we find that $\overline{\mathbf{b}}|_I \in \mathcal{C}$, but $\overline{\mathbf{b}}|_I \notin \mathcal{C}_J \oplus \mathcal{C}_{\overline{J}}$.

The proof of the lemma is now complete. □

Proof of Lemma 3.3. As $\overline{\Gamma}$ is a tree realization of $\mathcal{C}$, Lemma 2.1 shows that for any $\overline{\mathbf{b}} \in \mathfrak{B}(\overline{\Gamma})$, we have $\overline{\mathbf{b}}|_{e'} = \mathbf{0}$ only if $\overline{\mathbf{b}}|_I \in \mathcal{C}_{J(e')} \oplus \mathcal{C}_{\overline{J}(e')}$. Thus, we need only prove the converse.

Suppose that $\overline{\mathbf{b}} \in \mathfrak{B}(\overline{\Gamma})$ is such that $(\overline{\mathbf{b}}|_{J(e')}, \overline{\mathbf{b}}|_{\overline{J}(e')}) \in \mathcal{C}_{J(e')} \oplus \mathcal{C}_{\overline{J}(e')}$, but $\overline{\mathbf{b}}|_{e'} \neq \mathbf{0}$. Now, $\overline{\mathbf{b}}|_{e'} \in \overline{\mathcal{S}}_{e'} = \overline{\mathfrak{B}}|_{e'} = \mathfrak{B}|_{e'}$, the last equality being a consequence of the fact that $e' \neq \hat{e}$. Therefore, there exists a $\mathbf{b} \in \mathfrak{B}$ such that $\mathbf{b}|_{e'} = \overline{\mathbf{b}}|_{e'}$. Note that, by the hypothesis of the lemma, $\mathbf{b}|_I \notin \mathcal{C}_{J(e')} \oplus \mathcal{C}_{\overline{J}(e')}$.

Set $\widetilde{\mathbf{b}} = \Phi(\mathbf{b})$, so that $\widetilde{\mathbf{b}} \in \overline{\mathfrak{B}} \subseteq \mathfrak{B}(\overline{\Gamma})$. Observe that $\widetilde{\mathbf{b}}|_I = \mathbf{b}|_I$, and since $e' \neq \hat{e}$, we also have $\widetilde{\mathbf{b}}|_{e'} = \mathbf{b}|_{e'}$. Thus, $\widetilde{\mathbf{b}}|_{e'} = \overline{\mathbf{b}}|_{e'}$ and $\widetilde{\mathbf{b}}|_I \notin \mathcal{C}_{J(e')} \oplus \mathcal{C}_{\overline{J}(e')}$. But now, we have $\overline{\mathbf{b}} - \widetilde{\mathbf{b}} \in \mathfrak{B}(\overline{\Gamma})$, with $(\overline{\mathbf{b}} - \widetilde{\mathbf{b}})|_{e'} = \mathbf{0}$, and $(\overline{\mathbf{b}} - \widetilde{\mathbf{b}})|_{e'} \notin \mathcal{C}_{J(e')} \oplus \mathcal{C}_{\overline{J}(e')}$. This contradiction of Lemma 2.1 proves that there exists no $\overline{\mathbf{b}} \in \mathfrak{B}(\overline{\Gamma})$ such that $(\overline{\mathbf{b}}|_{J(e')}, \overline{\mathbf{b}}|_{\overline{J}(e')}) \in \mathcal{C}_{J(e')} \oplus \mathcal{C}_{\overline{J}(e')}$, but $\overline{\mathbf{b}}|_{e'} \neq \mathbf{0}$. □

APPENDIX C. EXAMPLE OF STATE-MERGING PROCEDURE

In this appendix. we illustrate the state-merging process described in Section 3 by using it to derive a minimal tree realization of the binary $[8, 4]$ Reed-Muller code $\mathcal{C}$ generated by

$$G_{RM} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}. \tag{31}$$

The index set of the code is assumed to be $I = \{1, 2, \ldots, 8\}$, where index i represents the ith coordinate of the code. The template for our tree realizations — consisting of a tree decomposition of I, symbol variables X_i $(i \in I)$, state variables S_e $(e \in \{1, 2, 3, 4, 5\})$, and local constraint codes $\mathcal{C}_v$ $(v \in \{1, 2, 3, 4, 5, 6\})$ — is depicted in Figure 11. We will describe our tree realizations by specifying the state spaces $\mathcal{S}_e$ and the local constraint codes C_v. All vector spaces and codes defined are over the binary field, $\mathbb{F}_2$. This appendix is simply meant to be an aid to helping the reader understand the state-merging process, so the details will be kept to a bare minimum.

We start with a trivial extension (cf. Example 2.1) of the depicted tree decomposition. This yields a tree realization Γ_{triv} of the Reed-Muller code $\mathcal{C}$, specified as follows. For the state spaces, we have $\mathcal{S}_1 = \mathcal{S}_2 = \mathcal{S}_4 = \mathcal{S}_5 = \mathbb{F}_2^2$, and $\mathcal{S}_3 = \mathbb{F}_2^4$. The local constraints C_1, C_2, C_5 and C_6 are $[4, 2]$ codes with basis vectors 1010 and 0101. The first two coordinates of these $[4, 2]$ codes correspond to the appropriate symbol variable pair (X_i, X_{i+1}) and the last two coordinates correspond to the appropriate state variable S_j. The code C_3 is the $[8, 4]$ Reed-Muller code generated by the matrix G_{RM} in (31), with the symbol variables S_1, S_2 and S_3 corresponding to the first two, next two, and last four coordinates, respectively, of C_3. Finally, the code C_4 is taken to be the $[8, 4]$ code generated by

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

with the symbol variables S_3, S_4 and S_5 corresponding to the first four, next two, and last two coordinates, respectively, of C_4.

The full behavior $\mathfrak{B}$ of Γ_{triv} is generated by the following four global configuration vectors:

X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	S_1	S_2	S_3	S_4	S_5
1	1	0	0	1	1	0	0	11	00	1100	11	00
0	1	1	0	0	1	1	0	01	10	0110	01	10
0	0	1	1	0	0	1	1	00	11	0011	00	11
0	0	1	1	1	1	0	0	00	11	1100	11	00

(32)

It is easy to see that this realization is not essential — for example, if we take e to be the central edge of the tree (associated with state variable S_3), then $\mathfrak{B}|_e$ is the 3-dimensional subspace of $\mathcal{S}_e = \mathbb{F}_2^4$ generated by 1100, 0110 and 0011.

The essentialization of Γ_{triv} is the tree realization Γ_0 specified as follows. The state spaces of Γ_0 are the same as those in Γ_{triv}, except that $\mathcal{S}_3$ is now the 3-dimensional subspace of $\mathbb{F}_2^4$ generated by 1100, 0110 and 0011. The local constraints of Γ_0 are the same as those in Γ_{triv}, except for C_4 which is now the $[8, 3]$ code generated by the matrix

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

The full behavior $\mathfrak{B}$ of Γ_0 is exactly the same as that of Γ_{triv}, which is shown in (32).

We will apply the state-merging process at the central edge of the tree in Figure 11. We will now denote this edge by $\hat{e}$, in keeping with the description in Section 3. Note that $\mathcal{S}_{\hat{e}} = \mathcal{S}_3$ in our notation above. Our first task is to determine the subspace, W, of $\mathcal{S}_{\hat{e}}$, as defined in (7). It is easily checked that $\mathcal{C}_J \oplus \mathcal{C}_{\overline{J}}$ is the subcode of $\mathcal{C}$ generated by 11110000 and 00001111, and as a consequence, we obtain $W = \{0000, 1111\}$. We identify the 2-dimensional vector space $\mathcal{S}_{\hat{e}}/W$ with $\mathbb{F}_2^2$ as follows: the coset $0011 + W$ is identified with 01, and the coset $0110 + W$ is identified with 10. With this identification, the mapping Φ defined in (8), when applied to $\mathfrak{B}$, yields the vector space $\overline{\mathfrak{B}} = \Phi(\mathfrak{B})$ generated by the following four global configuration vectors:

X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	S_1	S_2	S_3	S_4	S_5
1	1	0	0	1	1	0	0	11	00	01	11	00
0	1	1	0	0	1	1	0	01	10	10	01	10
0	0	1	1	0	0	1	1	00	11	01	00	11
0	0	1	1	1	1	0	0	00	11	01	11	00

From $\overline{\mathfrak{B}}$, we derive the tree realization $\overline{\Gamma}$ with state spaces $\overline{\mathcal{S}}_e = \overline{\mathfrak{B}}|_e$ for each edge e, and local constraints $\overline{C}_v = \overline{\mathfrak{B}}|_v$ for each vertex v. It is readily verified that $\overline{\mathcal{S}}_e = \mathcal{S}_e = \mathbb{F}_2^2$ for $e = 1, 2, 4, 5$, and $\overline{\mathcal{S}}_3 = \mathbb{F}_2^2$ as well. The local constraints $\overline{C}_v$ are the same $[4, 2]$ codes as C_v for $v = 1, 2, 5, 6$;

however, the codes $\overline{C}_v$ at the two remaining vertices are $[6,3]$ codes. The basis vectors for the codes $\overline{C}_3$ and $\overline{C}_4$ are given below:

$$\overline{C}_3: \begin{array}{ccc} S_1 & S_2 & S_3 \\ 11 & 00 & 01 \\ 01 & 10 & 10 \\ 00 & 11 & 01 \end{array} \qquad \overline{C}_4: \begin{array}{ccc} S_3 & S_4 & S_5 \\ 01 & 11 & 00 \\ 10 & 01 & 10 \\ 01 & 00 & 11 \end{array}$$

One can now check that further state merging does not change the realization $\overline{\Gamma}$, which means that $\overline{\Gamma}$ is a minimal tree realization of the Reed-Muller code $\mathcal{C}$. In fact, this is the realization depicted in [7, Figure 31].

Appendix D. Proof of Forward Direction of Theorem 4.3

Proof of (a) $\Rightarrow$ (b) *in Theorem 4.3.* Let $\mathcal{C} = \mathcal{C}_1 \oplus_r \mathcal{C}_2$ for codes $\mathcal{C}_1$ and $\mathcal{C}_2$ defined on the index sets I_1 and I_2, respectively. By definition, $I = I_1 \Delta I_2$. Set $J = I_1 - I_2$ and $\overline{J} = I_2 - I_1$, so that $(J, \overline{J})$ forms a partition of I. In what follows, words defined on the index set I_1 will be written in the form $\mathbf{x} = (\mathbf{x}|_J, \mathbf{x}|_{I_1 \cap I_2})$; words defined on the index set I_2 will be written in the form $\mathbf{x} = (\mathbf{x}|_{I_1 \cap I_2}, \mathbf{x}|_{\overline{J}})$; words defined on the index set I will be written as $\mathbf{x} = (\mathbf{x}|_J, \mathbf{x}|_{\overline{J}})$; and finally, words on the index set $I_1 \cup I_2$ will be written as $\mathbf{x} = (\mathbf{x}|_J, \mathbf{x}|_{I_1 \cap I_2}, \mathbf{x}|_{\overline{J}})$.

We begin by proving that $\dim(\mathcal{C}|_J) = \dim(\mathcal{C}_1)$. This is accomplished by a two-step argument: we first show that $\mathcal{C}|_J = \mathcal{C}_1|_J$, and then we show that $\mathcal{C}_1|_J \cong \mathcal{C}_1$.

If $\mathbf{a} \in \mathcal{C}|_J$, then there exists some $\mathbf{x} \in \mathcal{C}_1$, $\mathbf{y} \in \mathcal{C}_2$ such that $(\mathbf{x} \star \mathbf{y})|_J = \mathbf{a}$. However, $(\mathbf{x} \star \mathbf{y})|_J = \mathbf{x}|_J$ as J lies outside $I_1 \cap I_2$. Hence, $\mathbf{a} = \mathbf{x}|_J \in \mathcal{C}_1|_J$. Conversely, suppose that $\mathbf{a} \in \mathcal{C}_1|_J$. Then, there exists $\mathbf{z} \in \mathcal{C}_1^{(p)}$ such that $\mathbf{x} = (\mathbf{a}, \mathbf{z}) \in \mathcal{C}_1$. Since $\mathcal{C}_1^{(p)} = \mathcal{C}_2^{(p)}$, there exists $\mathbf{y} = (\mathbf{z}, \mathbf{b}) \in \mathcal{C}_2$. Now, $\mathbf{x} \star \mathbf{y} = (\mathbf{a}, \mathbf{0}, \mathbf{b})$, and hence $(\mathbf{a}, \mathbf{b}) \in \mathcal{C}$. Thus, $\mathbf{a} \in \mathcal{C}|_J$, which completes the proof of the fact that $\mathcal{C}|_J = \mathcal{C}_1|_J$.

Now, to show that $\mathcal{C}_1 \cong \mathcal{C}_1|_J$, let us consider the projection map $\pi : \mathcal{C}_1 \to \mathcal{C}_1|_J$ defined by $\pi(\mathbf{x}) = \mathbf{x}|_J$. This map is a homomorphism, with kernel isomorphic to $\mathcal{C}_1^{(s)}$, which is $\{\mathbf{0}\}$ by definition. Hence, π is in fact an isomorphism, which proves that $\mathcal{C}_1 \cong \mathcal{C}_1|_J$.

We have thus shown that $\dim(\mathcal{C}|_J) = \dim(\mathcal{C}_1)$. A similar argument yields the fact that $\dim(\mathcal{C}|_{\overline{J}}) = \dim(\mathcal{C}_2)$. Hence,

$$\dim(\mathcal{C}|_J) + \dim(\mathcal{C}|_{\overline{J}}) - \dim(\mathcal{C}) = \dim(\mathcal{C}_1) + \dim(\mathcal{C}_2) - \dim(\mathcal{C}_1 \oplus_r \mathcal{C}_2) = r,$$

by Corollary 4.2.

It remains to show that $\min\{|J|, |\overline{J}|\} \geq r$. Note that since $\dim(\mathcal{C}|_J) + \dim(\mathcal{C}|_{\overline{J}}) - \dim(\mathcal{C}) = r$, and $\dim(\mathcal{C}|_{\overline{J}}) \leq \dim(\mathcal{C})$, we must have $\dim(\mathcal{C}|_J) \geq r$. Therefore, $|J| \geq \dim(\mathcal{C}|_J) \geq r$. By a similar argument, we also have $|\overline{J}| \geq r$. □

Appendix E. Proof of Proposition 5.1

The proof of Proposition 5.1 requires the following lemma, which presents a property of the codes $\mathcal{C}_1$ and $\mathcal{C}_2$ obtained via the r-sum decomposition procedure of Section 4.

Lemma E.1. *Let $\mathcal{C}$ be a code defined on the index set I, and let $(J, \overline{J})$ be a partition of I, with $\dim(\mathcal{C}|_J) + \dim(\mathcal{C}|_{\overline{J}}) - \dim(\mathcal{C}) = r$. Suppose that $\mathcal{C}_1$ and $\mathcal{C}_2$ are the codes, defined on the respective index sets I_1 and I_2, that are obtained by the procedure described in the proof of Theorem 4.3. Then, for any $J_1 \subseteq J$, and any $J_2 \subseteq \overline{J}$, we have*

$$\dim(\mathcal{C}_1|_{J_1}) + \dim(\mathcal{C}_1|_{I_1 - J_1}) - \dim(\mathcal{C}_1) = \dim(\mathcal{C}|_{J_1}) + \dim(\mathcal{C}|_{I - J_1}) - \dim(\mathcal{C}), \quad (33)$$

$$\dim(\mathcal{C}_2|_{J_2}) + \dim(\mathcal{C}_2|_{I_2 - J_2}) - \dim(\mathcal{C}_2) = \dim(\mathcal{C}|_{J_2}) + \dim(\mathcal{C}|_{I - J_2}) - \dim(\mathcal{C}). \quad (34)$$

Proof. We use notation from the proof of the (b) $\Rightarrow$ (a) direction of Theorem 4.3. Thus, $\mathcal{C}$, $\mathcal{C}_1$ and $\mathcal{C}_2$ are generated by the matrices $\overline{G}$, G_1 and G_2 given by (11), (12) and (13), respectively, which we reproduce here for the sake of convenience.

$$\begin{aligned}\overline{G} &= \begin{bmatrix} I_{k_1} & A & \mathbf{O} & B \\ \mathbf{O} & \mathbf{O} & I_{k-k_1} & C \end{bmatrix},\\ G_1 &= \begin{bmatrix} I_{k_1} & A & X \end{bmatrix},\\ G_2 &= \begin{bmatrix} X & \mathbf{O} & B \\ \mathbf{O} & I_{k-k_1} & C \end{bmatrix}.\end{aligned}$$

For any matrix M, given a subset Z of the column indices of M, we will denote by $M|_Z$ the restriction of M to the columns indexed by Z. Thus,

$$\overline{G}|_J = \begin{bmatrix} I_{k_1} & A \\ \mathbf{O} & \mathbf{O} \end{bmatrix}, \quad G_1|_J = \begin{bmatrix} I_{k_1} & A \end{bmatrix},$$

$$\overline{G}|_{\overline{J}} = G_2|_{\overline{J}} = \begin{bmatrix} \mathbf{O} & B \\ I_{k-k_1} & C \end{bmatrix}.$$

Our proof of the lemma uses only elementary linear algebra. We prove (33) first. Consider any $J_1 \subseteq J$. It is clear that $\overline{G}|_{J_1} = \begin{bmatrix} G_1|_{J_1} \\ \mathbf{O} \end{bmatrix}$, and therefore, we have $\dim(\mathcal{C}|_{J_1}) = \mathsf{rank}(\overline{G}|_{J_1}) = \mathsf{rank}(G_1|_{J_1}) = \dim(\mathcal{C}_1|_{J_1})$. Next, note that $I - J_1 = (J - J_1) \dot{\cup} \overline{J}$, from which we have

$$\dim(\mathcal{C}|_{I-J_1}) = \mathsf{rank}\left(\begin{bmatrix} \overline{G}|_{J-J_1} & \overline{G}|_{\overline{J}} \end{bmatrix}\right).$$

Now, observe that by performing column operations on $\overline{G}|_{\overline{J}}$, we can bring it into the form

$$W = \begin{bmatrix} \mathbf{O} & B \\ I_{k-k_1} & \mathbf{O} \end{bmatrix}.$$

Hence,

$$\begin{aligned}\mathsf{rank}\left(\begin{bmatrix} \overline{G}|_{J-J_1} & \overline{G}|_{\overline{J}} \end{bmatrix}\right) &= \mathsf{rank}\left(\begin{bmatrix} \overline{G}|_{J-J_1} & W \end{bmatrix}\right)\\ &= \mathsf{rank}(I_{k-k_1}) + \mathsf{rank}\left(\begin{bmatrix} G_1|_{J-J_1} & B \end{bmatrix}\right)\\ &= k - k_1 + \mathsf{rank}\left(\begin{bmatrix} G_1|_{J-J_1} & B \end{bmatrix}\right).\end{aligned}$$

At this point, we have

$$\dim(\mathcal{C}|_{J_1}) + \dim(\mathcal{C}|_{I-J_1}) = \dim(\mathcal{C}_1|_{J_1}) + k - k_1 + \mathsf{rank}\left(\begin{bmatrix} G_1|_{J-J_1} & B \end{bmatrix}\right),$$

which upon re-arrangement yields

$$\dim(\mathcal{C}|_{J_1}) + \dim(\mathcal{C}|_{I-J_1}) - \dim(\mathcal{C}) = \dim(\mathcal{C}_1|_{J_1}) + \mathsf{rank}\left(\begin{bmatrix} G_1|_{J-J_1} & B \end{bmatrix}\right) - \dim(\mathcal{C}_1).$$

Thus, (33) would be proved if we could establish that $\dim(\mathcal{C}_1|_{I_1-J_1}) = \mathsf{rank}\left(\begin{bmatrix} G_1|_{J-J_1} & B \end{bmatrix}\right)$.

Now, $I_1 - J_1 = (J - J_1) \dot{\cup} I_X$, and hence,

$$\dim(\mathcal{C}_1|_{I_1-J_1}) = \mathsf{rank}\left(\begin{bmatrix} G_1|_{J-J_1} & X \end{bmatrix}\right).$$

Thus, we have to show that $\mathsf{rank}\left(\begin{bmatrix} G_1|_{J-J_1} & B \end{bmatrix}\right) = \mathsf{rank}\left(\begin{bmatrix} G_1|_{J-J_1} & X \end{bmatrix}\right)$. We will prove that the matrices B and X have identical column-spaces. Clearly, the desired result then follows.

Recall that for $i = 1, 2, \ldots, k_1$, the ith row of B can be uniquely expressed as a linear combination, $\sum_{j=1}^{r} \alpha_{i,j}\mathbf{b}_j$, of its first r rows $\mathbf{b}_1, \ldots, \mathbf{b}_r$. Furthermore, the ith row of X equals $\sum_{j=1}^{r} \alpha_{i,j}\mathbf{d}_j$ for the same $\alpha_{i,j}$'s, where $\mathbf{d}_1, \ldots, \mathbf{d}_r$ are the rows of the generator matrix, D_r, of the code Δ_r. In particular, the first r rows of X constitute the matrix D_r. Denote by B_r the submatrix of B comprised by its first r rows.

At this point, we must use a fundamental property of the matrix D_r, namely, that the columns of D_r form a maximal subset of $\mathbb{F}_q^r$ with the property that each pair of vectors from the subset is

linearly independent over $\mathbb{F}_q$. This is due to the fact that, by definition, each pair of columns of D_r is linearly independent over $\mathbb{F}_q$, and furthermore, the number of distinct one-dimensional subspaces of $\mathbb{F}_q^r$ is precisely equal to the number, $m_r = (q^r - 1)/(q - 1)$, of columns of D_r. Therefore, any (column) vector in $\mathbb{F}_q^r$ is a scalar multiple of some column of D_r.

In particular, any column of B_r is a scalar multiple of some column of D_r. But because of the way X was constructed, this implies that any column of B is a scalar multiple of some column of X. Thus, the column-space of B is a subspace of the column-space of X. However, we also have $\mathsf{rank}(B) = \mathsf{rank}(X)$, and so, the column-spaces of the two matrices are in fact identical. This proves that $\mathsf{rank}\left(\begin{bmatrix} G_1|_{J-J_1} & B \end{bmatrix}\right) = \mathsf{rank}\left(\begin{bmatrix} G_1|_{J-J_1} & X \end{bmatrix}\right)$, and (33) follows.

To show (34), consider any $J_2 \subseteq \overline{J}$. Arguments similar to the ones above establish that

$$\dim(\mathcal{C}|_{J_2}) = \dim(\mathcal{C}_2|_{J_2}) \quad \text{and} \quad \dim(\mathcal{C}|_{I-J_2}) = k_1 + \mathsf{rank}\left([I_{k-k_1} \quad C]\,\big|_{\overline{J}-J_2}\right). \tag{35}$$

Now, consider $\dim(\mathcal{C}_2|_{I_2-J_2}) = \mathsf{rank}(G_2|_{I_2-J_2})$. Noting that $I_2 - J_2 = I_X \,\dot\cup\, (\overline{J} - J_2)$, we see that the matrix $G_2|_{I_2-J_2}$ has the form

$$\begin{bmatrix} X & \mathbf{O} & B|_K \\ \mathbf{O} & I' & C|_K \end{bmatrix},$$

with $I' = (I_{k-k_1})|_{\overline{J}-J_2-K}$, for some $K \subseteq \overline{J} - J_2$. Since the columns of B are contained in the column-space of X, we can perform column operations on $G_2|_{I_2-J_2}$ to bring it into the form

$$W' = \begin{bmatrix} X & \mathbf{O} & \mathbf{O} \\ \mathbf{O} & I' & C|_K \end{bmatrix}.$$

Hence,

$$\begin{aligned} \mathsf{rank}(G_2|_{I_2-J_2}) \;=\; \mathsf{rank}(W') \;&=\; \mathsf{rank}(X) + \mathsf{rank}([I' \quad C|_K]) \\ &=\; (k_1 + k_2 - k) + \mathsf{rank}\left([I_{k-k_1} \quad C]\,\big|_{\overline{J}-J_2}\right). \end{aligned} \tag{36}$$

Some trivial manipulations of (35) and (36) yield (34), which proves the lemma. □

Proof of Proposition 5.1. Recall that $\Gamma^* = (T, \omega, (\mathcal{S}_e,\, e \in E(T)), (C_v,\, v \in V(T)))$, where $\mathcal{S}_e$ and C_v are as defined in (19) and (20). To show that Γ^* is the minimal realization $\mathcal{M}(\mathcal{C}; T, \omega)$, it is enough to show that for all $e \in E(T)$, $\dim(\mathcal{S}_e)$ equals the expression in (5), *i.e.*,

$$\dim(\mathcal{S}_e) = \dim(\mathcal{C}|_{J(e)}) + \dim(\mathcal{C}|_{\overline{J}(e)}) - \dim(\mathcal{C}). \tag{37}$$

Note that this is true when $e = \hat{e}$, since $\dim(\mathcal{S}_{\hat{e}}) = \dim(\Delta_r) = r$, and from (14), we have $r = \dim(\mathcal{C}|_{J(\hat{e})}) + \dim(\mathcal{C}|_{\overline{J}(\hat{e})}) - \dim(\mathcal{C})$. We must therefore show that (37) holds for $e \in E(T) - \{\hat{e}\} = E(T_{\hat{e}}) \cup E(\overline{T}_{\hat{e}})$. We will prove this for $e \in E(T_{\hat{e}})$; the proof for $e \in E(\overline{T}_{\hat{e}})$ is similar.

So, consider any $e \in E(T_{\hat{e}})$. One of the two components, T_e and $\overline{T}_e$, of $T - e$ is contained in $T_{\hat{e}}$. Without loss of generality, we may assume that it is T_e that is a subtree of $T_{\hat{e}}$, as depicted in Figure 12. Hence, $J(e) = \omega^{-1}(V(T_e)) \subseteq J(\hat{e})$. Now, by (19), $\mathcal{S}_e = \mathcal{S}_e^{(1)}$, the latter being the state space associated with e in $\mathcal{M}(\mathcal{C}_1; T_{\hat{e}}, \omega_1)$. Therefore, by (5),

$$\dim(\mathcal{S}_e^{(1)}) = \dim(\mathcal{C}_1|_{J(e)}) + \dim(\mathcal{C}_1|_{I_1-J(e)}) - \dim(\mathcal{C}_1).$$

But, by Lemma E.1, the above expression is equal to the expression on the right-hand side of (37). Hence, (37) holds for any $e \in E(T_{\hat{e}})$, and the proposition follows. □

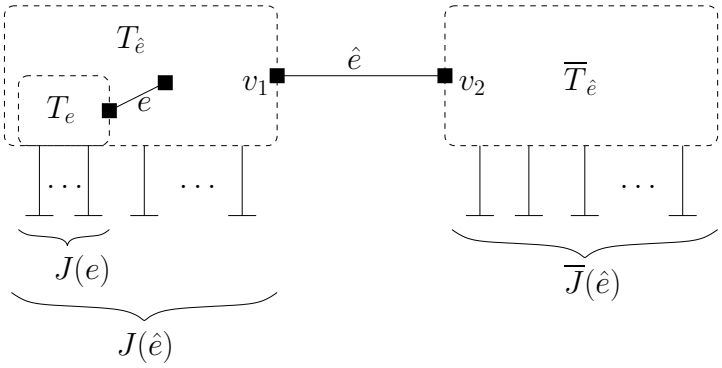

FIGURE 12. T_e is a subtree of $T_{\hat{e}}$.

REFERENCES

[1] S.M. Aji and R.J. McEliece, "The generalized distributive law," *IEEE Trans. Inform. Theory*, vol. 46, no. 2, pp. 325–343, 2000.

[2] S. Arnborg, D.G. Corneil and A. Proskurowski, "Complexity of finding embeddings in a k-tree," *SIAM J. Alg. Disc. Meth.*, vol. 8, pp. 277–284, 1987.

[3] S. Arnborg and A. Proskurowski, "Linear time algorithms for NP-hard problems restricted to partial k-trees," *Discrete Applied Mathematics*, vol. 23, no. 1, pp. 11–24, 1989.

[4] E.R. Berlekamp, R.J. McEliece, and H.C.A. van Tilborg, "On the inherent intractability of certain coding problems," *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 384–386, 1978.

[5] H.L. Bodlaender, "Dynamic programming on graphs of bounded treewidth," *Proc. 15th International Colloquium on Automata, Languages and Programming*, vol. 317, Lecture Notes in Computer Science, Springer-Verlag, pp. 105–118, 1988.

[6] H.L. Bodlaender, "A tourist guide through treewidth," *Acta Cybernetica*, vol. 11, pp. 1–23, 1993.

[7] G.D. Forney Jr., "Codes on graphs: normal realizations," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 520–548, Feb. 2001.

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

[9] T.R. Halford and K.M. Chugg, "The extraction and complexity limites of graphical models for linear codes," *IEEE Trans. Inform. Theory*, to appear.

[10] S.L. Hakimi and J.G. Bredeson, "Graph theoretic error-correcting codes," *IEEE Trans. Inform. Theory*, vol. IT-14, pp. 584–591, 1968.

[11] I.V. Hicks and N.B. McMurray, Jr., "The branch-width of graphs and their cycle matroids," *J. Combin. Theory, Ser. B*, vol. 97, pp. 681–692, 2007.

[12] P. Hliněný, "A parametrized algorithm for matroid branch-width," *SIAM J. Computing*, vol. 35, pp. 259–277, 2005.

[13] P. Hliněný, personal email communication, Sept. 2007.

[14] P. Hliněný, and S.-i. Oum, "Finding branch-decompositions and rank-decompositions," *SIAM J. Comput.*, vol. 38, no. 3, pp. 1012–1032, 2008.

[15] P. Hliněný, S.-i. Oum, D. Seese and G. Gottlob, "Width parameters beyond tree-width and their applications," *The Computer Journal*, vol. 51, no. 3, pp. 326–362, 2008.

[16] P. Hliněný and G. Whittle, "Matroid tree-width," *Europ. J. Combin.*, vol. 27, pp. 1117–1128, 2006.

[17] P. Hliněný and G. Whittle, "Addendum to matroid tree-width," to appear in *Europ. J. Combin.*. Preprint, 2008.

[18] N. Kashyap, "A decomposition theory for binary linear codes," *IEEE Trans. Inform. Theory*, vol. 54, no. 7, pp. 3035–3058, July 2008.

[19] N. Kashyap, "Matroid pathwidth and code trellis complexity," *SIAM J. Discrete Math.*, vol. 22, no. 1, pp. 256–272, 2008.

[20] N. Kashyap, "Constraint complexity of realizations of linear codes on arbitrary graphs," submitted to *IEEE Trans. Inform. Theory*, 2008. ArXiv e-print 0805.2199.

[21] F.R. Kschischang, B.J. Frey and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.

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

[23] H.-A. Loeliger, G.D. Forney Jr., T. Mittelholzer, and M.D. Trott, "Minimality and observability of group systems," *Linear Algebra & Appl.*, vol. 205–206, pp. 937–963, July 1994.
[24] H.-A. Loeliger and T. Mittelholzer, "Convolutional codes over groups," *IEEE Trans. Inform. Theory*, vol. 42, pp. 1660–1686, Nov. 1996.
[25] J.G. Oxley, *Matroid Theory*, Oxford University Press, Oxford, UK, 1992.
[26] N. Robertson and P.D. Seymour, "Graph minors. I. Excluding a forest," *J. Combin. Theory, Ser. B*, vol. 35, pp. 39–61, 1983.
[27] N. Robertson and P.D. Seymour, "Graph minors — a survey," in *Surveys in Combinatorics*, Cambridge University Press, 1985, pp. 153–171.
[28] P.D. Seymour, "Decomposition of regular matroids," *J. Combin. Theory, Series B*, vol. 28, pp. 305–359, 1980.
[29] K. Truemper, *Matroid Decomposition*, Academic Press, San Diego, 1992.
[30] J.H. van Lint, *Introduction to Coding Theory*, 3rd ed., Springer, Berlin, 1998.
[31] A. Vardy, "Trellis Structure of Codes," in *Handbook of Coding Theory*, R. Brualdi, C. Huffman and V. Pless, Eds., Amsterdam, The Netherlands: Elsevier, 1998.
[32] N. Wiberg, *Codes and Decoding on General Graphs*, Ph.D. thesis, Linköping University, Linköping, Sweden, 1996.
[33] N. Wiberg. H.-A. Loeliger and R. Koetter, "Codes and iterative decoding on general graphs," *Euro. Trans. Telecommun.*, vol. 6, pp. 513–525, Sept./Oct. 1995.