A DECOMPOSITION THEORY FOR BINARY LINEAR CODES

NAVIN KASHYAP

ABSTRACT. The decomposition theory of matroids initiated by Paul Seymour in the 1980's has had an enormous impact on research in matroid theory. This theory, when applied to matrices over the binary field, yields a powerful decomposition theory for binary linear codes. In this paper, we give an overview of this code decomposition theory, and discuss some of its implications in the context of the recently discovered formulation of maximum-likelihood (ML) decoding of a binary linear code over a binary-input discrete memoryless channel as a linear programming problem. We translate matroid-theoretic results of Grötschel and Truemper from the combinatorial optimization literature to give examples of non-trivial families of codes for which the ML decoding problem can be solved in time polynomial in the length of the code. One such family is that consisting of codes $\mathcal{C}$ for which the codeword polytope is identical to the Koetter-Vontobel fundamental polytope derived from the entire dual code $\mathcal{C}^\perp$. However, we also show that such families of codes are not good in a coding-theoretic sense — either their dimension or their minimum distance must grow sub-linearly with codelength. As a consequence, we have that decoding by linear programming, when applied to good codes, cannot avoid failing occasionally due to the presence of pseudocodewords.

1. INTRODUCTION

Historically, the theory of error-correcting codes has benefited greatly from the use of techniques from algebra and algebraic geometry. Combinatorial and graph-theoretic methods have also proven to be useful in the construction and analysis of codes, especially within the last ten years since the re-discovery of low-density parity-check codes. However, one area of mathematics that has, surprisingly, only played a relatively minor role in the development of coding theory is the field of matroid theory. The reason this is surprising is that, as anyone with even a basic understanding of the two fields would realize, coding theory and matroid theory are very closely related. The former deals with matrices over a finite field $\mathbb{F}$, and these objects are also of fundamental importance in the latter, where they go under the name of $\mathbb{F}$-representable matroids.

The connection with matroid theory has not gone unnoticed among coding theorists. Indeed, as far back as 1976, Greene [1] noticed that the Tutte polynomial of a matroid, when specialized to a linear code $\mathcal{C}$, was equivalent (in a certain sense) to the homogeneous weight-enumerator polynomial $W_\mathcal{C}(x, y) = \sum_i A_i x^i y^{n-i}$, where A_i is the number of words of weight i in $\mathcal{C}$, and n is the length of $\mathcal{C}$. The MacWilliams identities are then a special case of a known identity for the Tutte polynomial [2, Chapter 4]. The connection with matroids was also exploited by Barg [3] to derive MacWilliams-type identities for generalized-Hamming-weight enumerators of a code.

However, aside from such use of tools from matroid theory to re-derive results in coding theory that had already been proved by other means, each field seems to have had little impact on the other. Matroid theory has derived its inspiration largely from graph theory, and its most successful applications have traditionally been in the areas of combinatorial optimization and network flows. Very recently, matroid theory has found new applications in the rapidly evolving field of network coding [4].

Date: March 10, 2008.

This work was supported by a Discovery Grant from the Natural Sciences and Engineering Research Council (NSERC), Canada.

N. Kashyap is with the Department of Mathematics and Statistics, Queen's University, Kingston, ON K7L 3N6, Canada. Email: nkashyap@mast.queensu.ca.

On the other hand, coding theory (by which we mean the theory of error-correcting codes, in contrast to the theory of network coding) has been largely unconcerned with developments in combinatorial optimization, as the fundamental problems in the former seemed to be of a different nature from those in the latter. However, the recent re-formulation of the maximum-likelihood (ML) decoding problem, for a binary linear code over a binary-input discrete memoryless channel, as a linear programming problem [5] has opened a channel through which matroid-theoretic results in combinatorial optimization can be applied to coding theory. The key tool in these results is the use of the decomposition theory of matroids initiated by Seymour [6], [7]. Based on Seymour's seminal work, Grötschel and Truemper [8] showed that the minimization of a linear functional over the cycle polytope of a binary matroid could be solved in polynomial time for certain classes of matroids. This immediately implies that for the corresponding families of codes, the ML decoding problem can be solved in time polynomial in the length of the code. Given the fact that the ML decoding problem is known to be NP-hard in general [9], the existence of "non-trivial" classes of codes for which ML decoding can be implemented in polynomial time, is obviously a significant result. However, as we will show in this paper, for a code family to which the Grötschel-Truemper result applies, either the dimension or minimum distance of the codes in the family grows sub-linearly with codelength. Thus, such code families are not good from a coding-theoretic perspective. However, they do illustrate the important point that polynomial-time ML decoding is possible. Moreover, the matroid-theoretic arguments used by Grötschel and Truemper do not rule out the possibility that there may exist other code families for which polynomial-time ML decoding algorithms exist, which are also good in terms of rate and minimum distance.

The primary goal of this paper is to provide an exposition of the ideas needed to understand and apply the work of Grötschel and Truemper. As mentioned earlier, their work relies upon the machinery provided by Seymour's matroid decomposition theory, and so we will first present that theory in a coding-theoretic setting. Our presentation of this decomposition theory will be of a tutorial nature. We have attempted to keep the presentation self-contained to the extent possible; we do not provide complete proofs of some of the difficult theorems that form the basis of the theory.

We provide the relevant definitions and background from matroid theory in Section 2 of this paper. As explained in that section, binary matroids and binary linear codes are essentially the same objects. So, techniques applicable to binary matroids are directly applicable to binary linear codes as well. In particular, matroid decomposition techniques can be specialized to codes.

Of central importance in matroid theory is the notion of matroid minors. In the context of codes, a minor of a code $\mathcal{C}$ is any code that can be obtained from $\mathcal{C}$ by a sequence of shortening and puncturing operations. Minors have received little (if any) attention in coding theory, and this seems to be a remarkable oversight given the fact that they sometimes capture important structural properties of a code. For example, the presence or absence of certain minors (as stated precisely in Theorem 2.1) decides whether or not a code is graphic, *i.e.*, has a parity-check matrix that is the vertex-edge incidence matrix of some graph. Graphic codes have been studied previously in the information theory literature [10],[11], but the excluded-minor characterization of these codes appears to have been overlooked in these studies.

In Section 3, we introduce a notion of k-connectedness for codes, which is again a specialization of the corresponding notion for matroids. This is closely related to k-connectedness in graphs, and interestingly enough, is also related to the trellis complexity of a code [12]. We do not explore the latter relationship in any detail in this paper, instead referring the reader to [13], [14], where matroid methods are used to study the structure of codes with low trellis complexity.

The notion of k-connectedness plays an important role in Seymour's decomposition theory. An idea of why this is so can be obtained from the simple case of 2-connectedness: it follows from the relevant definitions that a code is not 2-connected if and only if it is the direct sum of smaller codes. Similar statements can be made for codes that are not 3- or 4-connected (or more precisely, not internally 4-connected — see Definition 4.4) via the code-composition operations of 2-sum and 3-sum introduced by Seymour [6]. These operations, as well as the $\overline{3}$-sum which is in a sense the dual operation to the 3-sum, are explained in detail in Section 4.

The operations of 2-, 3- and $\overline{3}$-sum have the non-trivial property that when two codes $\mathcal{C}_1$ and $\mathcal{C}_2$ are composed using one of these sums to form a code $\mathcal{C}$, then $\mathcal{C}_1$ and $\mathcal{C}_2$ are (up to code equivalence) minors of $\mathcal{C}$. The relationship between k-connectedness and these sums can then be summarized as follows: a binary linear code is 2-connected but not 3-connected (resp. 3-connected, but not internally 4-connected) iff it can be expressed as the 2-sum (resp. 3- or $\overline{3}$-sum) of codes $\mathcal{C}_1$ and $\mathcal{C}_2$, both of which are equivalent to minors of $\mathcal{C}$. It follows immediately from the above facts that any binary linear code is either 3-connected and internally 4-connected, or can be constructed from 3-connected, internally 4-connected minors of it by a sequence of operations of coordinate permutation, direct sum, 2-sum, 3-sum and $\overline{3}$-sum. In fact, given any code, such a decomposition of the code can be obtained in time polynomial in the length of the code.

This code decomposition theory has immediate applications to families of codes that are minor-closed in the sense that for each code $\mathcal{C}$ in such a family $\mathfrak{C}$, any code equivalent to a minor of $\mathcal{C}$ is also in $\mathfrak{C}$. Indeed, a code $\mathcal{C}$ is in a minor-closed family $\mathfrak{C}$ only if the indecomposable (*i.e.*, 3-connected, internally 4-connected) pieces obtained in the aforementioned decomposition of $\mathcal{C}$ are in $\mathfrak{C}$. The above necessary condition is also sufficient if the code family $\mathfrak{C}$ is additionally closed under the operations of direct sum, 2-sum, 3-sum and $\overline{3}$-sum. Thus, membership of an arbitrary code in such a family $\mathfrak{C}$ can be decided in polynomial time iff the membership in $\mathfrak{C}$ of 3-connected, internally 4-connected codes can be decided in polynomial time. A formal statement of these and other related facts can be found in Section 5 of our paper.

As an illustrative example, we also outline in Section 5 one of the major applications of Seymour's decomposition theory. This concerns the family of regular codes which are codes that do not contain as a minor any code equivalent to the $[7,4]$ Hamming code or its dual. Regular codes are also characterized by the property that given any parity-check matrix H of such a code, the 1's in H can be replaced by ± 1's in such a way that the resulting $0/\pm 1$ matrix is totally unimodular. A totally unimodular matrix is a real matrix all of whose square submatrices have determinants in $\{0, 1, -1\}$. These matrices are of fundamental importance in integer linear programming problems [15]. Seymour [6] proved that a binary linear code is regular iff it can be decomposed into codes that are either graphic, or duals of graphic codes, or equivalent to a special $[10,5,4]$ code he called R_{10}.

The application of the decomposition theory to linear programming, and in particular to ML decoding, is the subject of Section 6. Feldman *et al.* showed that the ML decoding problem for a length-n binary linear code $\mathcal{C}$ over a binary-input discrete memoryless channel can be formulated as a minimization problem $\min \sum_i \gamma_i c_i$, where $\gamma = (\gamma_1, \ldots, \gamma_n) \in \mathbb{R}^n$ is a certain cost vector derived from the received word and the channel transition probabilites, and the minimum is taken over all codewords $\mathbf{c} = (c_1, \ldots, c_n)$ in $\mathcal{C}$. Now, if $\mathcal{C}$ is a graphic code, then standard graph-theoretic techniques from combinatorial optimization can be used to find the minimizing codeword in time polynomial in n; a sketch of such an algorithm can be found in Appendix B. Grötschel and Truemper [8] additionally showed that this minimization could also be performed in polynomial time for certain minor-closed code families that are "almost-graphic" in a certain sense. Such a code family $\mathfrak{C}$ is characterized by the property that there exists a *finite* list of codes $\mathfrak{D}$ such that each $\mathcal{C} \in \mathfrak{C}$ can be decomposed in polynomial time in such a way that at each step of the decomposition, one of the pieces is either graphic or in $\mathfrak{D}$.

Grötschel and Truemper gave a polynomial-time algorithm that takes as input a length-n code $\mathcal{C}$ from an almost-graphic family $\mathfrak{C}$ and a cost vector $\gamma \in \mathbb{R}^n$, and constructs a codeword $\mathbf{c} \in \mathcal{C}$ achieving $\min \sum_i \gamma_i c_i$ by solving related minimization problems over the pieces of the decomposition that are graphic or in $\mathfrak{D}$. This algorithm is also outlined in Appendix B. Thus, the ML decoding problem can be solved in polynomial time for almost-graphic codes.

Grötschel and Truemper also gave several examples of almost-graphic families. Interestingly enough, one of these families is that consisting of codes $\mathcal{C}$ for which the codeword polytope (*i.e.*, the convex hull in $\mathbb{R}^n$ of the codewords in the length-n code $\mathcal{C}$) is identical to the Koetter-Vontobel fundamental polytope [16] derived from the entire dual code $\mathcal{C}^\perp$.

Unfortunately, the one truly original result in this paper is a negative result. We show that for codes in an almost-graphic family, either their dimension or their minimum distance grows sublinearly with codelength. One important implication of this is that decoding by linear programming, when applied to any good error-correcting code, must inevitably hit upon the occasional pseudocodeword, thus resulting in decoding failure.

We make some concluding remarks in Section 7. Some of the lengthier or more technical proofs of results from Sections 4 and 6 are given in appendices to preserve the flow of the presentation.

2. Matroids and Codes

We shall assume familiarity with coding theory; for relevant definitions, see [17]. We will mainly concern ourselves with binary linear codes, and use standard coding-theoretic notation throughout this paper. In particular, unless explicitly specified otherwise, we will the unqualified term "code" to mean "binary linear code". Thus, an $[n,k]$ code is a binary linear code of length n and dimension k, and an $[n,k,d]$ code is an $[n,k]$ code that has minimum distance d. Given a code $\mathcal{C}$, $\dim(\mathcal{C})$ denotes the dimension of $\mathcal{C}$, and $\mathcal{C}^\perp$ denotes the dual code of $\mathcal{C}$.

The main purpose of this section is to introduce concepts from matroid theory that are applicable to coding theory. We will largely follow the definitions and notation of Oxley [18]. We begin with a definition of matroids.

Definition 2.1. *A* matroid M *is an ordered pair* $(E,\mathcal{I})$ *consisting of a finite set* E *and a collection* $\mathcal{I}$ *of subsets of* E *satisfying the following three conditions:*

(i) $\emptyset \in \mathcal{I}$;
(ii) *if* $I \in \mathcal{I}$ *and* $J \subset I$, *then* $J \in \mathcal{I}$; *and*
(iii) *if* I_1, I_2 *are in* $\mathcal{I}$ *and* $|I_1| < |I_2|$, *then there exists*[1] $e \in I_2 - I_1$ *such that* $I_1 \cup \{e\} \in \mathcal{I}$.

The set E above is called the *ground set* of the matroid M, and the members of $\mathcal{I}$ are the *independent sets* of M. A maximal independent set, *i.e.*, a set $B \in \mathcal{I}$ such that $B \cup \{e\} \notin \mathcal{I}$ for any $e \in E - B$, is called a *basis* of M. It is a simple consequence of (iii) in Definition 2.1 that all bases of M have the same cardinality. The cardinality of any basis of M is defined to be the *rank* of M, denoted by $r(M)$.

A subset of E that is not in $\mathcal{I}$ is called a *dependent set*. Minimal dependent sets, *i.e.*, dependent sets all of whose proper subsets are in $\mathcal{I}$, are called *circuits*. It easily follows from the definitions that a subset of E is a dependent set if and only if it contains a circuit. A dependent set that can be expressed as a disjoint union of circuits is called a *cycle*.

The above definitions of independent and dependent sets, bases and rank simply try to abstract the notion of independence and dependence, bases and dimension, respectively, in a vector space over a field. Indeed, the most important class of matroids for our purposes is the class of binary matroids, which are simply vector spaces over the binary field, or to put it another way, binary linear codes.

Let H be a binary $m \times n$ matrix, and let $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ denote the column vectors of H. Set $E = \{1, 2, \ldots, n\}$ and take $\mathcal{I}$ to be the collection of subsets $I = \{i_1, i_2, \ldots, i_s\} \subset E$ such that the sequence of vectors $\mathbf{v}_{i_1}, \mathbf{v}_{i_2}, \ldots, \mathbf{v}_{i_s}$ is linearly independent over the binary field $\mathbb{F}_2$. It follows from elementary linear algebra that $(E,\mathcal{I})$ satisfies the definition of a matroid given above, and thus defines a matroid which we shall denote by $M[H]$. Note that $r(M[H])$ equals the rank (over $\mathbb{F}_2$) of the matrix H.

[1] In this paper, we will use $A - B$ to denote the set difference $A \cap B^c$. The more usual notation $A \setminus B$ has been reserved for the matroid operation of "deletion".

A matroid $M = (E, \mathcal{I})$ is called *binary* if it is isomorphic to $M[H]$ for some binary matrix H. Here, we say that two matroids $M = (E, \mathcal{I})$ and $M' = (E', \mathcal{I}')$ are *isomorphic*, denoted by $M \cong M'$, if there is a bijection $\psi : E \to E'$ such that for all $J \subset E$, it is the case that $J \in \mathcal{I}$ if and only if $\psi(J) \in \mathcal{I}'$.

A binary matrix H is also the parity-check matrix of some binary linear code $\mathcal{C}$. Note that $r(M[H]) = n - \dim(\mathcal{C})$. The code $\mathcal{C}$ and the binary matroid $M[H]$ are very closely related. Recall from coding theory that a codeword $\mathbf{c} = (c_1 c_2 \ldots c_n) \in \mathcal{C}$, $\mathbf{c} \neq \mathbf{0}$, is called *minimal* if its support $\mathsf{supp}(\mathbf{c}) = \{i : c_i = 1\}$ does not contain as a subset the support of any other nonzero codeword in $\mathcal{C}$. It is easily seen that $\mathbf{c}$ is a minimal codeword of $\mathcal{C}$ iff its support is a circuit of $M[H]$. It follows from this that for any $\mathbf{c} \in \{0,1\}^n$, $\mathbf{c} \neq \mathbf{0}$, we have $\mathbf{c} \in \mathcal{C}$ iff $\mathsf{supp}(\mathbf{c})$ is a cycle of $M[H]$. Furthermore, a routine verification shows that for binary matrices H and H', $M[H] = M[H']$ iff H and H' are parity-check matrices of the same code $\mathcal{C}$. This allows us to associate a unique binary matroid with each binary linear code $\mathcal{C}$, and vice versa.

Thus, binary matroids and binary linear codes are essentially the same objects. In particular, two codes are equivalent[2] if and only if their associated binary matroids are isomorphic. This association between codes and binary matroids allows us to use tools from matroid theory to study binary linear codes.

While many of the tools used to study matroids have their roots in linear algebra, there is another source that matroid theory draws from, namely, graph theory. Indeed, Whitney's founding paper on matroid theory [19] was an attempt to capture the fundamental properties of independence that are common to graphs and matrices.

Let $\mathcal{G}$ be a finite undirected graph (henceforth simply "graph") with edge set E. Define cyc to be the collection of edge sets of cycles (*i.e.*, closed walks) in $\mathcal{G}$. Define $I \subset E$ to be independent if I does not contain any member of cyc as a subset. Equivalently, I is independent if the subgraph of $\mathcal{G}$ induced by I is a forest. Setting $\mathcal{I}$ to be the collection of independent subsets of E, it turns out that $(E, \mathcal{I})$ is a matroid [18, Proposition 1.1.7]. This matroid is called the *cycle matroid* of $\mathcal{G}$, and is denoted by $M(\mathcal{G})$. A matroid that is isomorphic to the cycle matroid of some graph is called *graphic*.

Clearly, the circuits of $M(\mathcal{G})$ are the edge sets of simple cycles (*i.e.*, closed walks in which no intermediate vertex is visited twice) in $\mathcal{G}$. The nomenclature "cycle" for the disjoint union of circuits in a matroid actually stems from its use in the context of graphs. The bases of $M(\mathcal{G})$ are the unions of edge sets of spanning trees of the connected components of $\mathcal{G}$. Hence, $r(M(\mathcal{G})) = |V(\mathcal{G})| - t$, where $V(\mathcal{G})$ is the set of vertices of $\mathcal{G}$, and t is the number of connected components of $\mathcal{G}$.

It is not hard to show that a graphic matroid is binary. Indeed, let A be the vertex-edge incidence matrix of $\mathcal{G}$. This is the matrix $[a_{i,j}]$ whose rows and columns are indexed by the vertices and edges, respectively, of $\mathcal{G}$, where $a_{i,j} = 1$ if the jth edge is incident with the ith vertex, and $a_{i,j} = 0$ otherwise. It may be verified that $M(\mathcal{G}) \cong M[A]$ (see, *e.g.*, [18, Proposition 5.1.2]).

Given a graph $\mathcal{G}$, we will denote by $\mathcal{C}(\mathcal{G})$ the code associated (or identified) with the binary matroid $M(\mathcal{G})$. In other words, $\mathcal{C}(\mathcal{G})$ is the binary linear code that has the vertex-edge incidence matrix of $\mathcal{G}$ as a parity-check matrix. We will refer to such codes as *graphic codes*, and denote by Γ the set of all graphic codes. Graphic codes have made their appearance previously in the information theory literature [20, 10] (also see [11] and the references therein).

The repetition code of length n is a graphic code; it is the code obtained from the n-cycle C_n, the graph consisting of a single cycle on n vertices. However, not all binary codes are graphic. For example, it can be shown that the [7,4] Hamming code is not graphic. It is possible to give a precise characterization of the codes that are graphic in terms of excluded minors, a notion we need to first define.

[2] In coding theory, two binary linear codes are defined to be *equivalent* if one can be obtained from the other by a permutation of coordinates. In this paper, we will use the notation $\pi(\mathcal{C})$ to denote the code obtained by applying the coordinate permutation π to the code $\mathcal{C}$.

There are two well-known ways of obtaining codes of shorter length from a given parent code. One is via the operation of *puncturing*, in which one or more columns are deleted from a generator matrix of the parent code [17, p. 28]. The second method is called *shortening*, and involves one or more columns being deleted from a parity-check matrix of the parent code [17, p. 29]. Given a code $\mathcal{C}$ of length n with generator matrix G, and a subset $J \subset \{1, 2, \ldots, n\}$, we will denote by $\mathcal{C}/J$ the code obtained by puncturing the columns of G with indices in J, and by $\mathcal{C} \setminus J$ the code obtained by shortening at the columns of G with indices in J. Note that $\mathcal{C}/J$ is simply the restriction of the code $\mathcal{C}$ onto the coordinates not in J, and $\mathcal{C} \setminus J = (\mathcal{C}^{\perp}/J)^{\perp}$. The notation, though potentially confusing, has been retained from matroid theory, where the analogues of puncturing and shortening are called *contraction* and *deletion*, respectively.

Definition 2.2. *A* minor *of a code $\mathcal{C}$ is any code obtained from $\mathcal{C}$ via a (possibly empty) sequence of shortening and puncturing operations.*

It may easily be verified that the precise order in which the shortening and puncturing operations are performed is irrelevant. Hence, any minor of $\mathcal{C}$ may be unambiguously specified using notation of the form $\mathcal{C}/X \setminus Y$ (or equivalently, $\mathcal{C} \setminus Y/X$) for disjoint subsets $X, Y \subset \{1, 2, \ldots, n\}$; this notation indicates that $\mathcal{C}$ has been punctured at the coordinates indexed by X and shortened at the coordinates indexed by Y.

The above definition allows a code to be a minor of itself. A minor of $\mathcal{C}$ that is not $\mathcal{C}$ itself is called a *proper minor* of $\mathcal{C}$. Minors have not received much attention in classical coding theory, but they play a central role in matroid theory. We will not touch upon the subject of minors of general matroids, leaving the reader to refer to [18, Chapter 3] instead. However, we will briefly mention how the matroid operations of deletion and contraction specialize to the cycle matroids of graphs.

Let $\mathcal{G}$ be some graph, with edge set E. Given $e \in E$, define the graph $\mathcal{G} \setminus e$ to be the graph obtained by deleting the edge e along with any vertices that get isolated as a result of deleting e. Also, define $\mathcal{G}/e$ to be the graph obtained by contracting e, *i.e.*, deleting e and identifying the two vertices incident with e. The process of obtaining $\mathcal{G}/e$ from $\mathcal{G}$ is called *edge contraction*, and that of obtaining $\mathcal{G} \setminus e$ from $\mathcal{G}$ is of course called *edge deletion*. These operations are inductively extended to define $\mathcal{G} \setminus J$ and $\mathcal{G}/J$ for any $J \subset E$. A minor of a graph $\mathcal{G}$ is any graph obtained from $\mathcal{G}$ via a (possibly empty) sequence of edge deletions and contractions.

The operations of edge deletion and contraction are the graphic analogues of code shortening and puncturing, respectively. A mathematically precise statement of this is as follows: given a graph $\mathcal{G}$ with edge set E, and any $J \subset E$, we have [18, Equation 3.1.2 and Proposition 3.2.1]

$$\mathcal{C}(\mathcal{G})/J = \mathcal{C}(\mathcal{G}/J) \quad \text{and} \quad \mathcal{C}(\mathcal{G}) \setminus J = \mathcal{C}(\mathcal{G} \setminus J).$$

It follows that any minor of a graphic code is graphic.

Returning to the question of determining which codes are graphic, the answer can be succinctly given in terms of a list of forbidden minors by the following result of Tutte [21].

Theorem 2.1 ([21])**.** *A code is graphic if and only if it does not contain as a minor any code equivalent to the [7,4] Hamming code or its dual, or one of the codes $\mathcal{C}(K_5)^{\perp}$ and $\mathcal{C}(K_{3,3})^{\perp}$.*

In the statement of the above theorem, K_5 is the complete graph on five vertices, while $K_{3,3}$ is the complete bipartite graph with three vertices on each side. $\mathcal{C}(K_5)^{\perp}$ is the $[10, 4, 4]$ code with generator matrix

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

while $\mathcal{C}(K_{3,3})^\perp$ is the $[9, 5, 3]$ code with generator matrix

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

A proof of the theorem can be found in [18, Section 13.3]. On a related note, several authors have given algorithms for deciding whether or not a given code is graphic [22, 23, 24, 25], [26, Section 10.6]. These algorithms run in time polynomial in the size of the input, which can be a parity-check matrix for the code. We will use this fact later in the paper.

3. CONNECTEDNESS

As mentioned previously, matroid theory draws upon ideas from graph theory. A key concept in graph theory is the notion of k-connectedness for graphs [27, Section III.2]. Given a graph $\mathcal{G}$, we will let $V(\mathcal{G})$ denote the set of its vertices. A graph is *connected* if any pair of its vertices can be joined by a path; otherwise, it is *disconnected*. A maximal connected subgraph of $\mathcal{G}$ is a *connected component*, or simply *component*, of $\mathcal{G}$. Let $\mathcal{G} - W$ denote the graph obtained from $\mathcal{G}$ by deleting the vertices in W and all incident edges. If $\mathcal{G}$ is connected and, for some subset $W \subset V(\mathcal{G})$, $\mathcal{G} - W$ is disconnected, then we say that W is a *vertex cut* of $\mathcal{G}$, or that W *separates* $\mathcal{G}$. If $\mathcal{G}$ is a connected graph that has at least one pair of distinct non-adjacent vertices, the *connectivity* $\kappa(\mathcal{G})$ of $\mathcal{G}$ is defined to be the smallest integer j for which $\mathcal{G}$ has a vertex cut W with $|W| = j$. If $\mathcal{G}$ is connected, but has no pair of distinct non-adjacent vertices, $\kappa(\mathcal{G})$ is defined to be $|V(\mathcal{G})| - 1$. Finally, if $\mathcal{G}$ is disconnected, then we set $\kappa(\mathcal{G}) = 0$. For an integer $k > 0$, $\mathcal{G}$ is said to be *k-connected* if $\kappa(\mathcal{G}) \geq k$. Thus, a graph $\mathcal{G}$ with $|V(\mathcal{G})| \geq 2$ is connected if and only if it is 1-connected.

The notion of k-connectedness of graphs can be extended to matroids, but it has to be done carefully. One of the problems encountered when attempting to do so is that 1-connectedness of graphs does not extend directly to matroids. The reason for this is that for any disconnected graph $\mathcal{G}_1$, there is a connected graph $\mathcal{G}_2$ such that $M(\mathcal{G}_1) \cong M(\mathcal{G}_2)$ [18, Proposition 1.2.8]. So, the link between k-connectedness in graphs and that defined below for matroids begins with the case $k = 2$.

The definition we present of k-connectedness for matroids was formulated by Tutte [28]. We will once again restrict our attention to the case of binary matroids (*i.e.*, codes) only. Let $\mathcal{C}$ be a binary linear code of length n. We will hereafter use $[n]$ to denote the set of integers $\{1, 2, \ldots, n\}$, and for $J \subset [n]$, we set $J^c = \{i \in [n] : i \notin J\}$. To further alleviate notational confusion, for $J \subset [n]$, we will define $\mathcal{C}|_J$ to be the restriction of $\mathcal{C}$ onto its coordinates indexed by J. Equivalently, $\mathcal{C}|_J = \mathcal{C}/J^c$, the latter being the code obtained from $\mathcal{C}$ by puncturing the coordinates not in J.

Definition 3.1. *For a positive integer k, a partition (J, J^c) of $[n]$ is called a k-*separation *of $\mathcal{C}$ if*

$$\min\{|J|, |J^c|\} \geq k \tag{1}$$

and

$$\dim(\mathcal{C}|_J) + \dim(\mathcal{C}|_{J^c}) - \dim(\mathcal{C}) \leq k - 1. \tag{2}$$

*If $\mathcal{C}$ has a k-separation, then $\mathcal{C}$ is said to be k-*separated.

When equality occurs in (1), (J, J^c) is called a *minimal k-separation*. When equality occurs in (2), (J, J^c) is called an *exact k-separation*. Note that the expression on the left-hand side of (2) is always non-negative, since $\dim(\mathcal{C}|_J) + \dim(\mathcal{C}|_{J^c}) \geq \dim(\mathcal{C})$ for any $J \subset [n]$. This fact easily yields the following result.

Lemma 3.1. *$\mathcal{C}$ is 1-separated iff it is the direct sum of non-empty codes.*

Proof. Since $\dim(\mathcal{C}|_J) + \dim(\mathcal{C}|_{J^c}) \geq \dim(\mathcal{C})$ for any $J \subset [n]$, we see from Definition 3.1 that (J, J^c) is a 1-separation of $\mathcal{C}$ iff J, J^c are non-empty, and $\dim(\mathcal{C}|_J)+\dim(\mathcal{C}|_{J^c}) = \dim(\mathcal{C})$. Hence, (J, J^c) is a 1-separation of $\mathcal{C}$ iff J, J^c are non-empty, and $\mathcal{C}$ is the direct sum of $\mathcal{C}|_J$ and $\mathcal{C}|_{J^c}$. □

We now give the definition of k-connectedness of codes. Note that this definition starts with $k = 2$.

Definition 3.2. *For $k \geq 2$, a code $\mathcal{C}$ is defined to be k-*connected *if it has no k'-separation for any $k' < k$.*

Example 3.1. *Let $\mathcal{C}$ be the [7,3,4] simplex code with generator matrix*

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

Setting $J = \{1, 2, 3, 7\}$, we see that (J, J^c) forms a 3-separation of $\mathcal{C}$. Indeed, the rank of the submatrix of G formed by the columns indexed by J is 3, while the rank of the submatrix formed by the columns indexed by J^c is 2. In other words, $\dim(\mathcal{C}|_J) = 3$ and $\dim(\mathcal{C}|_{J^c}) = 2$. Thus, both (1) and (2) are satisfied with equality, which makes (J, J^c) a minimal as well as an exact separation.

It may be verified (for example, by exhaustive search) that there are no 1- or 2- separations for $\mathcal{C}$, and all 3-separations are minimal and exact. In particular, $\mathcal{C}$ is 2- and 3-connected, but not 4-connected.

The quantity, $\dim(\mathcal{C}|_J) + \dim(\mathcal{C}|_{J^c}) - \dim(\mathcal{C})$, appearing on the left-hand side of (2) in Definition 3.1 also arises as part of the definition of the state-complexity profile of a minimal trellis representation of a code [29],[12],[30],[31],[32]. To be precise, given a length-n code $\mathcal{C}$, the *state-complexity profile* of a code [31, Equation (1)] is defined to be the vector $\mathbf{s}(\mathcal{C}) = (s_0(\mathcal{C}), \ldots, s_n(\mathcal{C}))$, where $s_i(\mathcal{C}) = \dim(\mathcal{C}|_J) + \dim(\mathcal{C}|_{J^c}) - \dim(\mathcal{C})$ for $J = [i] \subset [n]$. Here, $[0]$ is defined to be the null set $\emptyset$. It is known [12] that $\mathbf{s}(\mathcal{C}) = \mathbf{s}(\mathcal{C}^\perp)$, or equivalently, for any $J \subset [n]$,

$$\dim(\mathcal{C}|_J) + \dim(\mathcal{C}|_{J^c}) - \dim(\mathcal{C}) = \dim(\mathcal{C}^\perp|_J) + \dim(\mathcal{C}^\perp|_{J^c}) - \dim(\mathcal{C}^\perp).$$

As a result, we obtain the interesting and useful fact, stated in the proposition below, that k-connectedness is a property that is invariant under the operation of taking code duals.

Proposition 3.2. *Let $\mathcal{C}$ be a binary linear code of length n. For any $k \geq 1$, a partition (J, J^c) of $[n]$ is a k-separation of $\mathcal{C}$ iff it is a k-separation of $\mathcal{C}^\perp$. Therefore, for any $k \geq 2$, $\mathcal{C}$ is k-connected iff $\mathcal{C}^\perp$ is k-connected.*

Consider again the simplex code of length 7 from Example 3.1. By Proposition 3.2 above, its dual — the $[7, 4]$ Hamming code — is also 2- and 3-connected, but not 4-connected.

The link between graph and code k-connectedness is strong, but they are not equivalent notions. The closest relation between the two occurs when $k = 2$. If $\mathcal{G}$ is a loopless graph without isolated vertices and $|V(\mathcal{G})| \geq 3$, then $\mathcal{C}(\mathcal{G})$ is 2-connected iff $\mathcal{G}$ is a 2-connected graph [18, Proposition 4.1.8]. To describe the relation between graph and code connectedness in general, we define the *connectivity*, $\lambda(\mathcal{C})$, of a code $\mathcal{C}$ to be the least positive integer k for which there is a k-separation of $\mathcal{C}$, if some k-separation exists for $\mathcal{C}$; $\lambda(\mathcal{C})$ is defined to be ∞ otherwise. Note that $\mathcal{C}$ is k-connected iff $\lambda(\mathcal{C}) \geq k$, and by Proposition 3.2, $\lambda(\mathcal{C}) = \lambda(\mathcal{C}^\perp)$. It can be shown [18, Corollary 8.2.7] that for a connected graph $\mathcal{G} \neq K_3$ having at least three vertices,

$$\lambda(\mathcal{C}(\mathcal{G})) = \min\{\kappa(\mathcal{G}), g(\mathcal{G})\},$$

where $g(\mathcal{G})$ denotes the girth (length of shortest cycle) of $\mathcal{G}$.

Now, our reason for presenting a notion of connectedness for codes is not just that it extends an idea from graph theory. Certain methods of code composition have been developed in matroid theory that relate to 2- and 3-separations. These code composition methods can be considered to be generalizations of direct sums, and they allow the result of Lemma 3.1 to be extended in a nontrivial manner, paving the way for the powerful decomposition theory of binary matroids initiated by Paul Seymour [6]. This decomposition theory allows one to decompose a binary linear code into smaller codes in a reversible manner, in such a way that the smaller codes are equivalent to minors of the original code. As we shall describe in detail in the next section, to find such a decomposition of a code, we need to find 1-, 2- or 3-separations in the code, if such separations exist.

For any fixed positive integer k, there are polynomial-time algorithms known (see [26, Section 8.4]) that, given a binary linear code $\mathcal{C}$, either find a k-separation of $\mathcal{C}$, or conclude that no such separation exists. Here, by "polynomial-time algorithm," we mean an algorithm that runs in time polynomial in the length of $\mathcal{C}$. For instance, the problem of deciding the existence of 1-separations in a code $\mathcal{C}$ is almost trivial. To do so, one takes a matrix A that is either a generator matrix or a parity-check matrix of $\mathcal{C}$, brings A to reduced row-echelon form (rref), removes all-zero rows if they exist, and finally constructs a certain bipartite graph $BG(A)$. For an $m \times n$ matrix A, the graph $BG(A)$ is defined as follows[3]: the vertex set of $BG(A)$ consists of a set of n left vertices $\{l_1, \ldots, l_n\}$ and a set of m right vertices $\{r_1, \ldots, r_m\}$; an edge connects the vertices l_j and r_i iff the (i, j)th entry of A is 1. The code $\mathcal{C}$ is 2-connected (*i.e.*, has no 1-separation) iff $BG(A)$ is connected [26, Lemma 3.3.19]. If $\mathcal{C}$ is not 2-connected, the connected components of $BG(A)$ induce the required 1-separation.

In general, for fixed integers k, l with $l \geq k$, the problem of finding a k-separation (J, J^c) of a code, with $\min\{|J|, |J^c|\} \geq l$, if it exists, can be solved in time polynomial in the length of the code, by an algorithm due to Cunningham and Edmonds (in [33]). We sketch the idea here. The algorithm is based on the fact that the following problem can be solved in time polynomial in n: for codes $\mathcal{C}_1$ and $\mathcal{C}_2$ each of length n, find a partition of $[n]$ that achieves

$$\min\{\dim(\mathcal{C}_1|_{J_1}) + \dim(\mathcal{C}_2|_{J_2}) : \ (J_1, J_2) \text{ is a partition of } [n]\}. \tag{3}$$

The above problem is solved using the *matroid intersection algorithm* [34], [26, Section 5.3], which we do not describe here.

The k-separation problem of interest to us is equivalent to the following problem for a fixed integer $l \geq k$: given a code $\mathcal{C}$ of length n, find a partition of $[n]$ that achieves

$$\min\{\dim(\mathcal{C}|_{J_1}) + \dim(\mathcal{C}|_{J_2}) : \ (J_1, J_2) \text{ is a partition of } [n], \ \min\{|J_1|, |J_2|\} \geq l\}. \tag{4}$$

Indeed, a k-separation $(|J|, |J^c|)$ of $\mathcal{C}$, with $\min\{|J|, |J^c|\} \geq l$, exists iff the minimum in (4) is at most $\dim(\mathcal{C}) + k - 1$. Now, the minimization in (4) can be solved by finding, for each pair of disjoint l-element subsets $E_1, E_2 \subset [n]$, the partition (J_1, J_2) of $[n]$ that achieves

$$\min\{\dim(\mathcal{C}|_{J_1}) + \dim(\mathcal{C}|_{J_2}) : \ (J_1, J_2) \text{ is a partition of } [n], \ J_1 \supset E_1, \ J_2 \supset E_2\}. \tag{5}$$

If (5) can be solved in time polynomial in n for each pair of disjoint l-element subsets $E_1, E_2 \subset [n]$, then (4) can also be solved in time polynomial in n, since there are $O(n^{2l})$ pairs (E_1, E_2).

It turns out that (5) can be solved in time polynomial in n by converting it to a minimization of the form in (3), which, as mentioned above, can be solved in polynomial time. The trick is to set $\mathcal{C}_1 = \mathcal{C}/E_2 \setminus E_1$ and $\mathcal{C}_2 = \mathcal{C}/E_1 \setminus E_2$, which are both codes of length $n - |E_1 \cup E_2|$. For notational convenience, we will let the coordinates of $\mathcal{C}_1$ and $\mathcal{C}_2$ retain their indices from $\mathcal{C}$, *i.e.*, the set of coordinate indices for $\mathcal{C}_1$, as well as for $\mathcal{C}_2$, is $[n] - (E_1 \cup E_2)$. It may easily be verified that for $J \subset [n] - (E_1 \cup E_2)$, $\dim(\mathcal{C}_i|_J) = \dim(\mathcal{C}|_{J \cup E_i}) - \dim(\mathcal{C}|_{E_i})$, $i = 1, 2$. Therefore, for any partition (J_1, J_2) of $[n] - (E_1 \cup E_2)$, setting $\overline{J_i} = J_i \cup E_i$, $i = 1, 2$, we have

$$\dim(\mathcal{C}_1|_{J_1}) + \dim(\mathcal{C}_2|_{J_2}) = \dim(\mathcal{C}|_{\overline{J}_1}) + \dim(\mathcal{C}|_{\overline{J}_2}) - \dim(\mathcal{C}|_{E_1}) - \dim(\mathcal{C}|_{E_2}), \tag{6}$$

and $(\overline{J_1}, \overline{J_2})$ is a partition of $[n]$. Conversely, for any partition $(\overline{J_1}, \overline{J_2})$ of $[n]$, setting $J_i = \overline{J_i} - E_i$, $i = 1, 2$, we see that (J_1, J_2) forms a partition of $[n] - (E_1 \cup E_2)$, and (6) is once again satisfied.

[3] If A is a parity-check matrix of the code, then $BG(A)$ is simply the corresponding Tanner graph.

Thus, we see that for a fixed pair of disjoint l-element subsets $E_1, E_2 \subset [n]$, given a code $\mathcal{C}$ of length n, if we set $\mathcal{C}_1 = \mathcal{C}/E_2 \setminus E_1$ and $\mathcal{C}_2 = \mathcal{C}/E_1 \setminus E_2$, then the minimum in (5) is equal to

$$\min\{\dim(\mathcal{C}_1|_{J_1})+\dim(\mathcal{C}_2|_{J_2}) :\ (J_1, J_2) \text{ is a partition of } [n]-(E_1\cup E_2)\}+\dim(\mathcal{C}|_{E_1})+\dim(\mathcal{C}|_{E_2}).$$

Therefore, for a given $\mathcal{C}$ and (E_1, E_2), the minimization problem

$$\min\{\dim(\mathcal{C}_1|_{J_1}) + \dim(\mathcal{C}_2|_{J_2}) :\ (J_1, J_2) \text{ is a partition of } [n] - (E_1 \cup E_2)\},$$

which is just an instance of (3), is equivalent to (5), as the two minima only differ by a constant. We conclude that since (3) can be solved in time polynomial in n, so can (5).

The above sketch does indeed give a polynomial-time algorithm for determining k-separations (J, J^c) with $\min\{|J|, |J^c|\} \geq l$, but the complexity of the algorithm is $O(n^{2l+\alpha_l})$ for some constant α_l that arises from the matroid intersection algorithm. Clearly, this is not very practical even for $l = 3$ or 4, and as we shall see in the next section, these values of l come up in the implementation of Seymour's decomposition theory. A more efficient, albeit more involved, algorithm for finding 2- and 3-separations is described in [26, Section 8.4].

As a final remark in this section, we mention that the fact that there exist algorithms for solving the minimization problems (3)–(5) that run in time polynomial in n neither contradicts nor sheds any further light on the NP-completeness results for the closely related problems considered in [31].

4. Code Composition and Decomposition

The code composition/decomposition methods described in this section were developed by Seymour in close analogy with a method of composing/decomposing graphs called *clique-sum*. In a clique-sum, two graphs, each containing a K_k subgraph (k-clique), are glued together by first picking a k-clique from each graph, sticking the two cliques together so as to form a single k-clique in the composite graph, and then deleting some or all of the edges from this clique. A formal description of clique-sum can be found in [6] or in [18, p. 420].

Our exposition of these code composition/decomposition techniques is based on Seymour's paper [6]. Let $\mathcal{C}$ and $\mathcal{C}'$ be binary linear codes of length n and n', respectively, and let m be an integer satisfying $0 \leq 2m < \min\{n, n'\}$. We first define a code $\mathcal{C} \parallel_m \mathcal{C}'$ as follows: if $G = [\mathbf{g}_1\ \mathbf{g}_2\ \ldots\ \mathbf{g}_n]$ and $G' = [\mathbf{g}'_1\ \mathbf{g}'_2\ \ldots\ \mathbf{g}'_{n'}]$ are generator matrices of $\mathcal{C}$ and $\mathcal{C}'$, respectively, then $\mathcal{C} \parallel_m \mathcal{C}'$ is the code with generator matrix

$$\begin{bmatrix} \mathbf{g}_1 & \ldots & \mathbf{g}_{n-m} & \mathbf{g}_{n-m+1} & \ldots & \mathbf{g}_n & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{0} & \ldots & \mathbf{0} & \mathbf{g}'_1 & \ldots & \mathbf{g}'_m & \mathbf{g}'_{m+1} & \ldots & \mathbf{g}'_{n'} \end{bmatrix}.$$

Thus, $\mathcal{C} \parallel_m \mathcal{C}'$ is a binary linear code of length $n + n' - m$. This code is almost like a direct sum of $\mathcal{C}$ and $\mathcal{C}'$ except that the two component codes overlap in m positions. Indeed, when $m = 0$, $\mathcal{C} \parallel_m \mathcal{C}'$ is the direct sum of $\mathcal{C}$ and $\mathcal{C}'$.

Codewords of $\mathcal{C} \parallel_m \mathcal{C}'$ are of the form $\mathbf{c} \parallel_m \mathbf{c}'$, for $\mathbf{c} = c_1 c_2 \ldots, c_n \in \mathcal{C}$ and $\mathbf{c}' = c'_1 c'_2 \ldots, c'_{n'} \in \mathcal{C}'$, where $\mathbf{c} \parallel_m \mathbf{c}' = \hat{c}_1 \hat{c}_2 \ldots \hat{c}_{n+n'-m}$ is defined to be

$$\hat{c}_i = \begin{cases} c_i & \text{for } 1 \leq i \leq n-m \\ c_i + c'_{i-n+m} & \text{for } n-m+1 \leq i \leq n \\ c'_{i-n+m} & \text{for } n+1 \leq i \leq n+n'-m \end{cases}$$

In other words, $\mathbf{c} \parallel_m \mathbf{c}'$ is the binary word of length $n + n' - m$ composed as follows: the first $n - m$ symbols of $\mathbf{c} \parallel_m \mathbf{c}'$ are equal to the first $n - m$ symbols of $\mathbf{c}$, the next m symbols of $\mathbf{c} \parallel_m \mathbf{c}'$ are equal to the coordinatewise modulo-2 sum of the last m symbols of $\mathbf{c}$ and the first m symbols of $\mathbf{c}'$, and the last $n' - m$ symbols of $\mathbf{c} \parallel_m \mathbf{c}'$ are equal to the last $n' - m$ symbols of $\mathbf{c}'$.

From $\mathcal{C} \parallel_m \mathcal{C}'$, we derive a new code, which we temporarily denote by $\mathcal{S}_m(\mathcal{C}, \mathcal{C}')$, by shortening at the m positions where $\mathcal{C}$ and $\mathcal{C}'$ are made to overlap. To be precise, let $J = \{n-m+1, n-m+2, \ldots, n\}$, and set $\mathcal{S}_m(\mathcal{C}, \mathcal{C}') = (\mathcal{C} \parallel_m \mathcal{C}')\setminus J$. Thus, $\mathcal{S}_m(\mathcal{C}, \mathcal{C}')$ is a code of length $n+n'-2m$ which,

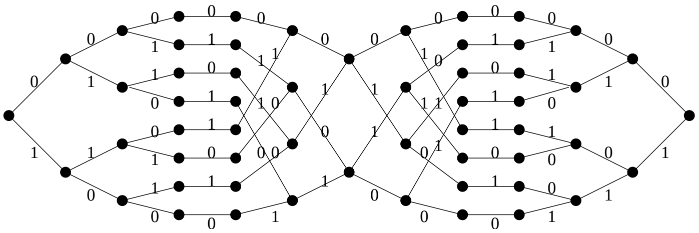

FIGURE 1. The minimal trellis of the code $\mathcal{C} \oplus_2 \mathcal{C}$ of Example 4.1.

by choice[4] of m, is greater than n and n'. Once again, note that $\mathcal{S}_0(\mathcal{C}, \mathcal{C}') = \mathcal{C} \parallel_0 \mathcal{C}' = \mathcal{C} \oplus \mathcal{C}'$, where $\oplus$ denotes direct sum.

We will actually only be interested in two instances of the above construction (other than the direct-sum case of $m = 0$), one of which is presented next, and the other is introduced later in Section 4.2.

4.1. **2-Sums.** The $\mathcal{S}_m(\mathcal{C}, \mathcal{C}')$ construction with $m = 1$ is called a 2-sum in certain special cases.

Definition 4.1. *Let $\mathcal{C}$, $\mathcal{C}'$ be codes of length at least three, such that*

(P1) $0 \ldots 01$ *is not a codeword of $\mathcal{C}$ or $\mathcal{C}^\perp$;*
(P2) $10 \ldots 0$ *is not a codeword of $\mathcal{C}'$ or $\mathcal{C}'^\perp$.*

Then, $\mathcal{S}_1(\mathcal{C}, \mathcal{C}')$ is called the 2-sum *of $\mathcal{C}$ and $\mathcal{C}'$, and is denoted by $\mathcal{C} \oplus_2 \mathcal{C}'$.*

Note that 2-sums are only defined for codes having the properties (P1) and (P2) listed in Definition 4.1. These properties can be equivalently stated as follows:

(P1$'$) $0 \ldots 01$ is not a codeword of $\mathcal{C}$, and the last coordinate of $\mathcal{C}$ is not identically zero;
(P2$'$) $10 \ldots 0$ is not a codeword of $\mathcal{C}'$, and the first coordinate of $\mathcal{C}'$ is not identically zero.

As we shall see below, (P1$'$) and (P2$'$) are more directly relevant to an analysis of the 2-sum construction.

Example 4.1. *Let $\mathcal{C}$ be the $[7, 3, 4]$ simplex code with the generator matrix given in Example 3.1. As this code satisfies both (P1) and (P2) in Definition 4.1, we can define $\mathcal{C} \oplus_2 \mathcal{C}$. Carrying out the 2-sum construction yields the $[12, 5, 4]$ code $\mathcal{C} \oplus_2 \mathcal{C}$ with generator matrix*

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

The minimal trellis for this code is shown in Figure 1. It is easily seen (from the state-complexity profile of the minimal trellis, for example) that, with $J = \{1, 2, 3, 4, 5, 6\}$, (J, J^c) is a 2-separation of $\mathcal{C} \oplus_2 \mathcal{C}$. It may further be verified that $\mathcal{C} \oplus_2 \mathcal{C}$ has no 1-separation, meaning that it is 2-connected. Finally, we note that by shortening $\mathcal{C} \oplus_2 \mathcal{C}$ at the 7th and 8th coordinates, and puncturing the 9th, 11th and 12th coordinates, we obtain the simplex code $\mathcal{C}$ again. In other words, $\mathcal{C}$ is a minor of $\mathcal{C} \oplus_2 \mathcal{C}$. These observations are not mere coincidences, as we shall see below.

The dimension and minimum distance of a 2-sum $\mathcal{C} \oplus_2 \mathcal{C}'$ can be related to the corresponding parameters of the component codes $\mathcal{C}$ and $\mathcal{C}'$. Given a binary word $\mathbf{x}$, we will use $w(\mathbf{x})$ to denote its Hamming weight, and for a code $\mathcal{C}$, we will let $d(\mathcal{C})$ denote the minimum distance of $\mathcal{C}$.

[4] The construction of $\mathcal{S}_m(\mathcal{C}, \mathcal{C}')$ would also work if we only required that $0 \le m < \min\{n, n'\}$, and not $0 \le 2m < \min\{n, n'\}$. The stronger condition is imposed so that the $\mathcal{S}_m(\mathcal{C}, \mathcal{C}')$ construction yields a code of length strictly greater than the lengths of the component codes $\mathcal{C}$ and $\mathcal{C}'$.

Proposition 4.1. *Let $\mathcal{C}$ and $\mathcal{C}'$ be codes for which $\mathcal{C} \oplus_2 \mathcal{C}'$ can be defined.*
(a) $\dim(\mathcal{C} \oplus_2 \mathcal{C}') = \dim(\mathcal{C}) + \dim(\mathcal{C}') - 1.$
(b) *If $\dim(\mathcal{C}) > 1$, then $d(\mathcal{C} \oplus_2 \mathcal{C}') \leq d(\mathcal{C} \setminus \{n\})$, where n is the length of $\mathcal{C}$. Similarly, if $\dim(\mathcal{C}') > 1$, then $d(\mathcal{C} \oplus_2 \mathcal{C}') \leq d(\mathcal{C}' \setminus \{1\})$. Otherwise, if $\dim(\mathcal{C}) = \dim(\mathcal{C}') = 1$, then $d(\mathcal{C} \oplus_2 \mathcal{C}') = d(\mathcal{C}) + d(\mathcal{C}') - 2$.*

Proof. Throughout this proof, n and n' denote the lengths of $\mathcal{C}$ and $\mathcal{C}'$, respectively. Also, we shall let $\mathbf{x} \parallel \mathbf{x}'$ denote the concatenation of binary words $\mathbf{x}$ and $\mathbf{x}'$.

(a) By definition, $\mathcal{C} \oplus_2 \mathcal{C}' = (\mathcal{C} \parallel_1 \mathcal{C}') \setminus \{n\}$, and so, the 2-sum is isomorphic to the subcode, $\mathcal{E}$, of $\mathcal{C} \parallel_1 \mathcal{C}'$ consisting of those codewords $\hat{c}_1 \hat{c}_2 \ldots \hat{c}_{n+n'-1}$ such that $\hat{c}_n = 0$. Since the last coordinate of $\mathcal{C}$ is not identically zero, $\mathcal{E}$ is a proper subcode of $\mathcal{C} \parallel_1 \mathcal{C}'$, and hence, $\dim(\mathcal{C} \oplus_2 \mathcal{C}') = \dim(\mathcal{E}) = \dim(\mathcal{C} \parallel_1 \mathcal{C}') - 1$.

We claim that the direct sum $\mathcal{C} \oplus \mathcal{C}'$ is in fact isomorphic (as a vector space over $\mathbb{F}_2$) to $\mathcal{C} \parallel_1 \mathcal{C}'$. Indeed, consider the map $\phi : \mathcal{C} \oplus \mathcal{C}' \to \mathcal{C} \parallel_1 \mathcal{C}'$ defined via $\phi(\mathbf{c} \parallel \mathbf{c}') = \mathbf{c} \parallel_1 \mathbf{c}'$, for $\mathbf{c} \in \mathcal{C}$ and $\mathbf{c}' \in \mathcal{C}'$. This is a homomorphism onto $\mathcal{C} \parallel_1 \mathcal{C}'$, but since $0\ldots01 \notin \mathcal{C}$ and $10\ldots0 \notin \mathcal{C}'$, we have $\ker(\phi) = \{\mathbf{0}\}$, which shows that ϕ is in fact an isomorphism.

Therefore, $\dim(\mathcal{C} \oplus_2 \mathcal{C}') = \dim(\mathcal{C} \parallel_1 \mathcal{C}') - 1 = \dim(\mathcal{C} \oplus \mathcal{C}') - 1$, which proves the result.

(b) If $\dim(\mathcal{C}) > 1$, then $\dim(\mathcal{C} \setminus \{n\}) \geq 1$. So, there exists a nonzero codeword in $\mathcal{C} \setminus \{n\}$. Let $\hat{\mathbf{c}}$ be a non-zero codeword of least weight in $\mathcal{C} \setminus \{n\}$. Since $\hat{\mathbf{c}}0 \in \mathcal{C}$, we have, by construction of $\mathcal{C} \oplus_2 \mathcal{C}'$, $\hat{\mathbf{c}} \parallel \mathbf{0} \in \mathcal{C} \oplus_2 \mathcal{C}'$, where $\mathbf{0}$ has length $n' - 1$. Thus, $d(\mathcal{C} \oplus_2 \mathcal{C}') \leq w(\hat{\mathbf{c}}) = d(\mathcal{C} \setminus \{n\})$. A similar argument shows that if $\dim(\mathcal{C}') > 1$, then $d(\mathcal{C} \oplus_2 \mathcal{C}') \leq d(\mathcal{C}' \setminus \{1\})$.

Suppose that $\dim(\mathcal{C}) = \dim(\mathcal{C}') = 1$, so that, by part (a) above, $\dim(\mathcal{C} \oplus_2 \mathcal{C}') = 1$ as well. Let $\mathbf{c} = (c_1, \ldots, c_n)$ and $\mathbf{c}' = (c'_1, \ldots, c'_{n'})$ be the unique nonzero codewords in $\mathcal{C}$ and $\mathcal{C}'$, respectively. By (P1′) and (P2′), we must have $c_n = c'_1 = 1$, and furthermore, $d(\mathcal{C}) = w(\mathbf{c})$ and $d(\mathcal{C}') = w(\mathbf{c}')$. By construction of $\mathcal{C} \oplus_2 \mathcal{C}'$, the unique non-zero codeword in $\mathcal{C} \oplus_2 \mathcal{C}'$ is $(c_1, \ldots, c_{n-1}, c'_2, \ldots, c'_{n'})$, which has weight equal to $w(\mathbf{c}) + w(\mathbf{c}') - 2$. □

An interesting property of 2-sums is that they behave just like direct sums under the operation of taking code duals. Note that by virtue of (P1) and (P2) in Definition 4.1, the 2-sum of $\mathcal{C}$ and $\mathcal{C}'$ can be defined if and only if the 2-sum of their duals, $\mathcal{C}^\perp$ and $\mathcal{C}'^\perp$, can be defined.

Proposition 4.2 ([18], Proposition 7.1.20)**.** *Let $\mathcal{C}$ and $\mathcal{C}'$ be codes for which $\mathcal{C} \oplus_2 \mathcal{C}'$ can be defined. Then,*

$$(\mathcal{C} \oplus_2 \mathcal{C}')^\perp = \mathcal{C}^\perp \oplus_2 \mathcal{C}'^\perp.$$

This result can be trivially derived from Theorem 7.3 in [35], but for completeness, we give a simple algebraic proof in Appendix A. As an example, the above result implies that the matrix $\overline{G}$ given in Example 4.1 is the parity-check matrix of the 2-sum of two copies of a [7,4] Hamming code.

While the properties of 2-sums presented above are interesting, the usefulness of 2-sums actually stems from the following theorem of Seymour [6], which is a result analogous to Lemma 3.1.

Theorem 4.3 ([6],Theorem 2.6)**.** *If $\mathcal{C}_1$ and $\mathcal{C}_2$ are codes of length n_1 and n_2, respectively, such that $\mathcal{C} = \mathcal{C}_1 \oplus_2 \mathcal{C}_2$, then (J, J^c) is an exact 2-separation of $\mathcal{C}$, for $J = \{1, 2, \ldots, n_1 - 1\}$. Furthermore, $\mathcal{C}_1$ and $\mathcal{C}_2$ are equivalent to minors of $\mathcal{C}$.*

Conversely, if (J, J^c) is an exact 2-separation of a code $\mathcal{C}$, then there are codes $\mathcal{C}_1$ and $\mathcal{C}_2$ of length $|J| + 1$ and $|J^c| + 1$, respectively, such that $\mathcal{C}$ is equivalent to $\mathcal{C}_1 \oplus_2 \mathcal{C}_2$.

The following corollary is a more concise statement of the above theorem, and is more in the spirit of Lemma 3.1.

Corollary 4.4. *A code $\mathcal{C}$ has an exact 2-separation iff there exist codes $\mathcal{C}_1$ and $\mathcal{C}_2$, both equivalent to proper minors of $\mathcal{C}$, such that $\mathcal{C}$ is equivalent to $\mathcal{C}_1 \oplus_2 \mathcal{C}_2$.*

Another corollary [18, Theorem 8.3.1], stated next, is a consequence of the fact that if $\mathcal{C}$ is a 2-connected code, then any 2-separation of $\mathcal{C}$ must be exact; if not, the 2-separation would be a 1-separation as well, which is impossible as $\mathcal{C}$ is 2-connected.

Corollary 4.5. *A 2-connected code $\mathcal{C}$ is not 3-connected iff there exist codes $\mathcal{C}_1$ and $\mathcal{C}_2$, both equivalent to proper minors of $\mathcal{C}$, such that $\mathcal{C}$ is equivalent to $\mathcal{C}_1 \oplus_2 \mathcal{C}_2$.*

We will not prove Theorem 4.3 in its entirety, referring the reader instead to Seymour's original proof, or the proof given in Oxley [18, Section 8.3]. However, we will describe an efficient construction of the components of the 2-sum when an exact 2-separation (J, J^c) of $\mathcal{C}$ is given, as it is a useful tool in code decomposition. This construction effectively proves the converse part of the theorem. Our description is based on the construction given in [26, Section 8.2].

Let $\mathcal{C}$ be a code of length n and dimension k, specified by a $k \times n$ generator matrix G, and let (J, J^c) be an exact 2-separation of $\mathcal{C}$. By permuting coordinates if necessary, we may assume that $J = \{1, 2, \ldots, m\}$ for some $m < n$. Let $G|_J$ and $G|_{J^c}$ denote the restrictions of G to the columns indexed by J and J^c, respectively; thus, $G = [G|_J \ \ G|_{J^c}]$. Let $\mathsf{rank}(G|_J) = k_1$ and $\mathsf{rank}(G|_{J^c}) = k_2$; since (J, J^c) is an exact 2-separation of $\mathcal{C}$, we have $k_1 + k_2 = k + 1$. Bring G into reduced row-echelon form (rref) over $\mathbb{F}_2$. Permuting coordinates within J and within J^c if necessary, $\mathsf{rref}(G)$ may be assumed to be of the form

$$\overline{G} = \begin{bmatrix} I_{k_1} & A & \mathbf{O} & B \\ \mathbf{O} & \mathbf{O} & I_{k_2-1} & C \end{bmatrix}, \tag{7}$$

where I_j, for $j = k_1, k_2 - 1$, denotes the $j \times j$ identity matrix, A is a $k_1 \times (|J| - k_1)$ matrix, B is a $k_1 \times (|J^c| - k_2 + 1)$ matrix, C is a $(k_2 - 1) \times (|J^c| - k_2 + 1)$ matrix, and the $\mathbf{O}$'s denote all-zeros matrices of appropriate sizes. As a concrete example, consider the matrix $\overline{G}$ given in Example 4.1, which is indeed of the above form, with $|J| = |J^c| = 6$, $k_1 = k_2 = 3$,

$$A = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} \text{ and } C = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix}.$$

The fact that the submatrix $\begin{bmatrix} \mathbf{O} & B \\ I_{k_2-1} & C \end{bmatrix}$ must have rank equal to $\mathsf{rank}(G|_{J^c}) = k_2$ implies that B must have rank 1. Hence, B is actually a matrix with at least one nonzero row, call it $\mathbf{b}$, and at least one nonzero column, call it $\widetilde{\mathbf{b}}$. Also, each row of B is either $\mathbf{0}$ or identical to $\mathbf{b}$, and $\widetilde{\mathbf{b}}$ is the length-k_1 column vector whose ith component is a 1 if the ith row of B is equal to $\mathbf{b}$, and is 0 otherwise. In other words, B is the product of the column vector $\widetilde{\mathbf{b}}$ with the row vector $\mathbf{b}$.

Now, define

$$G_1 = [I_{k_1} \ A \ \widetilde{\mathbf{b}}]$$

and

$$G_2 = \begin{bmatrix} 1 & \mathbf{0}^t & \mathbf{b} \\ \mathbf{0} & I_{k_2-1} & C \end{bmatrix} = [I_{k_2} \ C'],$$

where $\mathbf{0}$ denotes an all-zeros column-vector, and $C' = \begin{bmatrix} \mathbf{b} \\ C \end{bmatrix}$.

It is not hard to show that if $\mathcal{C}_1$ and $\mathcal{C}_2$ are the codes generated by G_1 and G_2, respectively, then $\mathcal{C}_1 \oplus_2 \mathcal{C}_2$ is the code generated by the matrix $\overline{G}$ in (7). Indeed, carefully going through the construction, it may be verified that all the rows of $\overline{G}$ are in $\mathcal{C}_1 \oplus_2 \mathcal{C}_2$. Hence, $\dim(\mathcal{C}_1 \oplus_2 \mathcal{C}_2) \geq \mathsf{rank}(\overline{G}) = k_1 + k_2 - 1$. However, by Proposition 4.1(a), we have that $\dim(\mathcal{C}_1 \oplus_2 \mathcal{C}_2) = \dim(\mathcal{C}_1) + \dim(\mathcal{C}_2) - 1 = k_1 + k_2 - 1$. Hence, $\dim(\mathcal{C}_1 \oplus_2 \mathcal{C}_2) = \mathsf{rank}(\overline{G})$, implying that $\overline{G}$ must be a generator matrix for $\mathcal{C}_1 \oplus_2 \mathcal{C}_2$.

Example 4.2. *For the matrix $\overline{G}$ in Example 4.1, we find the matrices G_1 and G_2 to be*

$$G_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}, \quad \textit{and} \quad G_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix},$$

which are indeed the generator matrices of the two simplex codes whose 2-sum is represented by $\overline{G}$.

It can also be observed that $\mathcal{C}_1$ and $\mathcal{C}_2$ are minors of $\mathcal{C}_1 \oplus_2 \mathcal{C}_2$. Indeed, to obtain $\mathcal{C}_1$ as a minor of $\mathcal{C}_1 \oplus_2 \mathcal{C}_2$, we proceed as follows. Let j be the index of a column of $\overline{G}$ in which the submatrix B has a nonzero column. For the matrix of Example 4.1, j could be either 10, 11 or 12. Define $J_1 = \{|J|+1, |J|+2, \ldots, |J|+k_2-1\}$, and $J_2 = J^c - (J_1 \cup \{j\})$. Then, $\mathcal{C}_1 = (\mathcal{C}_1 \oplus_2 \mathcal{C}_2) \setminus J_1/J_2$. To obtain $\mathcal{C}_2$ as a minor, let j' be the index of a row of $\overline{G}$ in which the submatrix B has a nonzero row. For the matrix of Example 4.1, j' could be either 1, 2 or 3. The j'th column of $\overline{G}$ is of the form $[0 \ldots 0\ 1\ 0 \ldots 0]^t$, the single 1 being the j'th entry. Define $J_1' = \{1, 2, \ldots, k_1\} - \{j'\}$ and $J_2' = \{k_1+1, k_1+2, \ldots, |J|\}$. Then, $\mathcal{C}_2 = (\mathcal{C}_1 \oplus_2 \mathcal{C}_2) \setminus J_1'/J_2'$. Proofs of these statements just involve consistency checking, so are left as an easy exercise.

In summary, the procedure described above takes as input a $k \times n$ generator matrix G for $\mathcal{C}$, and an exact 2-separation (J, J^c) of it, and produces as output a permutation π of the coordinates of $\mathcal{C}$, and the generator matrices of two codes $\mathcal{C}_1$ and $\mathcal{C}_2$, such that $\mathcal{C} = \pi(\mathcal{C}_1 \oplus_2 \mathcal{C}_2)$. The codes $\mathcal{C}_1$ and $\mathcal{C}_2$ are both equivalent to proper minors of $\mathcal{C}$. The entire procedure can be carried out in $O(k^2 n)$ time, which is the run-time complexity of bringing a $k \times n$ matrix to reduced row-echelon form via elementary row operations. All other parts of the procedure can be performed in $O(n)$ time; for example, since (J, J^c) is given, it only takes $O(n)$ time to find the permutation, π^{-1}, that takes $\mathsf{rref}(G)$ to the matrix $\overline{G}$ in (7).

A straightforward combination of Lemma 3.1 and Corollary 4.5 yields the following theorem, which illustrates the utility of the matroid-theoretic tools presented so far.

Theorem 4.6 ([18], Corollary 8.3.4)**.** *Every code that is not 3-connected can be constructed from 3-connected proper minors of it by a sequence of operations of coordinate permutation, direct sum and 2-sum.*

The decomposition of a code via direct sums and 2-sums implicit in the above theorem can be carried out in time polynomial in the length of the code. This is due to the following two facts:

(a) as described in Section 3, there are polynomial-time algorithms for finding 1- and 2-separations in a code, if they exist; and

(b) given an exact 2-separation of a code $\mathcal{C}$, there is a polynomial-time procedure that produces codes $\mathcal{C}_1$ and $\mathcal{C}_2$, both equivalent to proper minors of $\mathcal{C}$, and a permutation π of the coordinate set of $\mathcal{C}$, such that $\mathcal{C} = \pi(\mathcal{C}_1 \oplus_2 \mathcal{C}_2)$.

However, direct sums and 2-sums are not enough for our purposes, nor were they enough for Seymour's theory of matroid decomposition. Seymour also had to extend the graph-theoretic technique of 3-clique-sum to matroids (in fact, to binary matroids only). The corresponding operation on binary matroids is called 3-sum.

4.2. **3-Sums.**

The special case of the $\mathcal{S}_3(\mathcal{C}, \mathcal{C}')$ construction called 3-sum is somewhat more complex in definition than the 2-sum. Recall that for a binary word $\mathbf{x}$, $w(\mathbf{x})$ denotes its Hamming weight.

Definition 4.2. *Let $\mathcal{C}, \mathcal{C}'$ be codes of length at least seven, such that*

(A1) *no codeword of $\mathcal{C}$ or $\mathcal{C}^\perp$ is of the form $0 \ldots 0\,\mathbf{x}$, where $\mathbf{x}$ is a length-3 word with $w(\mathbf{x}) \in \{1, 2\}$;*

(A2) *no codeword of $\mathcal{C}'$ or $\mathcal{C}'^\perp$ is of the form $\mathbf{x}\,0 \ldots 0$, where $\mathbf{x}$ is a length-3 word with $w(\mathbf{x}) \in \{1, 2\}$; and*

(A3) $0\dots0111 \in \mathcal{C}$ *and* $1110\dots0 \in \mathcal{C}'$.

Then, $\mathcal{S}_3(\mathcal{C},\mathcal{C}')$ *is called the* 3-sum *of* $\mathcal{C}$ *and* $\mathcal{C}'$*, and is denoted by* $\mathcal{C} \oplus_3 \mathcal{C}'$.

It is perhaps worth commenting upon the use of the terms "2-sum" and "3-sum" to denote codes of the form $\mathcal{S}_m(\mathcal{C},\mathcal{C}')$ for $m = 1$ and $m = 3$, respectively. The nomenclature stems from the analogy with the k-clique-sum of graphs, wherein two graphs are glued along a k-clique. Note that a 2-clique is a single edge (hence $m = 1$) and 3-clique is a triangle of three edges (hence $m = 3$). This also explains why we do not consider an operation of the form $\mathcal{S}_2(\mathcal{C},\mathcal{C}')$.

3-sums are only defined for codes having the properties (A1)–(A3) listed in the above definition. It is obvious that an equivalent statement of (A1)–(A3) is the following:

(B1) $0\dots0111$ is a minimal codeword of $\mathcal{C}$, and $\mathcal{C}^\perp$ has no nonzero codeword supported entirely within the last three coordinates of $\mathcal{C}^\perp$; and

(B2) $1110\dots0$ is a minimal codeword of $\mathcal{C}'$, and $\mathcal{C}'^\perp$ has no nonzero codeword supported entirely within the first three coordinates of $\mathcal{C}'^\perp$.

In fact, (B1) and (B2) above are exact translations of the matroid-theoretic language used by Seymour in his definition of 3-sum [6]. Another equivalent way of expressing these conditions is the following:

(B1′) $0\dots0111$ is a minimal codeword of $\mathcal{C}$, and the restriction of $\mathcal{C}$ onto its last three coordinates is $\{0,1\}^3$; and

(B2′) $1110\dots0$ is a minimal codeword of $\mathcal{C}'$, and the restriction of $\mathcal{C}'$ onto its first three coordinates is $\{0,1\}^3$.

The equivalence of (B1) and (B1′) is a consequence of the easily verifiable fact that if $0\dots0111 \in \mathcal{C}$, then $\mathcal{C}^\perp$ has no nonzero codeword supported entirely within the last three coordinates of $\mathcal{C}^\perp$ if and only if all possible 3-bit words appear in the last three coordinates of $\mathcal{C}$. The equivalence of (B2) and (B2′) is analogous. It follows immediately from (B1′) and (B2′) that $\mathcal{C}\oplus_3\mathcal{C}'$ can be defined only if $\min\{\dim(\mathcal{C}), \dim(\mathcal{C}')\} \geq 3$ and $\max\{d(\mathcal{C}), d(\mathcal{C}')\} \leq 3$.

Example 4.3. *Let* $\mathcal{C}$ *and* $\mathcal{C}'$ *be the [7,4] Hamming codes given by the generator matrices* G *and* G'*, respectively, below.*

$$G = \begin{bmatrix} 1&1&0&0&0&0&1\\ 1&0&1&0&0&1&0\\ 0&1&1&0&1&0&0\\ 1&1&1&1&0&0&0 \end{bmatrix}, \quad G' = \begin{bmatrix} 1&0&0&0&0&1&1\\ 0&1&0&0&1&0&1\\ 0&0&1&0&1&1&0\\ 0&0&0&1&1&1&1 \end{bmatrix}.$$

$\mathcal{C}$ *and* $\mathcal{C}'$ *satisfy the conditions in Definition 4.2, so their 3-sum can be defined. The code* $\mathcal{C} \oplus_3 \mathcal{C}'$ *works out to be the* $[8,4,4]$ *extended Hamming code with generator matrix*

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

The minimal trellis of this code is shown in Figure 2.

With $J = \{1,2,3,4\}$, (J, J^c) *is an exact 3-separation of* $\mathcal{C} \oplus_3 \mathcal{C}'$*. It may be verified that, in fact,* $\lambda(\mathcal{C} \oplus_3 \mathcal{C}') = 3$*. Furthermore, puncturing any coordinate of* $\mathcal{C} \oplus_3 \mathcal{C}'$ *yields a* $[7,4]$ *Hamming code. Thus,* $\mathcal{C}$ *and* $\mathcal{C}'$ *are (up to code equivalence) minors of* $\mathcal{C} \oplus_3 \mathcal{C}'$.

The dimension and minimum distance of $\mathcal{C}\oplus_3\mathcal{C}'$ can be related to $\mathcal{C}$ and $\mathcal{C}'$ in a manner analogous to Proposition 4.1 for 2-sums.

Proposition 4.7. *Let* $\mathcal{C}$ *and* $\mathcal{C}'$ *be codes of length* n *and* n'*, respectively, for which* $\mathcal{C}\oplus_3\mathcal{C}'$ *can be defined.*

(a) $\dim(\mathcal{C}\oplus_3\mathcal{C}') = \dim(\mathcal{C}) + \dim(\mathcal{C}') - 4$.

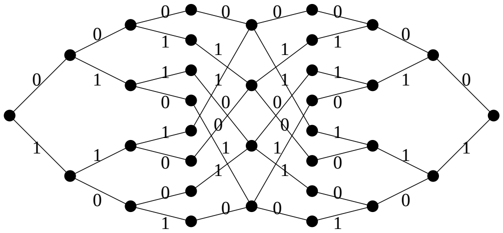

FIGURE 2. The minimal trellis of the code $\mathcal{C} \oplus_3 \mathcal{C}'$ of Example 4.3.

(b) *If* $\dim(\mathcal{C}) > 3$*, then* $d(\mathcal{C}\oplus_3\mathcal{C}') \leq d(\mathcal{C} \setminus \{n-2, n-1, n\})$*; and if* $\dim(\mathcal{C}') > 3$*, then* $d(\mathcal{C}\oplus_3\mathcal{C}') \leq d(\mathcal{C}' \setminus \{1, 2, 3\})$.

Proof. We will only prove (a) as the proof of (b) is analogous to the relevant part of the proof of Proposition 4.1(b). Let $\mathbf{x} \parallel \mathbf{x}'$ denote the concatenation of binary sequences $\mathbf{x}$ and $\mathbf{x}'$.

We prove the proposition by first showing that $\dim(\mathcal{C} \parallel_3 \mathcal{C}') = \dim(\mathcal{C}) + \dim(\mathcal{C}') - 1$, and then showing that $\dim(\mathcal{C} \oplus_3 \mathcal{C}') = \dim(\mathcal{C} \parallel_3 \mathcal{C}') - 3$. The first of these equalities follows directly from the observation that the mapping

$$\phi(\mathbf{c} \parallel \mathbf{c}') = \mathbf{c} \parallel_3 \mathbf{c}', \quad \mathbf{c} \in \mathcal{C}, \mathbf{c}' \in \mathcal{C}',$$

defines a homomorphism from the direct sum $\mathcal{C} \oplus \mathcal{C}'$ onto $\mathcal{C} \parallel_3 \mathcal{C}'$, with $\dim(\ker(\phi)) = 1$; indeed, $\ker(\phi)$ consists of the two words **0** and $0 \ldots 0111 \parallel 1110 \ldots 0$.

To prove that $\dim(\mathcal{C} \oplus_3 \mathcal{C}') = \dim(\mathcal{C} \parallel_3 \mathcal{C}') - 3$, we observe that $\mathcal{C} \oplus_3 \mathcal{C}'$ is isomorphic to the subcode, $\mathcal{E}$, of $\mathcal{C} \parallel_3 \mathcal{C}'$ consisting of those codewords $\hat{c}_1\hat{c}_2 \ldots \hat{c}_{n+n'-3}$ such that $\hat{c}_{n-2}\hat{c}_{n-1}\hat{c}_n = 000$. Therefore, $\dim(\mathcal{C} \oplus_3 \mathcal{C}') = \dim(\mathcal{E})$.

Since the restriction of $\mathcal{C}$ onto its last three coordinates is $\{0,1\}^3$ (property (B1′)), and the restriction of $\mathcal{C}'$ onto its first three coordinates is $\{0,1\}^3$ (property (B2′)), the restriction of $\mathcal{C} \parallel_3 \mathcal{C}'$ onto its $(n-2)$th, $(n-1)$th and nth coordinates is also $\{0,1\}^3$. Therefore, $\dim(\mathcal{E}) = \dim(\mathcal{C} \parallel_3 \mathcal{C}') - 3$, and hence, $\dim(\mathcal{C} \oplus_3 \mathcal{C}') = \dim(\mathcal{C} \parallel_3 \mathcal{C}') - 3$, which completes the proof of the proposition. □

We remark that the case of $\dim(\mathcal{C}) = \dim(\mathcal{C}') = 3$ has been deliberately left out from the statement of Proposition 4.7(b). In this case, there is no useful expression or upper bound for the minimum distance of $\mathcal{C} \oplus_3 \mathcal{C}'$. Since $\dim(\mathcal{C} \oplus_3 \mathcal{C}') = 2$ in this case, one may just as well identify the codeword of least weight among the three nonzero codewords in the code.

An important difference between 2-sums and 3-sums is that the result of Proposition 4.2 does not directly extend to 3-sums. The reason for this is that for codes $\mathcal{C}$ and $\mathcal{C}'$ satisfying (A1)–(A3) in Definition 4.2, the 3-sum $\mathcal{C}^\perp\oplus_3\mathcal{C}'^\perp$ cannot even be defined. Indeed, while (A1) and (A2) are invariant under the operation of taking duals, (A3) is not — if $0 \ldots 0111 \in \mathcal{C}$, then $0 \ldots 0111 \notin \mathcal{C}^\perp$. To determine the dual of a 3-sum, we need to define a "dual" operation, namely the $\overline{3}$-sum.

Definition 4.3. *Let* $\mathcal{C}, \mathcal{C}'$ *be codes of length at least seven, such that*

(A1′) *no codeword of* $\mathcal{C}$ *or* $\mathcal{C}^\perp$ *is of the form* $0 \ldots 0\,\mathbf{x}$*, where* $\mathbf{x}$ *is a length-3 word with* $w(\mathbf{x}) \in \{1, 2\}$*;*

(A2′) *no codeword of* $\mathcal{C}'$ *or* $\mathcal{C}'^\perp$ *is of the form* $\mathbf{x}\,0 \ldots 0$*, where* $\mathbf{x}$ *is a length-3 word with* $w(\mathbf{x}) \in \{1, 2\}$*; and*

(A3′) $0 \ldots 0111 \in \mathcal{C}^\perp$ *and* $1110 \ldots 0 \in \mathcal{C}'^\perp$.

The $\overline{3}$-sum *of* $\mathcal{C}$ *and* $\mathcal{C}'$*, denoted by* $\mathcal{C} \mathbin{\overline{\oplus}_3} \mathcal{C}'$*, is defined as*

$$\mathcal{C} \mathbin{\overline{\oplus}_3} \mathcal{C}' = \overline{\mathcal{C}} \oplus_3 \overline{\mathcal{C}'},$$

where $\overline{\mathcal{C}} = \mathcal{C} \bigcup (0 \ldots 0111 + \mathcal{C})$ *and* $\overline{\mathcal{C}'} = \mathcal{C}' \bigcup (1110 \ldots 0 + \mathcal{C}')$.

Note that (A1$'$) and (A2$'$) are identical to (A1) and (A2), respectively. To ensure that the above definition can in fact be made, it must be verified that the 3-sum $\overline{\mathcal{C}} \oplus_3 \overline{\mathcal{C}'}$ can actually be defined for codes $\mathcal{C}$ and $\mathcal{C}'$ satisfying (A1$'$)–(A3$'$). So, let $\mathcal{C}$ and $\mathcal{C}'$ be codes satisfying (A1$'$)–(A3$'$). We need to verify that $\overline{\mathcal{C}}$ and $\overline{\mathcal{C}'}$ satisfy (A1)–(A3) in Definition 4.2.

By their very definition, $\overline{\mathcal{C}}$ and $\overline{\mathcal{C}'}$ satisfy (A3). Furthermore, since $\overline{\mathcal{C}}^{\perp} \subset \mathcal{C}^{\perp}$ and $\overline{\mathcal{C}'}^{\perp} \subset \mathcal{C}'^{\perp}$, we see that no codeword of $\overline{\mathcal{C}}^{\perp}$ is of the form $0\ldots0\,\mathbf{x}$ as in (A1), and no codeword of $\overline{\mathcal{C}'}^{\perp}$ is of the form $\mathbf{x}\,0\ldots0$ as in (A2). Finally, if $0\ldots0\,\mathbf{x} \in \overline{\mathcal{C}}$ for some length-3 word $\mathbf{x}$ with $w(\mathbf{x}) \in \{1,2\}$, then since $0\ldots0\,\mathbf{x} \notin \mathcal{C}$ by (A1$'$), it must be that $0\ldots0\,\mathbf{x}$ is in $0\ldots0111 + \mathcal{C}$. But in this case, $0\ldots0\,\overline{\mathbf{x}} \in \mathcal{C}$, where $\overline{\mathbf{x}} = 111 + \mathbf{x}$. So, $w(\overline{\mathbf{x}}) \in \{1,2\}$ as well, which is impossible by (A1$'$). Therefore, $\overline{\mathcal{C}}$ cannot contain any word of the form $0\ldots0\,\mathbf{x}$ as in (A1). By analogous reasoning, $\overline{\mathcal{C}'}$ cannot contain any word of the form $\mathbf{x}\,0\ldots0$ as in (A2). We have thus verified that $\overline{\mathcal{C}}$ and $\overline{\mathcal{C}'}$ satisfy (A1)–(A3), and so $\overline{\mathcal{C}} \oplus_3 \overline{\mathcal{C}'}$ can be defined.

Note that (A3$'$) implies that $0\ldots0111 \notin \mathcal{C}$ and $1110\ldots0 \notin \mathcal{C}'$, and hence, $\dim(\overline{\mathcal{C}}) = \dim(\mathcal{C})+1$ and $\dim(\overline{\mathcal{C}'}) = \dim(\mathcal{C}')+1$. Furthermore, letting n denote the length of $\mathcal{C}$, we have $\overline{\mathcal{C}} \setminus \{n-2, n-1, n\} = \mathcal{C} \setminus \{n-2, n-1, n\}$ and $\overline{\mathcal{C}'} \setminus \{1,2,3\} = \mathcal{C}' \setminus \{1,2,3\}$. Therefore, by virtue of Proposition 4.7, we have the following result.

Proposition 4.8. *Let $\mathcal{C}$ and $\mathcal{C}'$ be codes of length n and n', respectively, for which $\mathcal{C} \mathbin{\overline{\oplus}_3} \mathcal{C}'$ can be defined.*

(a) $\dim(\mathcal{C} \mathbin{\overline{\oplus}_3} \mathcal{C}') = \dim(\mathcal{C}) + \dim(\mathcal{C}') - 2$.

(b) *If* $\dim(\mathcal{C}) > 2$, *then* $d(\mathcal{C} \mathbin{\overline{\oplus}_3} \mathcal{C}') \le d(\mathcal{C} \setminus \{n-2, n-1, n\})$; *and if* $\dim(\mathcal{C}') > 2$, *then* $d(\mathcal{C} \mathbin{\overline{\oplus}_3} \mathcal{C}') \le d(\mathcal{C}' \setminus \{1,2,3\})$.

The $\overline{3}$-sum is the dual operation to 3-sum, in a sense made precise by the proposition below, the proof of which we defer to Appendix A. This result is stated, without explicit proof, in [8, p. 316] and [26, p. 184]; Truemper [8, 26] refers to 3-sum and $\overline{3}$-sum as Δ-sum and Y-sum, respectively. The result can also be derived from Theorem 7.3 in [35].

Proposition 4.9. *For codes $\mathcal{C}$ and $\mathcal{C}'$ be codes for which $\mathcal{C} \oplus_3 \mathcal{C}'$ can be defined, we have*

$$(\mathcal{C} \oplus_3 \mathcal{C}')^{\perp} = \mathcal{C}^{\perp} \mathbin{\overline{\oplus}_3} \mathcal{C}'^{\perp}.$$

It follows from the last result that a code that is expressible as a 3-sum can also be expressed as a $\overline{3}$-sum. Indeed, if $\mathcal{C} = \mathcal{C}_1 \oplus_3 \mathcal{C}_2$, then by the above proposition, $\mathcal{C}^{\perp} = \mathcal{C}_1^{\perp} \mathbin{\overline{\oplus}_3} \mathcal{C}_2^{\perp}$. The latter, by definition, is $\overline{\mathcal{C}_1^{\perp}} \oplus_3 \overline{\mathcal{C}_2^{\perp}}$, and so again taking duals and using the above proposition, we obtain

$$\mathcal{C} = \left(\overline{\mathcal{C}_1^{\perp}} \oplus_3 \overline{\mathcal{C}_2^{\perp}}\right)^{\perp} = \left(\overline{\mathcal{C}_1^{\perp}}\right)^{\perp} \mathbin{\overline{\oplus}_3} \left(\overline{\mathcal{C}_2^{\perp}}\right)^{\perp}.$$

We record this as a corollary to Proposition 4.9.

Corollary 4.10. *If $\mathcal{C}_1 \oplus_3 \mathcal{C}_2$ can be defined, then*

$$\mathcal{C}_1 \oplus_3 \mathcal{C}_2 = \left(\overline{\mathcal{C}_1^{\perp}}\right)^{\perp} \mathbin{\overline{\oplus}_3} \left(\overline{\mathcal{C}_2^{\perp}}\right)^{\perp}.$$

Having presented some of the simpler properties of 3-sums, we next state a highly non-trivial result of Seymour that illustrates how 3-sums are to be used. The statement of this result is the 3-sum analogue of Theorem 4.3, but there are some important differences between the two that we will point out after stating the result.

Theorem 4.11 ([6],Theorems 2.9 and 4.1)**.** *If $\mathcal{C}_1$ and $\mathcal{C}_2$ are codes of length n_1 and n_2, respectively, such that $\mathcal{C} = \mathcal{C}_1 \oplus_3 \mathcal{C}_2$, then (J, J^c) is an exact 3-separation of $\mathcal{C}$ for $J = \{1, 2, \ldots, n_1 - 3\}$. Furthermore, if $\mathcal{C}$ is 3-connected, then $\mathcal{C}_1$ and $\mathcal{C}_2$ are equivalent to proper minors of $\mathcal{C}$.*

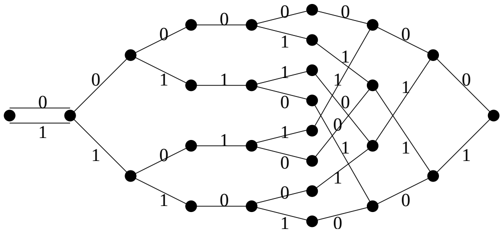

FIGURE 3. The minimal trellis of the code $\mathcal{C} = \mathcal{C}_1 \oplus_3 \mathcal{C}_2$ of Example 4.4.

Conversely, if (J, J^c) is an exact 3-separation of a code $\mathcal{C}$, with $\min\{|J|, |J^c|\} \geq 4$, then there are codes $\mathcal{C}_1$ and $\mathcal{C}_2$ of length $|J|+3$ and $|J^c|+3$, respectively, such that $\mathcal{C}$ is equivalent to $\mathcal{C}_1 \oplus_3 \mathcal{C}_2$.

A couple of key differences between the statements of Theorems 4.3 and 4.11 must be stressed. For a code to be expressible as a 2-sum, it is sufficient that there exist an exact 2-separation. However, to make the analogous conclusion about 3-sums, Theorem 4.11 not only asks for the existence of an exact 3-separation (J, J^c), but also adds the additional hypothesis that $\min\{|J|, |J^c|\} \geq 4$. We will have more to say about this a little later.

There is a second major difference between the statements of the two theorems. Theorem 4.3 states that if $\mathcal{C} = \mathcal{C}_1 \oplus_2 \mathcal{C}_2$, then $\mathcal{C}_1$ and $\mathcal{C}_2$ are always minors of $\mathcal{C}$, up to coordinate permutation. However, when $\mathcal{C} = \mathcal{C}_1 \oplus_3 \mathcal{C}_2$, Theorem 4.11 imposes the condition that $\mathcal{C}$ be 3-connected in order to conclude that $\mathcal{C}_1$ and $\mathcal{C}_2$ are equivalent to minors of $\mathcal{C}$. If the 3-connectedness requirement for $\mathcal{C}$ is dropped, the conclusion does not hold in general, as the following example shows.

Example 4.4. *Take $\mathcal{C}_1$ to be the $[7, 4, 1]$ code with generator matrix*

$$G_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \end{bmatrix},$$

and let $\mathcal{C}_2$ be the $[7, 4, 3]$ Hamming code with generator matrix

$$G_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

These codes satisfy (A1)–(A3) of Definition 4.2, and their 3-sum, $\mathcal{C}_1 \oplus_3 \mathcal{C}_2$, is the $[8, 4, 1]$ code $\mathcal{C}$ generated by

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

The minimal trellis of this code is shown in Figure 3. Note that $\mathcal{C}$ is not 3-connected. In fact, it is not even 2-connected — it has a 1-separation (J, J^c) with $J = \{1\}$. Now, $\mathcal{C}_1$ can be obtained as a minor of $\mathcal{C}$ by puncturing $\mathcal{C}$ at the last coordinate. However, $\mathcal{C}_2$ is not a minor of $\mathcal{C}$, since puncturing $\mathcal{C}$ at any coordinate does not yield a $[7, 4, 3]$ code, and shortening always yields a code of dimension less than 4.

We mention in passing that if $\mathcal{C} = \mathcal{C}_1 \oplus_3 \mathcal{C}_2$, then the fact that $\mathcal{C}_1$ and $\mathcal{C}_2$ are equivalent to minors of $\mathcal{C}$ whenever $\mathcal{C}$ is 3-connected is far more difficult to prove (see [6, Section 4]) than the corresponding part of Theorem 4.3.

We observed above that the mere existence of an exact 3-separation is not enough for Theorem 4.11 to conclude that a code $\mathcal{C}$ is expressible as a 3-sum; the 3-separation (J, J^c) must also satisfy $\min\{|J|, |J^c|\} \geq 4$, *i.e.*, must not be minimal[5]. It is also implicit in the statement of Theorem 4.11 that the existence of a non-minimal 3-separation is a necessary condition for a code to be a 3-sum. Indeed, if $\mathcal{C} = \mathcal{C}_1 \oplus_3 \mathcal{C}_2$, then $\mathcal{C}_1$ and $\mathcal{C}_2$ must each have length at least 7, as per Definition 4.2. So, with $J = \{1, 2, \ldots, n_1 - 3\}$ as in the statement of Theorem 4.11, it must be true that $|J| \geq 4$, and similarly, $|J^c| \geq 4$.

The following definition allows for a compact statement of a corollary to Theorem 4.11 along the lines of Corollary 4.5.

Definition 4.4. *A 3-connected code is* internally 4-connected *if all its 3-separations are minimal.*

Internal 4-connectedness is a notion that lies properly between 3-connectedness and 4-connectedness — a 3-connected code that is not 4-connected can be internally 4-connected. Note that any 3-separation in a 3-connected code must be exact, and so we can state the following corollary to Theorem 4.11.

Corollary 4.12. *A 3-connected code $\mathcal{C}$ is not internally 4-connected iff there exist codes $\mathcal{C}_1$ and $\mathcal{C}_2$, both equivalent to proper minors of $\mathcal{C}$, such that $\mathcal{C}$ is equivalent to $\mathcal{C}_1 \oplus_3 \mathcal{C}_2$.*

As in the case of 2-sums, we provide an efficient construction of the components of the 3-sum when an exact, non-minimal 3-separation (J, J^c) of $\mathcal{C}$ is given. This construction furnishes a proof of the converse part of Theorem 4.11. Our description is based loosely on the constructions given in [26, Section 8.3] and [8].

Let $\mathcal{C}$ be a code of length n and dimension k, specified by a $k \times n$ generator matrix G, and let (J, J^c) be an exact 3-separation of $\mathcal{C}$, with $|J| \geq 4$ and $|J^c| \geq 4$. By permuting coordinates if necessary, we may assume that $J = \{1, 2, \ldots, m\}$ for some m such that $4 \leq m \leq n - 4$. Let $G|_J$ and $G|_{J^c}$ denote the restrictions of G to the columns indexed by J and J^c, respectively; thus, $G = [G|_J \;\, G|_{J^c}]$. Let $\mathsf{rank}(G|_J) = k_1$ and $\mathsf{rank}(G|_{J^c}) = k_2$; since (J, J^c) is an exact 3-separation of $\mathcal{C}$, we have $k_1 + k_2 = k + 2$. Note that $k_1 \leq k$ implies that $k + k_2 \geq k_1 + k_2 = k + 2$, so that $k_2 \geq 2$; similarly, $k_1 \geq 2$.

Bring G into reduced row-echelon form (rref) over $\mathbb{F}_2$. Permuting coordinates within J and within J^c if necessary, $\mathsf{rref}(G)$ may be assumed to be of the form

$$\overline{G} = \begin{bmatrix} I_{k_1} & A & \mathbf{O} & B \\ \mathbf{O} & \mathbf{O} & I_{k_2-2} & C \end{bmatrix}, \tag{8}$$

where I_j, for $j = k_1, k_2 - 2$, denotes the $j \times j$ identity matrix, A is a $k_1 \times (|J| - k_1)$ matrix, B is a $k_1 \times (|J^c| - k_2 + 2)$ matrix, C is a $(k_2 - 2) \times (|J^c| - k_2 + 2)$ matrix, and the $\mathbf{O}$'s denote all-zeros matrices of appropriate sizes. As a concrete example, consider the matrix $\overline{G}$ given in Example 4.3, which is indeed of the above form, with $|J| = |J^c| = 4$, $k_1 = k_2 = 3$,

$$A = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \quad \text{and} \quad C = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}.$$

The fact that the submatrix $\begin{bmatrix} \mathbf{O} & B \\ I_{k_2-2} & C \end{bmatrix}$ must have rank equal to $\mathsf{rank}(G|_{J^c}) = k_2$ implies that B must have rank 2. Hence, B has two linearly independent rows, call them $\mathbf{x}$ and $\mathbf{y}$, which form a basis of the row-space of B. In particular, each row of B is either $\mathbf{0}$, $\mathbf{x}$, $\mathbf{y}$ or $\mathbf{x} + \mathbf{y}$.

Now, define the $(k_1 + 1) \times (|J| + 3)$ matrix

$$G_1 = \begin{bmatrix} I_{k_1} & A & D \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix},$$

[5] The definition of a minimal k-separation is given immediately after Definition 3.1.

where D is a $k_1 \times 3$ matrix whose ith row is defined as

$$i\text{th row of } D = \begin{cases} 000 & \text{if } i\text{th row of } B \text{ is } \mathbf{0} \\ 001 & \text{if } i\text{th row of } B \text{ is } \mathbf{x} \\ 010 & \text{if } i\text{th row of } B \text{ is } \mathbf{y} \\ 100 & \text{if } i\text{th row of } B \text{ is } \mathbf{x}+\mathbf{y}; \end{cases}$$

and the bottom row of G_1, represented by $[\mathbf{0}\ \mathbf{0}\ \mathbf{1}]$, is simply $0\ldots0111$.

Next, define the $(k_2+1) \times (|J^c|+3)$ matrix

$$G_2 = \left[\begin{array}{ccc} I_3 & \mathbf{O} & X \\ \mathbf{O} & I_{k_2-2} & C \end{array}\right] = [I_{k_2+1}\, C''],$$

where

$$X = \left[\begin{array}{c} \mathbf{x}+\mathbf{y} \\ \mathbf{y} \\ \mathbf{x} \end{array}\right] \quad \text{and} \quad C'' = \left[\begin{array}{c} X \\ C \end{array}\right].$$

A straightforward verification yields that if $\mathcal{C}_1$ and $\mathcal{C}_2$ are the codes generated by G_1 and G_2, respectively, then $\dim(\mathcal{C}_1) = k_1+1$, $\dim(\mathcal{C}_2) = k_2+1$, and $\mathcal{C}_1, \mathcal{C}_2$ satisfy properties (A1)–(A3) in Definition 4.2, so $\mathcal{C}_1 \oplus_3 \mathcal{C}_2$ can be defined. The construction of G_1 and G_2 above is carefully crafted to ensure that all the rows of $\overline{G}$ are in $\mathcal{C}_1 \oplus_3 \mathcal{C}_2$. Hence, $\dim(\mathcal{C}_1 \oplus_3 \mathcal{C}_2) \geq \mathsf{rank}(\overline{G}) = k_1+k_2-2$. However, by Proposition 4.7, we have that $\dim(\mathcal{C}_1 \oplus_3 \mathcal{C}_2) = \dim(\mathcal{C}_1) + \dim(\mathcal{C}_2) - 4 = k_1+k_2-2$. Hence, $\dim(\mathcal{C}_1 \oplus_3 \mathcal{C}_2) = \mathsf{rank}(\overline{G})$, implying that $\overline{G}$ must be a generator matrix for $\mathcal{C}_1 \oplus_3 \mathcal{C}_2$. Note that according to Theorem 4.11, if $\mathcal{C}$ is 3-connected, then $\mathcal{C}_1$ and $\mathcal{C}_2$ are equivalent to proper minors of $\mathcal{C}$.

The procedure described above can be formalized into an algorithm that takes as input a $k \times n$ generator matrix G for $\mathcal{C}$, and an exact, non-minimal 3-separation (J, J^c) of it, and produces as output a permutation π of the coordinates of $\mathcal{C}$, and the generator matrices of two codes $\mathcal{C}_1$ and $\mathcal{C}_2$, such that $\mathcal{C} = \pi(\mathcal{C}_1 \oplus_3 \mathcal{C}_2)$. The codes $\mathcal{C}_1$ and $\mathcal{C}_2$ are both equivalent to proper minors of $\mathcal{C}$. This procedure can be carried out in $O(k^2 n)$ time for the same reasons as in the 2-sum case.

For the matrix $\overline{G}$ in Example 4.3, we find the matrices G_1 and G_2 to be

$$G_1 = \left[\begin{array}{ccccccc} 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{array}\right], \quad \text{and} \quad G_2 = \left[\begin{array}{ccccccc} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}\right],$$

which are indeed generator matrices of the two Hamming codes whose 3-sum is represented by $\overline{G}$.

It must be pointed out that Theorem 4.11 also holds when the 3-sums in its statement are replaced by $\overline{3}$-sums. This is a consequence of Proposition 3.2, which shows that (J, J^c) is a 3-separation of a code $\mathcal{C}$ iff it is a 3-separation of the dual code $\mathcal{C}^\perp$. Hence, applying Theorem 4.11 to $\mathcal{C}^\perp$, and dualizing via Proposition 4.9, we see that the 3-sums in the statement of Theorem 4.11 can be replaced by $\overline{3}$-sums. In particular, we also have the following corollary to Theorem 4.11.

Corollary 4.13. *A 3-connected code $\mathcal{C}$ is not internally 4-connected iff there exist codes $\mathcal{C}_1$ and $\mathcal{C}_2$, both equivalent to proper minors of $\mathcal{C}$, such that $\mathcal{C}$ is equivalent to $\mathcal{C}_1 \overline{\oplus}_3 \mathcal{C}_2$.*

Putting together Theorem 4.6 with 4.12 and 4.13, we obtain the following theorem, which summarizes the code decomposition theory presented up to this point.

Theorem 4.14. *A binary linear code either is 3-connected and internally 4-connected, or can be constructed from 3-connected, internally 4-connected proper minors of it by a sequence of operations of coordinate permutation, direct sum, 2-sum and 3-sum (or $\overline{3}$-sum).*

The decomposition of Theorem 4.14 can be carried out in time polynomial in the length of the code, since

(a) the decomposition of Theorem 4.6 can be carried out in polynomial time;
(b) as mentioned in Section 3, there are polynomial-time algorithms for finding non-minimal 3-separations (*i.e.*, 3-separations (J, J^c) with $\min\{|J|, |J^c|\} \geq 4$) in a code, if they exist; and
(c) there is a polynomial-time procedure that, given an exact, non-minimal 3-separation of a 3-connected code $\mathcal{C}$, produces codes $\mathcal{C}_1$ and $\mathcal{C}_2$, both equivalent to proper minors of $\mathcal{C}$, and a permutation π of the coordinate set of $\mathcal{C}$, such that $\mathcal{C} = \pi(\mathcal{C}_1 \oplus_3 \mathcal{C}_2)$ or, via Corollary 4.10, $\mathcal{C} = \pi(\mathcal{C}_1 \,\overline{\oplus}_3\, \mathcal{C}_2)$.

Note that the theorem does not guarantee uniqueness of the code decomposition.

4.3. **Code-Decomposition Trees.** A binary tree is a convenient data structure for storing a decomposition of a code via direct sums, 2-sums, and 3-sums. Recall that a proper (or full) binary tree is a rooted tree such that every node of the tree has either zero or two children. We will drop the adjective "proper" as proper binary trees are the only kind of binary trees we are interested in. A node without any children is called a *leaf*. Each non-leaf node has two children, and we will distinguish between the two, calling one the *left* child and the other the *right* child.

Let $\mathcal{C}$ be a binary linear code. A *code-decomposition tree* for $\mathcal{C}$ is a binary tree $\mathcal{T}$ defined as follows. Each node v of $\mathcal{T}$ stores a triple ($\mathsf{v.code}$,$\mathsf{v.perm}$,$\mathsf{v.sum}$), where $\mathsf{v.code}$ is a binary linear code, $\mathsf{v.perm}$ is either NULL or a permutation of the coordinate set of $\mathsf{v.code}$, and $\mathsf{v.sum}$ $\in \{\odot, \oplus, \oplus_2, \oplus_3, \overline{\oplus}_3\}$. For each node v of $\mathcal{T}$, the triple ($\mathsf{v.code}$,$\mathsf{v.perm}$,$\mathsf{v.sum}$) must adhere to the following rules:

(R1) if v is the root node, then $\mathsf{v.code}$ is the code $\mathcal{C}$ itself;
(R2) $\mathsf{v.perm} =$ NULL iff v is a leaf;
(R3) $\mathsf{v.sum} = \odot$ iff v is a leaf;
(R4) if v is a non-leaf node, then
  (i) $\mathsf{lchild.code}$ and $\mathsf{rchild.code}$ are proper minors of $\mathsf{v.code}$; and
  (ii) the permutation $\mathsf{v.perm}$ applied to the sum

$$(\mathsf{lchild.code})\ \ \mathsf{v.sum}\ \ (\mathsf{rchild.code})$$

  yields $\mathsf{v.code}$,

where lchild and rchild above respectively refer to the left and right children of v.

In particular, (R4) ensures that for any node v other than the root node in the tree, $\mathsf{v.code}$ is a proper minor of $\mathcal{C}$.

We will identify the leaves of any code-decomposition tree with the codes that they store. Theorem 4.14 guarantees that each code $\mathcal{C}$ has a code-decomposition tree in which each leaf is 3-connected and internally 4-connected. Such a code-decomposition tree will be called *complete*. A complete code-decomposition tree for $\mathcal{C}$ can be constructed in time polynomial in the length of the code $\mathcal{C}$, since the decomposition of Theorem 4.14 can be carried out in polynomial time. Note that if $\mathcal{C}$ itself is 3-connected and internally 4-connected, then there is exactly one code-decomposition tree for it, which is the tree consisting of the single node $\mathcal{C}$ — or more precisely, the single node that stores $(\mathcal{C}, \text{NULL}, \odot)$.

Finally, a code-decomposition tree is called *3-homogeneous* (resp. $\overline{3}$-*homogeneous*) if for each non-leaf node v in the tree, if $\mathsf{v.sum} \in \{\oplus, \oplus_2, \oplus_3\}$ (resp. $\mathsf{v.sum} \in \{\oplus, \oplus_2, \overline{\oplus}_3\}$). Thus, in a 3-homogeneous (resp. $\overline{3}$-homogeneous) tree, no $\overline{3}$-sums (resp. 3-sums) are used. It follows from Corollaries 4.12 and 4.13 that if a code-decomposition tree has a node of the form $\mathsf{v} = (\mathcal{C}, \pi, \overline{\oplus}_3)$, with $\mathcal{C}$ being 3-connected, then the subtree with v as a root can be replaced with a different subtree in which the root node is $\mathsf{v'} = (\mathcal{C}, \pi', \oplus_3)$. Therefore, every code has a complete, 3-homogeneous (or $\overline{3}$-homogeneous) code-decomposition tree, which again can be constructed in time polynomial in the length of the code.

Having described in detail Seymour's decomposition theory in the context of binary linear codes, we now turn to some of its applications. This theory mainly derives its applications from families of codes that are minor-closed, and such families form the subject of the next section.

5. Minor-Closed Families of Codes

A family $\mathfrak{C}$ of binary linear codes is defined to be *minor-closed* if for each $\mathcal{C} \in \mathfrak{C}$, every code equivalent to a minor of $\mathcal{C}$ is also in $\mathfrak{C}$. Note that this definition automatically implies that a minor-closed family, $\mathfrak{C}$, of codes is closed under code equivalence, *i.e.*, if $\mathcal{C} \in \mathfrak{C}$, then all codes equivalent to $\mathcal{C}$ are also in $\mathfrak{C}$.

A non-trivial example of a minor-closed family is the set of all graphic codes, since any minor of a graphic code is graphic. We will encounter other examples of minor-closed families (regular codes and geometrically perfect codes) further on in this paper. We mention in passing another interesting example of such a family — codes of bounded trellis state-complexity. Recall from Section 3 that the state-complexity profile of a length-n code $\mathcal{C}$ is defined to be the vector $\mathbf{s}(\mathcal{C}) = (s_0(\mathcal{C}), \ldots, s_n(\mathcal{C}))$, where $s_i(\mathcal{C}) = \dim(\mathcal{C}|_J) + \dim(\mathcal{C}|_{J^c}) - \dim(\mathcal{C})$ for $J = [i] \subset [n]$. Define $s_{\max}(\mathcal{C}) = \max_{i \in [n]} s_i(\mathcal{C})$. For a fixed integer $w > 0$, let TC_w denote the family of codes $\mathcal{C}$ such that there exists a code $\mathcal{C}'$ equivalent to $\mathcal{C}$ with $s_{\max}(\mathcal{C}') \leq w$. Then, TC_w is minor-closed [13], [14]. A similar statement holds for the family of codes that have a cycle-free normal realization (cf. [36]) whose state-complexity is bounded by w.

A general construction of minor-closed families is obtained by fixing a collection $\mathcal{F}$ of codes, and defining $\mathfrak{C}_{\mathcal{F}}$ to be the set of all codes $\mathcal{C}$ such that no minor of $\mathcal{C}$ is equivalent to any $\mathcal{C}' \in \mathcal{F}$. As an example, let $\mathcal{F} = \{\mathcal{H}_7, \mathcal{H}_7^\perp, \mathcal{C}(K_5)^\perp, \mathcal{C}(K_{3,3})^\perp\}$, where $\mathcal{H}_7$ is the $[7,4]$ Hamming code[6]. By Theorem 2.1, $\mathfrak{C}_{\mathcal{F}}$ in this case is precisely the family of graphic codes. It is clear that $\mathfrak{C}_{\mathcal{F}}$ is a minor-closed family for any fixed $\mathcal{F}$. In fact, every minor-closed family can be obtained in this manner. Indeed, let $\mathfrak{C}$ be a minor-closed family of codes. A code $\mathcal{D}$ is said to be an *excluded minor* of $\mathfrak{C}$ if $\mathcal{D} \notin \mathfrak{C}$, but every proper minor of $\mathcal{D}$ is in $\mathfrak{C}$. It is not hard to verify that a code $\mathcal{C}$ is in $\mathfrak{C}$ iff no minor of $\mathcal{C}$ is an excluded minor of $\mathfrak{C}$. Theorem 2.1 is an example of such an *excluded-minor characterization*, and we will see more such examples (Theorems 5.3 and 6.2) further below. Thus, taking $\mathcal{F}$ to be the collection of all excluded minors of $\mathfrak{C}$, we have that $\mathfrak{C} = \mathfrak{C}_{\mathcal{F}}$. A tantalizing conjecture of Robertson and Seymour asserts that any minor-closed family $\mathfrak{C}$ of binary linear codes has only *finitely many* excluded minors [37, Conjecture 1.2].

Let $\mathfrak{C}$ be a minor-closed family of codes. By Theorem 4.14, every $\mathcal{C} \in \mathfrak{C}$ can be constructed from 3-connected, internally 4-connected codes in $\mathfrak{C}$ using direct sums, 2-sums, and 3- or $\overline{3}$-sums. The converse need not always be true, *i.e.*, it is not necessarily true that if a code $\mathcal{C}$ has a decomposition via direct sums, 2-sums, and 3- or $\overline{3}$-sums into codes in $\mathfrak{C}$, then $\mathcal{C} \in \mathfrak{C}$. Of course, the converse does hold if $\mathfrak{C}$ is also closed under the operations of direct sum, 2-sum, 3-sum and $\overline{3}$-sum. As usual, $\mathfrak{C}$ is defined to be closed under direct sum (resp. 2-sum, 3-sum, $\overline{3}$-sum) if for any pair of codes in $\mathcal{C}$, their direct sum (resp. 2-sum, 3-sum, $\overline{3}$-sum, if it can be defined) is also in $\mathfrak{C}$. We summarize this in the following proposition.

Proposition 5.1. *Let $\mathfrak{C}$ be a minor-closed family of codes that is also closed under the operations of direct sum, 2-sum, 3-sum and $\overline{3}$-sum. Then, the following are equivalent for a code $\mathcal{C}$.*

(i) *$\mathcal{C}$ is in $\mathfrak{C}$.*
(ii) *The leaves of some code-decomposition tree for $\mathcal{C}$ are in $\mathfrak{C}$.*
(iii) *The leaves of some complete, 3-homogeneous or $\overline{3}$-homogeneous code-decomposition tree for $\mathcal{C}$ are in $\mathfrak{C}$.*
(iv) *The leaves of every code-decomposition tree for $\mathcal{C}$ are in $\mathfrak{C}$.*

[6] From now on, $\mathcal{H}_7$ will always denote the $[7,4]$ Hamming code.

Proof. (i) implies (iv) since the leaves of any code-decomposition tree of $\mathcal{C}$ are minors of $\mathcal{C}$, and $\mathfrak{C}$ is minor-closed. The implications (iv) $\Rightarrow$ (iii) and (iii) $\Rightarrow$ (ii) are trivial. (ii) implies (i) since $\mathfrak{C}$ is closed under direct-sums, 2-sums, 3-sums and $\overline{3}$-sums. □

Since a complete code-decomposition tree of any code $\mathcal{C}$ can be constructed in time polynomial in the length of $\mathcal{C}$, we have the following corollary to the above result.

Corollary 5.2. *Let $\mathfrak{C}$ be a minor-closed family of codes that is also closed under the operations of direct sum, 2-sum, 3-sum and $\overline{3}$-sum. Then the following are equivalent statements.*

(i) *It can be decided in polynomial time whether or not a given code $\mathcal{C}$ is in $\mathfrak{C}$.*

(ii) *It can be decided in polynomial time whether or not a given 3-connected, internally 4-connected code $\mathcal{C}$ is in $\mathfrak{C}$.*

The first major application of results such as the above — the application which was in fact the motivation for Seymour's matroid decomposition theory — relates to totally unimodular matrices. A real matrix A is said to be *totally unimodular* if the determinant of every square submatrix of A is in $\{0, 1, -1\}$. In particular, each entry of a totally unimodular matrix is in $\{0, 1, -1\}$. Such matrices are of fundamental importance in combinatorial optimization and network flow problems, because total unimodularity is closely related to integer linear programming [15].

A binary matrix is defined to be *regular* if its 1's can be replaced by ±1's in such a way that the resulting matrix is totally unimodular. Consequently, a binary linear code is defined to be *regular* if it has a regular parity-check matrix. It turns out that for a regular code, *every* parity-check matrix is regular [26, Corollary 9.2.11]. Furthermore, given a regular binary matrix B, there is a polynomial-time algorithm that converts B to a totally unimodular matrix by assigning signs to the 1's in B [26, Corollary 9.2.7]. Thus, regular codes form the key to understanding total unimodularity. The following theorem, due to Tutte [38], provides an elegant excluded-minor characterization of regular codes.

Theorem 5.3. *A binary linear code is regular iff it does not contain as a minor any code equivalent to the [7,4] Hamming code or its dual.*

It follows from the theorem that the family of regular codes, which we will denote by $\mathfrak{R}$, is minor-closed, since it is of the form $\mathfrak{C}_{\mathcal{F}}$ for $\mathcal{F} = \{\mathcal{H}_7, \mathcal{H}_7^\perp\}$. Furthermore, $\mathfrak{R}$ is closed under the taking of code duals, *i.e.*, the dual of a regular code is also regular. This is because a code $\mathcal{C}$ contains $\mathcal{H}_7$ as a minor iff its dual $\mathcal{C}^\perp$ contains $\mathcal{H}_7^\perp$ as a minor. It can further be shown [18, p. 437] that $\mathfrak{R}$ is closed under the operations of direct sum, 2-sum, 3-sum and $\overline{3}$-sum.

Note that by Theorem 2.1, $\mathfrak{R}$ contains the family of graphic codes, and hence, the family of *co-graphic* codes as well, which are codes whose duals are graphic. Using a long and difficult argument, Seymour [6] proved that the 3-connected, internally 4-connected codes in $\mathfrak{R}$ are either graphic, co-graphic, or equivalent to a particular isodual code that he called R_{10}, which is neither graphic nor co-graphic.

Theorem 5.4 ([18], Corollary 13.2.6). *If $\mathcal{C}$ is a 3-connected, internally 4-connected regular code, then $\mathcal{C}$ is either graphic, co-graphic, or equivalent to R_{10}, which is the $[10, 5, 4]$ code with parity-check matrix*

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

Thus, Seymour's decomposition theory shows that any regular code (and so by assignment of signs, any totally unimodular matrix) can be constructed by piecing together — via direct sums, 2-sums, and 3-sums or $\overline{3}$-sums — graphic codes, co-graphic codes, and codes equivalent to R_{10}.

Also, membership in the family of regular codes can be decided in polynomial time. Indeed, as mentioned at the end of Section 2, there are polynomial-time algorithms for deciding whether or not a given code is graphic. Given an $m \times n$ parity-check matrix H for a code, a generator matrix for the code can be computed using elementary row operations on H in $O(m^2 n)$ time. Thus, the dual of a code can be determined in polynomial time, and hence it can be decided in polynomial time whether or not a given code is co-graphic. Hence, from Corollary 5.2 and Theorem 5.4, it follows that there is a polynomial-time algorithm for determining whether or not a given code is regular. The best such algorithm known is due to Truemper [39], which runs in $O((m+n)^3)$ time. Truemper's algorithm is also based on Seymour's decomposition theory, but it implements a highly efficient procedure for carrying out the decomposition.

While the application of Seymour's decomposition theory to regular codes is interesting, it is not very useful, perhaps, from a coding-theoretic perspective. However, in the next section, we give an application that should be of some interest to a coding theorist.

6. Application: ML Decoding

The recent work of Feldman, Wainwright and Karger [5] shows that ML decoding of a binary linear code $\mathcal{C}$ over a binary-input discrete memoryless channel can be formulated as a linear program (LP). Recall that the ML decoding problem is: given a received word $\mathbf{y}$ at the channel output, find a codeword $\mathbf{x} \in \mathcal{C}$ that maximizes the probability, $\Pr[\mathbf{y}|\mathbf{x}]$, of receiving $\mathbf{y}$ conditioned on the event that $\mathbf{x}$ was transmitted. As observed by Feldman *et al.*, under the assumption of a binary-input discrete memoryless channel, given a received word $\mathbf{y} = y_1 y_2 \ldots y_n$, the problem of determining $\arg\max_{\mathbf{x}\in\mathcal{C}} \Pr[\mathbf{y}|\mathbf{x}]$ is equivalent to the problem of finding $\arg\min_{\mathbf{x}\in\mathcal{C}} \langle \gamma, \mathbf{x} \rangle$, where $\gamma = (\gamma_1, \gamma_2, \ldots, \gamma_n)$ is given by

$$\gamma_i = \log\left(\frac{\Pr[y_i|x_i = 0]}{\Pr[y_i|x_i = 1]}\right) \tag{9}$$

and $\langle \cdot\,, \cdot \rangle$ is the standard inner product on $\mathbb{R}^n$. Here, for the inner product $\langle \gamma, \mathbf{x} \rangle$ to make sense, a binary codeword $\mathbf{x} = x_1 x_2 \ldots x_n \in \mathcal{C}$ is identified with the real vector $(x_1, x_2, \ldots, x_n) \in \{0,1\}^n \subset \mathbb{R}^n$.

The above formulation shows ML decoding to be equivalent to the minimization of a linear function over a finite set $\mathcal{C} \subset \{0,1\}^n$. Let $P(\mathcal{C})$ be the *codeword polytope* of $\mathcal{C}$, *i.e.*, the convex hull in $\mathbb{R}^n$ of the finite set $\mathcal{C}$. It can be shown that the set of vertices of $P(\mathcal{C})$ coincides with $\mathcal{C}$. The key point now is that over a polytope P, a linear function ϕ attains its minimum value $\phi_{\min} = \min\{\phi(\mathbf{x}) : \mathbf{x} \in P\}$ at a vertex of P. In particular, $\min_{\mathbf{x}\in\mathcal{C}} \langle \gamma, \mathbf{x} \rangle = \min_{\mathbf{x}\in P(\mathcal{C})} \langle \gamma, \mathbf{x} \rangle$, Thus, ML decoding is equivalent to finding a vertex of the polytope $P(\mathcal{C})$ that achieves $\min_{\mathbf{x}\in P(\mathcal{C})} \langle \gamma, \mathbf{x} \rangle$, which is a classic LP.

However, ML decoding of an arbitrary code is known to be NP-hard [9]. So, in general, solving the above LP over the codeword polytope is also NP-hard. A strategy often followed in such a situation is to "relax" the problem. The idea is to look for a polytope that contains the code as a subset of its vertex set, but which has some property that allows an LP defined over it to be solved more easily. Such a polytope is called a *relaxation* of the codeword polytope $P(\mathcal{C})$.

A certain relaxation of the codeword polytope has received much recent attention [5],[16],[40]. This is the polytope which, given a code $\mathcal{C}$ of length n, and a subset $H \subset \mathcal{C}^\perp$, is defined as

$$Q(H) = \bigcap_{\mathbf{h}\in H} P(\mathbf{h}^\perp),$$

where $P(\mathbf{h}^\perp)$ is the codeword polytope of the code $\mathbf{h}^\perp = \{\mathbf{c} \in \mathbb{F}_2^n :\ \langle \mathbf{h}, \mathbf{c} \rangle \equiv 0 \pmod 2\}$. Note that since $\mathcal{C} \subset \mathbf{h}^\perp$ for any $\mathbf{h} \in \mathcal{C}^\perp$, we have that $P(\mathcal{C}) \subset \bigcap_{\mathbf{h}\in H} P(\mathbf{h}^\perp) = Q(H)$ for any $H \subset \mathcal{C}^\perp$. In particular, $P(\mathcal{C}) \subset Q(\mathcal{C}^\perp)$ for any code $\mathcal{C}$.

For any $H \subset \mathcal{C}^\perp$, the polytope $Q(H)$ contains $\mathcal{C}$ as a subset of its vertex set, $\mathcal{V}(H)$. This is because $\mathcal{C} \subset Q(H) \cap \{0,1\}^n$, and since $Q(H)$ is contained within the n-cube $[0,1]^n$, we also have $Q(H) \cap \{0,1\}^n \subset \mathcal{V}(H)$. Thus, $Q(H)$ is indeed a relaxation of $P(\mathcal{C})$. Consequently the LP $\min_{\mathbf{x}\in Q(H)}\langle \gamma, \mathbf{x}\rangle$, where γ is the vector defined via (9), constitutes a relaxation of the LP that represents ML decoding.

Now, any standard LP-solving algorithm requires that the LP to be solved have its constraints be represented via linear inequalities. The advantage of using the relaxation $Q(H)$ is that there is a convenient such representation of the constraint $\mathbf{x} \in Q(H)$. The polytope $Q(H)$ can also be expressed as (see *e.g.* [5, Theorem 4] or [16, Lemma 26]),

$$Q(H) = \bigcap_{\mathbf{h}\in H} \Pi(\mathsf{supp}(\mathbf{h})), \tag{10}$$

where for $S \subset [n]$, $\Pi(S)$ denotes the polyhedron

$$\Pi(S) = \bigcap_{\substack{J \subset S \\ |J| \text{ odd}}} \left\{ (x_1, \ldots, x_n) \in [0,1]^n : \sum_{j\in J} x_j - \sum_{i \in S\setminus J} x_i \le |J| - 1 \right\}. \tag{11}$$

The efficiency of a practical LP solver (like, say, the simplex or ellipsoid algorithm) depends on the size of the LP representation, which is proportional to the number of variables and linear inequalities forming the constraints. If H above consists of the rows of a parity-check matrix of a low-density parity-check (LDPC) code, then the representation of $Q(H)$ given by (10)–(11) has size linear in the codelength n. So, the ellipsoid algorithm, for example, would be guaranteed to solve the LP $\min_{\mathbf{x}\in Q(H)}\langle \gamma, \mathbf{x}\rangle$ in time polynomial in n. However, as we explain next, this LP is no longer equivalent to ML decoding in general.

Let $\mathcal{J}(H) = \mathcal{V}(H) \cap \{0,1\}^n$ denote the set of *integral vertices* (*i.e.*, vertices all of whose coordinates are integers) of $Q(H)$. We noted above that $\mathcal{C} \subset \mathcal{J}(H)$. If H is a spanning subset (over $\mathbb{F}_2$) of $\mathcal{C}^\perp$, so that the vectors in H form (the rows of) a parity-check matrix of $\mathcal{C}$, then we in fact have $\mathcal{C} = \mathcal{J}(H)$. This is because if $\mathbf{x} \in \{0,1\}^n$ is not in $\mathcal{C}$, then $\mathbf{x} \notin \mathbf{h}^\perp$ for some $\mathbf{h} \in H$, and hence, $\mathbf{x} \notin P(\mathbf{h}^\perp) \supset Q(H)$. The polytope $Q(H)$ in this case is the "fundamental polytope" of Vontobel and Koetter [16] (or equivalently, the "projected polytope" $\overline{Q}$ of Feldman *et al.* [5, p. 958]). The fact that $\mathcal{C} = \mathcal{J}(H)$ for such a polytope $Q(H)$ implies that the polytope has the following "ML certificate" property [5, Proposition 2]: if the LP $\min_{\mathbf{x}\in Q(H)}\langle \gamma, \mathbf{x}\rangle$, where γ is the vector defined via (9), attains its minimum at some $\mathbf{x} \in \mathcal{J}(H)$, then $\mathbf{x}$ is guaranteed to be an ML codeword. However, it is possible that the above LP attains its minimum at some non-integral vertex $\mathbf{x} \in \mathcal{V}(H) - \mathcal{J}(H)$, in which case decoding via linear programming over $Q(H)$ fails. The non-integral vertices of $Q(H)$ are called "pseudocodewords".

It is naturally of interest to know when a code $\mathcal{C}$ has a fundamental polytope $Q(H)$ (for some spanning subset H of $\mathcal{C}^\perp$) without pseudocodewords. For such codes, ML decoding can be exactly implemented as an LP over $Q(H)$. Clearly, $Q(H)$ has no pseudocodewords iff $\mathcal{C} = \mathcal{V}(H)$, or equivalently, $P(\mathcal{C}) = Q(H)$. But since $P(\mathcal{C}) \subset Q(\mathcal{C}^\perp) \subset Q(H)$, this obviously implies that we must have $P(\mathcal{C}) = Q(\mathcal{C}^\perp)$. Conversely, if $P(\mathcal{C}) = Q(\mathcal{C}^\perp)$, then we may simply take $H = \mathcal{C}^\perp$ to obtain a fundamental polytope $Q(H)$ without pseudocodewords. We record this observation as a lemma.

Lemma 6.1. *Let $\mathcal{C}$ be a binary linear code. There exists a spanning subset, H, of $\mathcal{C}^\perp$ such that the polytope $Q(H)$ has no pseudocodewords iff $P(\mathcal{C}) = Q(\mathcal{C}^\perp)$.*

A code $\mathcal{C}$ for which $P(\mathcal{C}) = Q(\mathcal{C}^\perp)$ holds will be called *geometrically perfect,* and we will denote by $\mathfrak{G}$ the family of all such codes. So the question then is: which codes are geometrically perfect? An answer to this was provided by Barahona and Grötschel [41], who showed that the relationship $P(\mathcal{C}) = Q(\mathcal{C}^\perp)$ is equivalent to Seymour's "sums-of-circuits" property for binary matroids [7].

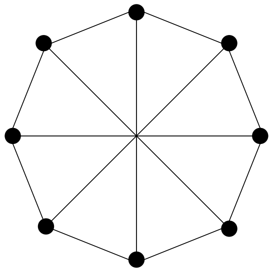

FIGURE 4. The graph V_8.

The following theorem is thus equivalent to Seymour's characterization of binary matroids with the sums-of-circuits property.

Theorem 6.2 ([41], Theorem 3.5)**.** *A binary linear code $\mathcal{C}$ is geometrically perfect iff $\mathcal{C}$ does not contain as a minor any code equivalent to $\mathcal{H}_7^\perp$, R_{10} or $\mathcal{C}(K_5)^\perp$.*

By the above theorem, $\mathfrak{G}$ is minor-closed. Moreover, since none of $\mathcal{H}_7^\perp$, R_{10} or $\mathcal{C}(K_5)^\perp$ is graphic, no graphic code can contain any of them as a minor, and so graphic codes are geometrically perfect.

Grötschel and Truemper [8, Section 4] showed that $\mathfrak{G}$ is closed under the operations of direct sum, 2-sum and $\overline{3}$-sum, but is not closed under 3-sum. They also observed [8, p. 326] that any code in $\mathfrak{G}$ can be constructed via direct sums, 2-sums and $\overline{3}$-sums from graphic codes and copies of the codes $\mathcal{H}_7$, $\mathcal{C}(K_{3,3})^\perp$ and $\mathcal{C}(V_8)^\perp$, where V_8 is the graph in Figure 4. Indeed, this result is implied by Theorems 6.4, 6.9 and 6.10 in [7]. It is not hard to verify that the codes $\mathcal{H}_7$, $\mathcal{C}(K_{3,3})^\perp$ and $\mathcal{C}(V_8)^\perp$ are in fact in $\mathfrak{G}$. Putting these facts together, we obtain the following theorem.

Theorem 6.3. *For a binary linear code $\mathcal{C}$, the following are equivalent statements.*

(i) *$\mathcal{C}$ is geometrically perfect,* i.e., *$P(\mathcal{C}) = Q(\mathcal{C}^\perp)$.*
(ii) *Each leaf in some complete, $\overline{3}$-homogeneous code-decomposition tree for $\mathcal{C}$ is either graphic, or equivalent to one of the codes $\mathcal{H}_7$, $\mathcal{C}(K_{3,3})^\perp$ and $\mathcal{C}(V_8)^\perp$.*
(iii) *Each leaf in every complete, $\overline{3}$-homogeneous code-decomposition tree for $\mathcal{C}$ is either graphic, or equivalent to one of the codes $\mathcal{H}_7$, $\mathcal{C}(K_{3,3})^\perp$ and $\mathcal{C}(V_8)^\perp$.*

Proof. (i) implies (iii) follows directly from the fact that any code in $\mathfrak{G}$ can be constructed via direct sums, 2-sums and $\overline{3}$-sums from graphic codes and copies (up to equivalence) of the codes $\mathcal{H}_7$, $\mathcal{C}(K_{3,3})^\perp$ and $\mathcal{C}(V_8)^\perp$. The implication (iii) $\Rightarrow$ (ii) is trivial. Finally, (ii) $\Rightarrow$ (i) holds since graphic codes and the codes $\mathcal{H}_7$, $\mathcal{C}(K_{3,3})^\perp$ and $\mathcal{C}(V_8)^\perp$ are all in $\mathfrak{G}$, and $\mathfrak{G}$ is closed under direct sum, 2-sum and $\overline{3}$-sum. □

Since a complete, $\overline{3}$-homogeneous code-decomposition tree for a code can be constructed in polynomial time, and testing for graphicness or equivalence to $\mathcal{H}_7$, $\mathcal{C}(K_{3,3})^\perp$ and $\mathcal{C}(V_8)^\perp$ can also be carried out in polynomial time, we have the following corollary to the above theorem.

Corollary 6.4. *It can be decided in polynomial time whether or not a given code $\mathcal{C}$ is geometrically perfect,* i.e., *has the property $P(\mathcal{C}) = Q(\mathcal{C}^\perp)$.*

However, this is only half the story. If $\mathcal{C}$ is a geometrically perfect code, the algorithm guaranteed by the above result will determine this to be the case, but will not produce a "small" subset $H \subset \mathcal{C}^\perp$ such that $P(\mathcal{C}) = Q(H)$. The only information we would have is that H can be taken to be the *entire* dual code $\mathcal{C}^\perp$. While it would then be true that the LP $\min_{\mathbf{x} \in Q(\mathcal{C}^\perp)} \langle \gamma, \mathbf{x} \rangle$ is equivalent to ML decoding, the representation of $Q(\mathcal{C}^\perp)$ given by (10)–(11) still has size exponential in the codelength n. It is thus unclear whether there is an LP-solving algorithm than can be used to solve this LP efficiently in practice[7]. Fortunately, as we shall describe next, for the family, $\mathfrak{G}$, of geometrically perfect codes, ML decoding can always be implemented in time polynomial in codelength, *not* using an LP-solving algorithm, but by means of a combinatorial optimization algorithm that uses code decompositions. One major feature of this combinatorial algorithm is that, for any given code $\mathcal{C}$ (not necessarily geometrically perfect), if a suitable decomposition (as in Definition 6.1 below) of $\mathcal{C}$ exists, that decomposition can be used to efficiently solve *any* instance of the LP $\min_{\mathbf{x}\in P(\mathcal{C})}\langle\gamma,\mathbf{x}\rangle$. This means that the determination of a suitable code decomposition of $\mathcal{C}$, which only needs to be done once, can even be done "off-line". When $\mathcal{C}$ is geometrically perfect, such a suitable decomposition always exists, and in fact, can be determined in time polynomial in the length of $\mathcal{C}$, as we explain next.

In a series of papers [42], [43], [39] (see also [26]), Truemper carried out a detailed examination of matroid decompositions, from which a particularly interesting observation concerning geometrically perfect codes could be inferred. Let $\mathfrak{G}_0$ be the sub-family of $\mathfrak{G}$ that consists of all graphic codes and codes equivalent to one of $\mathcal{H}_7$, $\mathcal{C}(K_{3,3})^\perp$ and $\mathcal{C}(V_8)^\perp$. As observed in the proof of Corollary 6.6 in [8], Truemper's analysis of matroid decompositions could be used to show that a code $\mathcal{C} \in \mathfrak{G} - \mathfrak{G}_0$ can always be composed via a 2-sum or a $\overline{3}$-sum from a code $\mathcal{C}_1 \in \mathfrak{G}_0$, and a code $\mathcal{C}_2 \in \mathfrak{G}$, both of which may be taken to be codes equivalent to minors of $\mathcal{C}$. In fact, a much more precise statement may be deduced from Theorem 2.5 in [8]. If $\mathcal{C}$ is a 2-connected code in $\mathfrak{G} - \mathfrak{G}_0$, then it follows from Theorem 2.5 in [8] that $\mathcal{C}$ can always be decomposed into codes $\mathcal{C}_1$ and $\mathcal{C}_2$, both equivalent to minors of $\mathcal{C}$, in at least one of the following ways:

(a) $\mathcal{C}$ is equivalent to $\mathcal{C}_1 \oplus_2 \mathcal{C}_2$, for some code $\mathcal{C}_1 \in \mathfrak{G}_0$ with $\dim(\mathcal{C}_1) \geq 2$, and some 2-connected code $\mathcal{C}_2 \in \mathfrak{G}$; or

(b) $\mathcal{C}$ is equivalent to $\mathcal{C}_1 \,\overline{\oplus}_3\, \mathcal{C}_2$, for some code $\mathcal{C}_1 \in \mathfrak{G}_0$ with $\dim(\mathcal{C}_1) \geq 3$, and some 2-connected code $\mathcal{C}_2 \in \mathfrak{G}$.

The decomposition of $\mathcal{C}$ into $\mathcal{C}_1$ and $\mathcal{C}_2$, along with the coordinate permutation that takes $\mathcal{C}_1 \oplus_2 \mathcal{C}_2$ or $\mathcal{C}_1 \,\overline{\oplus}_3\, \mathcal{C}_2$ (as the case may be) to $\mathcal{C}$, can be determined in polynomial time. These facts have some significant consequences, one of which is that any code in $\mathfrak{G}$ is ML-decodable in polynomial time. However, rather than state these results just for the class of geometrically perfect codes, we will state and prove them more generally for codes that are "almost graphic" in the sense that they can be composed from graphic codes and finitely many other codes.

Recall that Γ denotes the family of graphic codes.

Definition 6.1. *A minor-closed family of codes $\mathfrak{C}$ is defined to be* almost-graphic *if there exists a finite sub-family $\mathfrak{D} \subset \mathfrak{C}$ with the following property:*

for any 2-connected code $\mathcal{C} \in \mathfrak{C}$, either $\mathcal{C} \in \Gamma \cup \mathfrak{D}$, or there are codes $\mathcal{C}_1$ and $\mathcal{C}_2$, along with a permutation π of the coordinate set of $\mathcal{C}$, such that

(a) *$\mathcal{C}_1$ and $\mathcal{C}_2$ are equivalent to minors of $\mathcal{C}$;*

(b) *$\mathcal{C}_2$ is 2-connected; and*

(c) *either* (i) $\mathcal{C} = \pi(\mathcal{C}_1 \oplus_2 \mathcal{C}_2)$, *with* $\mathcal{C}_1 \in \Gamma \cup \mathfrak{D}$, $\dim(\mathcal{C}_1) \geq 2$, *or* (ii) $\mathcal{C} = \pi(\mathcal{C}_1 \,\overline{\oplus}_3\, \mathcal{C}_2)$, *with* $\mathcal{C}_1 \in \Gamma \cup \mathfrak{D}$, $\dim(\mathcal{C}_1) \geq 3$.

If there exists a constant $l > 0$ such that for any length-n code $\mathcal{C} \in \mathfrak{C} - (\Gamma \cup \mathfrak{D})$, the components $\mathcal{C}_1$, $\mathcal{C}_2$ and π of the above decomposition can be determined in time $O(n^l)$, then the family of codes $\mathfrak{C}$ is said to be polynomially almost-graphic (PAG).

Note that the 3-sum is conspicuous by its absence from the above definition. We will give an explanation of this at the end of this section.

[7] It has been shown in [8] that for geometrically perfect codes, the ML-decoding LP can in fact be solved in polynomial time by the ellipsoid algorithm. It has also been noted there that the ellipsoid algorithm is — in a straightforward implementation — of "doubtful practical relevance."

Definition 6.1 clearly implies that any code in an almost-graphic family $\mathfrak{C}$ has a $\overline{3}$-homogeneous code-decomposition tree. The definition in fact implies that a 2-connected code in $\mathfrak{C}$ has a decomposition tree with the property that each leaf is in $\Gamma \cup \mathfrak{D}$, and for each non-leaf node v in the tree, $\mathsf{lchild.code}$ is a leaf (and hence, is in $\Gamma \cup \mathfrak{D}$), where lchild is the left child of v. Such a code-decomposition tree will be called $(\Gamma \cup \mathfrak{D})$-*unary*. If $\mathfrak{C}$ is PAG, a $\overline{3}$-homogeneous, $(\Gamma \cup \mathfrak{D})$-unary decomposition tree can be constructed for any 2-connected code $\mathcal{C} \in \mathfrak{C}$ in time polynomial in the length of $\mathcal{C}$.

The family, Γ, of graphic codes is trivially PAG. From the discussion prior to the above definition, the family, $\mathfrak{G}$, of geometrically perfect codes is also PAG. Other examples of PAG families are the code families $\mathfrak{C}_{\mathcal{F}}$ (cf. Section 5) for $\mathcal{F} = \{\mathcal{H}_7, \mathcal{C}(K_5)^\perp\}$ and $\mathcal{F} = \{\mathcal{H}_7^\perp, \mathcal{C}(K_5)^\perp\}$, and the family of co-graphic codes without a $\mathcal{C}(K_5)^\perp$ minor [8, Theorem 2.5]. PAG codes inherit some of the properties of graphic codes. For example, it is known [44],[11] that the ML decoding problem over a memoryless binary symmetric channel can be solved in polynomial time for the family of graphic codes using Edmonds' matching algorithm [45],[46]. A much stronger decoding result can in fact be proved for graphic codes, and more generally for PAG codes. This is based on the following optimization result proved in [8], an argument for which is sketched in Appendix B.

Theorem 6.5 ([8], Theorem 6.5)**.** *Let $\mathfrak{C}$ be a PAG family of codes. There exists a constant $l > 0$ such that given any length-n code $\mathcal{C}$ in $\mathfrak{C}$ and any $\gamma \in \mathbb{R}^n$, a codeword $\mathbf{c}_{\min} \in \mathcal{C}$ achieving $\min_{\mathbf{c}\in\mathcal{C}} \langle \gamma, \mathbf{c}\rangle$ (or equivalently, $\min_{\mathbf{x}\in P(\mathcal{C})} \langle \gamma, \mathbf{x}\rangle$) can be determined in $O(n^l)$ time.*

It should be noted that an actual implementation of the polynomial-time algorithm implicit in Theorem 6.5 (and outlined in Appendix B) requires arithmetic over the real numbers, unless the vector γ has only rational coordinates. So, in practice, finite-precision arithmetic used in any computer implementation of the algorithm could only approximate the linear cost function $\langle \gamma, \mathbf{c}\rangle$ for an arbitrary $\gamma \in \mathbb{R}^n$.

As mentioned earlier, ML decoding over a binary-input discrete memoryless channel can be formulated as a linear program [5]. Therefore, we have the following corollary to the above theorem, again with the caveat that a true implementation of a polynomial-time algorithm for ML decoding would require real-number arithmetic.

Corollary 6.6. *The maximum-likelihood decoding problem over a binary-input discrete memoryless channel can be solved in polynomial time for a PAG family of codes. In particular, geometrically perfect codes are ML-decodable in polynomial time.*

Another problem that is known to be NP-hard in general is the problem of determining the minimum distance of a code [47]. For graphic codes, this is equivalent to the problem of finding the girth of a graph, which can be solved in time polynomial in the number of edges of the graph — one of the earliest such algorithms published [48] runs in $O(n^{3/2})$ time in the worst case, where n is the number of edges. As a consequence of Theorem 6.5, we also have that the minimum distance problem for a PAG family of codes can be solved in polynomial time.

Corollary 6.7. *The minimum distance of any code in a PAG family can be determined in polynomial time.*

Proof. Let $\mathcal{C}$ be a code of length n containing at least one nonzero codeword. For $i = 1, 2, \ldots, n$, define $\gamma^{(i)} = (1, \ldots, 1, -n, 1, \ldots, 1)$, with the $-n$ appearing in the ith coordinate of $\gamma^{(i)}$. Note that $\min_{\mathbf{c}\in\mathcal{C}} \langle \gamma^{(i)}, \mathbf{c}\rangle$ is always achieved by a codeword in $\mathcal{C}$ with ith coordinate equal to 1, if such a codeword exists. Indeed, the minimum-achieving codeword $\mathbf{c}^{(i)}$ is the codeword of least Hamming weight among codewords in $\mathcal{C}$ that have a 1 in the ith coordinate; if there is no codeword in $\mathcal{C}$ with a 1 in the ith coordinate, then $\mathbf{c}^{(i)} = \mathbf{0}$. Note that if $\mathbf{c}^{(i)} \neq \mathbf{0}$, then $\langle \gamma^{(i)}, \mathbf{c}^{(i)}\rangle = w(\mathbf{c}^{(i)}) - 1 - n$. Therefore, the minimum distance of $\mathcal{C}$ is given by

$$d = n + 1 + \min_{i\in[n]} \min_{\mathbf{c}\in\mathcal{C}} \langle \gamma^{(i)}, \mathbf{c}\rangle.$$

For a PAG family of codes $\mathfrak{C}$, given any code $\mathcal{C} \in \mathfrak{C}$, each of the minimization problems $\min_{\mathbf{c}\in\mathcal{C}}\langle\gamma^{(i)}, \mathbf{c}\rangle$ can be solved in polynomial time, and hence the minimum distance of $\mathcal{C}$ can be determined in polynomial time. □

We point out that in all the minimization problems that must be solved (recursively going down the code-decomposition tree, as explained in Appendix B) to determine the minimum distance of a length-n code, the cost vectors have integer coefficients of magnitude at most n^2. So, the minimum distance of a code from a PAG family can be determined in polynomial time using finite-precision arithmetic.

The downside of PAG (and more generally, almost-graphic) code families is that they are not very good from a coding-theoretic perspective. 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.

Theorem 6.8. *An almost-graphic family of codes cannot be asymptotically good.*

In particular, the family of geometrically perfect codes is not asymptotically good. The theorem is proved in Appendix C. In view of Lemma 6.1, the above result has the following very interesting corollary.

Corollary 6.9. *Let $\mathfrak{C}$ be a family of binary linear codes with the following property: for each $\mathcal{C} \in \mathfrak{C}$, there exists a parity-check matrix H for $\mathcal{C}$, such that the corresponding fundamental polytope $Q(H)$ has no non-integral vertices (pseudocodewords). Then, $\mathfrak{C}$ is not an asymptotically good code family.*

Loosely speaking, this means that linear-programming decoding, when applied to a "good" code, must suffer on occasion from decoding failure due to the presence of pseudocodewords, even if *all* possible parity checks (dual codewords) are used in the constraints (10)–(11) of the LP. Given the close relationship between linear-programming decoding and iterative decoding using the min-sum algorithm [49], a similar result is likely to hold for iterative decoding as well.

We end this section with an explanation of why 3-sums were left out of Definition 6.1. The proof of Theorem 6.5 given in Appendix B relies crucially on the fact that a $\overline{3}$-homogeneous, $(\Gamma \cup \mathfrak{D})$-unary code-decomposition tree can be constructed in polynomial time for a code from a PAG family. So, the result is actually true for any code family for which such trees can be constructed in polynomial time. Now, if 3-sums were allowed in Definition 6.1, it is no longer obvious that codes from the resulting code family would still have $\overline{3}$-homogeneous, $(\Gamma \cup \mathfrak{D})$-unary code-decomposition trees. There is good reason to think that this could still be true, especially in light of Corollary 4.10, which states that a code has a 3-sum decomposition only if it has a $\overline{3}$-sum decomposition. Indeed, that result may lead us to believe that if a code $\mathcal{C}$ has a $(\Gamma \cup \mathfrak{D})$-unary code-decomposition tree which contains 3-sums, then replacing the 3-sums in the tree with $\overline{3}$-sums in the manner prescribed by Corollary 4.10 should result in a $\overline{3}$-homogeneous, $(\Gamma \cup \mathfrak{D})$-unary code-decomposition tree. However, to show that this is the case, it would have to be verified that if $\mathcal{C} = \mathcal{C}_1 \oplus_3 \mathcal{C}_2$, with $\mathcal{C}_1$, $\mathcal{C}_2$ satisfying (a) and (b) of Definition 6.1, then (in the notation of Corollary 4.10) $\overline{\mathcal{C}_1}$ and $\overline{\mathcal{C}_2}$ also satisfy (a) and (b). Now, it is not hard to check that if $\mathcal{C}_1$ is graphic, then so is $\overline{\mathcal{C}_1}$. However, unless $\mathcal{C}$ is 3-connected, there is no guarantee that $\overline{\mathcal{C}_1}$ and $\overline{\mathcal{C}_2}$ are equivalent to minors of $\mathcal{C}$, even though $\mathcal{C}_1$ and $\mathcal{C}_2$ are given to be equivalent to minors of $\mathcal{C}$.

It is in fact quite likely to be true that if $\mathcal{C} = \mathcal{C}_1 \oplus_3 \mathcal{C}_2$, with $\mathcal{C}_1$ and $\mathcal{C}_2$ equivalent to minors of $\mathcal{C}$, then $\overline{\mathcal{C}_1}$ and $\overline{\mathcal{C}_2}$ are also equivalent to minors of $\mathcal{C}$. So, it is quite possible that if Definition 6.1 were to include 3-sums as well, then the resulting code families would still have $\overline{3}$-homogeneous, $(\Gamma \cup \mathfrak{D})$-unary code-decomposition trees, and hence Theorem 6.5 would continue to hold. However, a rigorous proof of this would take us far outside the main theme of our paper, and Definition 6.1 as it stands is good enough for our purposes.

7. Concluding Remarks

A natural question to ask upon studying the decomposition theory presented in this paper is whether one can define k-sums for $k \geq 4$ that have the same attractive properties as 2-, 3- and $\overline{3}$-sums. Ideally, such a k-sum, denoted by $\oplus_k$, would have the following property for some fixed integer $l \geq k$:

> a k-connected code $\mathcal{C}$ has a k-separation (J, J^c) with $\min\{|J|, |J^c|\} \geq l$ iff $\mathcal{C} = \pi(\mathcal{C}_1 \oplus_k \mathcal{C}_2)$ for some permutation π of the coordinates of $\mathcal{C}$, and codes $\mathcal{C}_1$ and $\mathcal{C}_2$ equivalent to minors of $\mathcal{C}$.

It is indeed possible to define k-sums in such a way that we have $\mathcal{C} = \pi(\mathcal{C}_1 \oplus_k \mathcal{C}_2)$ iff $\mathcal{C}$ has a k-separation (J, J^c) with $\min\{|J|, |J^c|\} \geq l$. The tricky part is ensuring that the component codes $\mathcal{C}_1$ and $\mathcal{C}_2$ are retained as minors of $\mathcal{C}$, and this appears to be difficult in general. However, we have some preliminary results that indicate that even without the last property, such k-sums can be used as the building blocks of a decomposition theory that ties in beautifully with Forney's theory of cycle-free realizations of linear codes [36]. This theory would make further deep connections with matroid theory, particularly with the notions of matroid branchwidth and treewidth [50], [51]. An exposition of this theory will be given in a future paper.

While the decomposition theory in this paper has been presented mainly in the context of binary linear codes, it is possible to extend some of it to linear codes over arbitrary finite fields as well. The definitions of minors and k-connectedness can be obviously extended to nonbinary codes. Also, a 2-sum operation can be defined for codes over an arbitrary finite field $\mathbb{F}$, and the entire theory outlined in Section 4.1 does carry over. However, there is no known 3-sum operation defined for nonbinary codes that has a property analogous to that stated in Theorem 4.11. Again, it seems that if we are prepared to give up the requirement that the components of a k-sum be retained as minors of the composite code, then it is possible to develop a powerful decomposition theory for nonbinary codes just like that for binary codes.

We end with a pointer to a very interesting direction of current research in matroid theory. This involves the resolution of two conjectures whose statements (in the context of codes) we give below. Recall from Section 5 that the notation $\mathfrak{C}_{\mathcal{F}}$, for some fixed collection $\mathcal{F}$ of codes, refers to the set of all binary linear codes $\mathcal{C}$ such that no minor of $\mathcal{C}$ is equivalent to any $\mathcal{C}' \in \mathcal{F}$. We extend that notation to codes over an arbitrary finite field $\mathbb{F}$ as well.

Conjecture 7.1 ([37], Conjecture 1.2)**.** *If $\mathfrak{C}$ is a minor-closed class of codes over a finite field $\mathbb{F}$, then $\mathfrak{C} = \mathfrak{C}_{\mathcal{F}}$ for some finite collection of codes $\mathcal{F}$.*

Informally, the above conjecture states that any minor-closed class of codes is characterized by a finite list of excluded minors.

Conjecture 7.2 ([37], Conjecture 1.3)**.** *Let $\mathcal{M}$ be a fixed code. Given a length-n code $\mathcal{C}$, it is decidable in time polynomial in n whether or not $\mathcal{C}$ contains $\mathcal{M}$ as a minor.*

The two conjectures together imply that the membership of a code in a minor-closed class can always be decided in polynomial time. To put it another way, if a property of codes is preserved under the action of taking minors, then it should be decidable in polynomial time whether or not a given code has that property. It should be pointed out that both conjectures have been shown to be true in the context of graphic codes, as part of the celebrated Graph Minor Project of Robertson and Seymour [52],[53]. The Graph Minor Project has had a profound impact on modern graph theory [54], and its extension to $\mathbb{F}$-representable matroids (equivalently, codes over $\mathbb{F}$) is bound to have a similar influence on matroid theory and, as a consequence, on coding theory.

Appendix A. Proofs of Propositions 4.2 and 4.9

Proof of Proposition 4.2: Let $\mathcal{C}$ and $\mathcal{C}'$ be $[n, k]$ and $[n', k']$ codes, respectively, for which $\mathcal{C} \oplus_2 \mathcal{C}'$ can be defined. We want to show that $(\mathcal{C} \oplus_2 \mathcal{C}')^\perp = \mathcal{C}^\perp \oplus_2 \mathcal{C}'^\perp$.

First, observe that, by Proposition 4.1(a),

$$\begin{aligned}\dim((\mathcal{C} \oplus_2 \mathcal{C}')^{\perp}) &= (n+n'-2) - \dim(\mathcal{C} \oplus_2 \mathcal{C}') = (n+n'-2) - (k+k'-1) \\ &= (n-k) + (n'-k') - 1 = \dim(\mathcal{C}^{\perp} \oplus_2 \mathcal{C}'^{\perp}).\end{aligned}$$

Therefore, it is enough to show that $\mathcal{C}^{\perp} \oplus_2 \mathcal{C}'^{\perp} \subset (\mathcal{C} \oplus_2 \mathcal{C}')^{\perp}$.

Given a binary word $\mathbf{x} = x_1 x_2 \ldots x_n$ and a positive integer $m \le n$, let $\mathbf{x}_{:m}$ denote the length-m prefix of $\mathbf{x}$, and let $\mathbf{x}_{m:}$ denote the length-m suffix of $\mathbf{x}$. Also, we denote the concatenation of two binary words $\mathbf{x}$ and $\mathbf{x}'$ by $\mathbf{x} \parallel \mathbf{x}'$.

Consider an arbitrary codeword, $\overline{\mathbf{x}}$, of $\mathcal{C}^{\perp} \oplus_2 \mathcal{C}'^{\perp}$. Such a word is of the form $\mathbf{x}_{[:n-1]} \parallel \mathbf{x}'_{[n'-1:]}$ for some $\mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{C}^{\perp}$ and $\mathbf{x}' = (x'_1, \ldots, x'_{n'}) \in \mathcal{C}'^{\perp}$ such that $x_n = x'_1$. We will show that $\mathbf{x}_{[:n-1]} \parallel \mathbf{x}'_{[n-1:]}$ as above must also be in $(\mathcal{C} \oplus_2 \mathcal{C}')^{\perp}$.

Let $\overline{\mathbf{c}}$ be an arbitrary codeword of $(\mathcal{C} \oplus_2 \mathcal{C}')$. Such a codeword is of the form $\mathbf{c}_{[:n-1]} \parallel \mathbf{c}'_{[n'-1:]}$ for some $\mathbf{c} = (c_1, \ldots, c_n) \in \mathcal{C}$ and $\mathbf{c}' = (c'_1, \ldots, c'_{n'}) \in \mathcal{C}'$ such that $c_n = c'_1$. The dot product $\overline{\mathbf{c}} \cdot \overline{\mathbf{x}}$ evaluates to $\sum_{i=1}^{n-1} c_i x_i + \sum_{i=2}^{n'} c'_i x'_i$, all addition operations being performed modulo 2. But since $\mathbf{c} \in \mathcal{C}$ and $\mathbf{x} \in \mathcal{C}^{\perp}$, we have $\sum_{i=1}^{n} c_i x_i = 0$, from which we obtain $\sum_{i=1}^{n-1} c_i x_i = c_n x_n$. Similarly, $\sum_{i=2}^{n'} c'_i x'_i = c'_1 x'_1$. Hence, $\overline{\mathbf{c}} \cdot \overline{\mathbf{x}} = c_n x_n + c'_1 x'_1 = 0$, since $c_n = c'_1$ and $x_n = x'_1$. As $\overline{\mathbf{c}}$ is an arbitrary codeword of $(\mathcal{C} \oplus_2 \mathcal{C}')$, we have shown that $\overline{\mathbf{x}} \in (\mathcal{C} \oplus_2 \mathcal{C}')^{\perp}$.

Therefore, $\mathcal{C}^{\perp} \oplus_2 \mathcal{C}'^{\perp} \subset (\mathcal{C} \oplus_2 \mathcal{C}')^{\perp}$, which completes the proof. □

The proof of Proposition 4.9 closely resembles that of Proposition 4.2, so we continue to use the notation introduced in the latter proof.

Proof of Proposition 4.9: Let $\mathcal{C}$ and $\mathcal{C}'$ be $[n,k]$ and $[n',k']$ codes, respectively, for which $\mathcal{C} \oplus_3 \mathcal{C}'$ can be defined. Observe that, by Propositions 4.7 and 4.8, we have

$$\begin{aligned}\dim((\mathcal{C} \oplus_3 \mathcal{C}')^{\perp}) &= (n+n'-6) - \dim(\mathcal{C} \oplus_3 \mathcal{C}') = (n+n'-6) - (k+k'-4) \\ &= (n-k) + (n'-k') - 2 = \dim(\mathcal{C}^{\perp} \,\overline{\oplus}_3\, \mathcal{C}'^{\perp}).\end{aligned}$$

Therefore, to prove Proposition 4.9, it is enough to show that $\mathcal{C}^{\perp} \,\overline{\oplus}_3\, \mathcal{C}'^{\perp} \subset (\mathcal{C} \oplus_3 \mathcal{C}')^{\perp}$.

It is easily seen that an arbitrary codeword, $\widehat{\mathbf{x}}$, of $\mathcal{C}^{\perp} \,\overline{\oplus}_3\, \mathcal{C}'^{\perp}$ must be of the form $\mathbf{x}_{[:n-3]} \parallel \mathbf{x}'_{[n'-3:]}$ for some $\mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{C}^{\perp}$ and $\mathbf{x}' = (x'_1, \ldots, x'_{n'}) \in \mathcal{C}'^{\perp}$ such that $(x_{n-2}, x_{n-1}, x_n) = (x'_1, x'_2, x'_3)$. We will show that any such $\widehat{\mathbf{x}}$ is also in $(\mathcal{C} \oplus_3 \mathcal{C}')^{\perp}$.

Let $\widehat{\mathbf{c}}$ be an arbitrary codeword of $(\mathcal{C} \oplus_3 \mathcal{C}')$, so that it is of the form $\mathbf{c}_{[:n-3]} \parallel \mathbf{c}'_{[n'-3:]}$ for some $\mathbf{c} = (c_1, \ldots, c_n) \in \mathcal{C}$ and $\mathbf{c}' = (c'_1, \ldots, c'_{n'}) \in \mathcal{C}'$ such that $(c_{n-2}, c_{n-1}, c_n) = (c'_1, c'_2, c'_3)$. The dot product $\widehat{\mathbf{c}} \cdot \widehat{\mathbf{x}}$ evaluates to $\sum_{i=1}^{n-3} c_i x_i + \sum_{i=4}^{n'} c'_i x'_i$, all addition operations being performed modulo 2. But since $\mathbf{c} \in \mathcal{C}$ and $\mathbf{x} \in \mathcal{C}^{\perp}$, we have $\sum_{i=1}^{n} c_i x_i = 0$, from which we obtain $\sum_{i=1}^{n-3} c_i x_i = \sum_{i=n-2}^{n} c_i x_i$. Similarly, $\sum_{i=4}^{n'} c'_i x'_i = \sum_{i=1}^{3} c'_i x'_i$. Hence, $\widehat{\mathbf{c}} \cdot \widehat{\mathbf{x}} = \sum_{i=n-2}^{n} c_i x_i + \sum_{i=1}^{3} c'_i x'_i$, which equals 0 since $c_{n+i-3} = c'_i$ and $x_{n+i-3} = x'_i$ for $i = 1, 2, 3$. As $\widehat{\mathbf{c}}$ is an arbitrary codeword of $(\mathcal{C} \oplus_3 \mathcal{C}')$, we have shown that $\widehat{\mathbf{x}} \in (\mathcal{C} \oplus_3 \mathcal{C}')^{\perp}$.

Therefore, $\mathcal{C}^{\perp} \,\overline{\oplus}_3\, \mathcal{C}'^{\perp} \subset (\mathcal{C} \oplus_3 \mathcal{C}')^{\perp}$, which completes the proof. □

APPENDIX B. SKETCH OF PROOF OF THEOREM 6.5

We will only provide a sketch of the proof of Theorem 6.5, as it is a result extant in the literature [8, Theorem 6.5]. The purpose of sketching out the proof is that it outlines the polynomial-time algorithm that determines

$$\min_{\mathbf{c} \in \mathcal{C}} \langle \gamma, \mathbf{c} \rangle, \tag{12}$$

for a length-n code $\mathcal{C}$ from a PAG family, and a cost vector $\gamma \in \mathbb{R}^n$.

The proof of the theorem is by induction using the following lemma and the fact (explained further below) that the minimization of a linear cost function can be done in polynomial time for graphic codes.

Lemma B.1. (a) *Let $\mathcal{C} = \mathcal{C}_1 \oplus_2 \mathcal{C}_2$. Then the minimum in (12) can be obtained by solving two minimization problems of the form $\min_{\mathbf{c}\in\mathcal{C}_1} \langle \alpha, \mathbf{c}\rangle$ and one problem of the form $\min_{\mathbf{c}\in\mathcal{C}_2} \langle \beta, \mathbf{c}\rangle$.*
(b) *Let $\mathcal{C} = \mathcal{C}_1 \overline{\oplus}_3 \mathcal{C}_2$. Then the minimum in (12) can be obtained by solving four minimization problems of the form $\min_{\mathbf{c}\in\mathcal{C}_1} \langle \alpha, \mathbf{c}\rangle$ and one problem of the form $\min_{\mathbf{c}\in\mathcal{C}_2} \langle \beta, \mathbf{c}\rangle$.*

Proof. Let n, n_1 and n_2 denote the lengths of $\mathcal{C}$, $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively. Given $\gamma \in \mathbb{R}^n$, let $M = 1 + \sum_{i=1}^n |\gamma_i|$.

(a) Let $\mathcal{C} = \mathcal{C}_1 \oplus_2 \mathcal{C}_2$. Define $\alpha^{(0)} = (\alpha_1^{(0)}, \ldots, \alpha_{n_1}^{(0)})$ and $\alpha^{(1)} = (\alpha_1^{(1)}, \ldots, \alpha_{n_1}^{(1)})$ as follows:

$$\alpha_i^{(0)} = \begin{cases} \gamma_i & i = 1, \ldots, n_1 - 1 \\ M & i = n_1 \end{cases}$$

$$\alpha_i^{(1)} = \begin{cases} \gamma_i & i = 1, \ldots, n_1 - 1 \\ -M & i = n_1 \end{cases}$$

For $j = 0, 1$, determine $\mu_j = \min_{\mathbf{c}\in\mathcal{C}_1} \langle \alpha^{(j)}, \mathbf{c}\rangle$, and a minimum-achieving codeword $\mathbf{c}^{(j)} = (c_1^{(j)}, \ldots, c_{n_1}^{(j)}) \in \mathcal{C}_1$. By choice of M, we have that $c_{n_1}^{(0)} = 0$, while $c_{n_1}^{(1)} = 1$.

Now, define $\beta = (\beta_1, \ldots, \beta_{n_2})$ as follows:

$$\beta_i = \begin{cases} \mu_1 - \mu_0 + M & i = 1 \\ \gamma_{n_1+i-2} & i = 2, \ldots, n_2. \end{cases}$$

Solve for $\widehat{\mu} = \min_{\mathbf{c}\in\mathcal{C}_2} \langle \beta, \mathbf{c}\rangle$, and find a codeword $\hat{\mathbf{c}} = (\hat{c}_1, \ldots, \hat{c}_{n_2}) \in \mathcal{C}_2$ achieving this minimum.

We claim that $\min_{\mathbf{c}\in\mathcal{C}} \langle \gamma, \mathbf{c}\rangle = \mu_0 + \widehat{\mu}$, and that a minimum-achieving codeword in $\mathcal{C}$ is

$$\mathbf{c}_{\min} = \begin{cases} (c_1^{(0)}, \ldots, c_{n_1-1}^{(0)}, \hat{c}_2, \ldots, \hat{c}_{n_2}) & \text{if } \hat{c}_1 = 0 \\ (c_1^{(1)}, \ldots, c_{n_1-1}^{(1)}, \hat{c}_2, \ldots, \hat{c}_{n_2}) & \text{if } \hat{c}_1 = 1. \end{cases}$$

We will first show that for each $\mathbf{c} \in \mathcal{C}$, we have $\langle \gamma, \mathbf{c}\rangle \geq \mu_0 + \widehat{\mu}$. Pick an arbitrary $\mathbf{c} \in \mathcal{C}$. There exists a unique pair of codewords $\mathbf{x} = (x_1, \ldots, x_{n_1}) \in \mathcal{C}_1$, $\mathbf{y} = (y_1, \ldots, y_{n_2}) \in \mathcal{C}_2$ such that $x_{n_1} = y_1$, and $(x_1, \ldots, x_{n_1-1}, y_2, \ldots, y_{n_2}) = \mathbf{c}$. Suppose that $x_{n_1} = y_1 = 0$. We then have

$$\langle \gamma, \mathbf{c}\rangle = \langle \alpha^{(0)}, \mathbf{x}\rangle + \langle \beta, \mathbf{y}\rangle \geq \mu_0 + \widehat{\mu}.$$

Next, suppose that $x_{n_1} = y_1 = 1$. In this case, we have

$$\begin{aligned} \langle \gamma, \mathbf{c}\rangle &= \langle \alpha^{(1)}, \mathbf{x}\rangle - \alpha_{n_1}^{(1)} + \langle \beta, \mathbf{y}\rangle - \beta_1 \\ &= \langle \alpha^{(1)}, \mathbf{x}\rangle - (-M) + \langle \beta, \mathbf{y}\rangle - (\mu_1 - \mu_0 + M) \\ &\geq \mu_1 - (-M) + \widehat{\mu} - (\mu_1 - \mu_0 + M) \;=\; \mu_0 + \widehat{\mu}. \end{aligned}$$

Thus, $\langle \gamma, \mathbf{c}\rangle \geq \mu_0 + \widehat{\mu}$, as desired.

It is now enough to show that $\langle \gamma, \mathbf{c}_{\min}\rangle = \mu_0 + \widehat{\mu}$. By definition of $\mathbf{c}_{\min}$,

$$\langle \gamma, \mathbf{c}_{\min}\rangle = \begin{cases} \langle \alpha^{(0)}, \mathbf{c}^{(0)}\rangle + \langle \beta, \hat{\mathbf{c}}\rangle & \text{if } \hat{c}_1 = 0 \\ \langle \alpha^{(1)}, \mathbf{c}^{(1)}\rangle - (-M) + \langle \beta, \hat{\mathbf{c}}\rangle - (\mu_1 - \mu_0 + M) & \text{if } \hat{c}_1 = 1. \end{cases}$$

In either case, $\langle \gamma, \mathbf{c}_{\min}\rangle = \mu_0 + \widehat{\mu}$.

(b) Let $\mathcal{C} = \mathcal{C}_1 \overline{\oplus}_3 \mathcal{C}_2$. Note that from (A1′)–(A3′) in Definition 4.3, it follows that the restriction of $\mathcal{C}_1$ (resp. $\mathcal{C}_2$) onto its last (resp. first) three coordinates is $\{000, 011, 101, 110\}$.

Define $\alpha^{(j)} = (\alpha_1^{(j)}, \ldots, \alpha_{n_1}^{(j)})$, $j = 0, 1, 2, 3$, as follows: $\alpha_i^{(j)} = \gamma_i$ for $i = 1, \ldots, n_1 - 3$, and

$$\begin{array}{rcrcrcl} \alpha_{n_1-2}^{(0)} & = & \alpha_{n_1-1}^{(0)} & = & \alpha_{n_1}^{(0)} & = & M; \\ \alpha_{n_1-2}^{(1)} & = & -\alpha_{n_1-1}^{(1)} & = & -\alpha_{n_1}^{(1)} & = & M; \\ -\alpha_{n_1-2}^{(2)} & = & \alpha_{n_1-1}^{(2)} & = & -\alpha_{n_1}^{(2)} & = & M; \\ -\alpha_{n_1-2}^{(3)} & = & -\alpha_{n_1-1}^{(3)} & = & \alpha_{n_1}^{(3)} & = & M. \end{array}$$

For $j = 0, 1, 2, 3$, determine $\mu_j = \min_{\mathbf{c} \in \mathcal{C}_1} \langle \alpha^{(j)}, \mathbf{c} \rangle$, and a minimum-achieving codeword $\mathbf{c}^{(j)} = (c_1^{(j)}, \ldots, c_{n_1}^{(j)}) \in \mathcal{C}_1$. By choice of the $\alpha^{(j)}$'s, we have that

$$(c_{n_1-2}^{(j)}, c_{n_1-1}^{(j)}, c_{n_1}^{(j)}) = \begin{cases} 000 & \text{if } j = 0 \\ 011 & \text{if } j = 1 \\ 101 & \text{if } j = 2 \\ 110 & \text{if } j = 3. \end{cases}$$

Now, take $\beta = (\beta_1, \ldots, \beta_{n_2})$ to be

$$\beta_i = \begin{cases} -(\mu_0 + \mu_1 - \mu_2 - \mu_3)/2 + M & i = 1 \\ -(\mu_0 - \mu_1 + \mu_2 - \mu_3)/2 + M & i = 2 \\ -(\mu_0 - \mu_1 - \mu_2 + \mu_3)/2 + M & i = 3 \\ \gamma_{n_1+i-6} & i = 4, \ldots, n_2. \end{cases}$$

Solve for $\widehat{\mu} = \min_{\mathbf{c} \in \mathcal{C}_2} \langle \beta, \mathbf{c} \rangle$, and find a codeword $\hat{\mathbf{c}} = (\hat{c}_1, \ldots, \hat{c}_{n_2}) \in \mathcal{C}_2$ achieving this minimum. It may be verified that $\min_{\mathbf{c} \in \mathcal{C}} \langle \gamma, \mathbf{c} \rangle = \mu_0 + \widehat{\mu}$, and that a minimum-achieving codeword in $\mathcal{C}$ is

$$\mathbf{c}_{\text{min}} = \begin{cases} (c_1^{(0)}, \ldots, c_{n_1-3}^{(0)}, \hat{c}_4, \ldots, \hat{c}_{n_2}) & \text{if } (\hat{c}_1, \hat{c}_2, \hat{c}_3) = 000 \\ (c_1^{(1)}, \ldots, c_{n_1-3}^{(1)}, \hat{c}_4, \ldots, \hat{c}_{n_2}) & \text{if } (\hat{c}_1, \hat{c}_2, \hat{c}_3) = 011 \\ (c_1^{(2)}, \ldots, c_{n_1-3}^{(2)}, \hat{c}_4, \ldots, \hat{c}_{n_2}) & \text{if } (\hat{c}_1, \hat{c}_2, \hat{c}_3) = 101 \\ (c_1^{(3)}, \ldots, c_{n_1-3}^{(3)}, \hat{c}_4, \ldots, \hat{c}_{n_2}) & \text{if } (\hat{c}_1, \hat{c}_2, \hat{c}_3) = 110. \end{cases}$$

The details of the verification are along the lines of that in part (a), and are left to the reader. □

Suppose that we have a PAG code family $\mathfrak{C}$, and must solve (12) for a given length-n code $\mathcal{C} \in \mathfrak{C}$ and cost vector $\gamma = (\gamma_1, \ldots, \gamma_n) \in \mathbb{R}^n$. Note that if $\mathcal{C}$ can be expressed as a direct sum, $\mathcal{C}_1 \oplus \mathcal{C}_2$, of codes $\mathcal{C}_1$ and $\mathcal{C}_2$ of lengths n_1 and n_2, respectively, then

$$\min_{\mathbf{c} \in \mathcal{C}} \langle \gamma, \mathbf{c} \rangle = \min_{\mathbf{c}^{(1)} \in \mathcal{C}_1} \langle \gamma^{(1)}, \mathbf{c}^{(1)} \rangle + \min_{\mathbf{c}^{(2)} \in \mathcal{C}_2} \langle \gamma^{(2)}, \mathbf{c}^{(2)} \rangle,$$

where $\gamma^{(1)} = (\gamma_1, \ldots, \gamma_{n_1})$ and $\gamma^{(2)} = (\gamma_{n_1+1}, \ldots, \gamma_n)$. Therefore, to show that (12) can be solved in time polynomial in n, it is enough to show that there is a polynomial-time algorithm in the case when $\mathcal{C}$ is 2-connected.

It follows from Definition 6.1 that there exists a finite sub-family $\mathfrak{D} \subset \mathfrak{C}$ such that each 2-connected code $\mathcal{C} \in \mathfrak{C}$ has a $\overline{3}$-homogeneous, $(\Gamma \cup \mathfrak{D})$-unary code-decomposition tree, which can be constructed in polynomial time. So, an algorithm for solving (12) for a 2-connected code $\mathcal{C} \in \mathfrak{C}$ would use Lemma B.1 to recursively go down the code-decomposition tree starting from the root node, solving at most four minimization problems at each leaf of the tree. Recall that each leaf of a $(\Gamma \cup \mathfrak{D})$-unary code-decomposition tree is a code in $\Gamma \cup \mathfrak{D}$. Since such a tree for a length-n code can have at most n leaves, the algorithm for solving (12) would run in time polynomial in n, provided that there is a polynomial-time algorithm for solving (12) for codes $\mathcal{C} \in \Gamma$, *i.e.*, graphic codes.

Indeed, there exists a polynomial-time algorithm for solving (12) for graphic codes. Note that the minimization problem $\min \langle \gamma, \mathbf{c} \rangle = \min \sum_{i=1}^n \gamma_i c_i$ over a graphic code $\mathcal{C}(\mathcal{G})$ is equivalent to the problem of finding the minimum-weight Eulerian subgraph in the graph $\mathcal{G}$ whose edges e_i, $i = 1, 2, \ldots, n$, are given the weights γ_i. An Eulerian subgraph of a graph is a subgraph in which each vertex has even degree.

The Eulerian subgraph problem can be solved as follows[8]. Given a subset T of the vertices of $\mathcal{G}$, a *T-join* is a set of edges J of $\mathcal{G}$ such that a vertex v in $\mathcal{G}$ has odd degree with respect to J if and only if $v \in T$. Let N be the set of edges $\{e_i : \gamma_i < 0\}$, and let T be the subset of vertices with odd degree wrt N. Define the graph $\mathcal{G}'$ to be the graph $\mathcal{G}$ but with edge-weights $\gamma_i' = |\gamma_i|$, $i = 1, 2, \ldots, n$. Find a minimum-weight T-join, J, in $\mathcal{G}'$. The symmetric difference $J \Delta N$ is the required minimum-weight Eulerian subgraph of $\mathcal{G}$. A minimum-weight T-join can be found in polynomial time [55, Chapter 29], and so, a minimum-weight Eulerian subgraph of $\mathcal{G}$ can be determined in polynomial time.

Appendix C. Proof of Theorem 6.8

We will first prove that the family, Γ, of graphic codes is not asymptotically good. This is probably a "folk" theorem, but we could not find an explicit proof in the literature. Our proof relies on the fact [56] that for a graph $\mathcal{G} = (V, E)$ with girth g and average degree $\overline{\delta} = 2|E|/|V| \ge 2$, the number of vertices satisfies the so-called *Moore bound*:

$$|V| \ge \begin{cases} 1 + \overline{\delta} \sum_{i=0}^{\lfloor g/2 \rfloor - 1} (\overline{\delta} - 1)^i & \text{if } g \text{ is odd} \\ 2 \sum_{i=0}^{\lfloor g/2 \rfloor - 1} (\overline{\delta} - 1)^i & \text{if } g \text{ is even.} \end{cases}$$

For our purposes, the weaker bound $|V| \ge 2\,(\overline{\delta} - 1)^{\lfloor g/2 \rfloor - 1}$ is enough, as we then have, for $\overline{\delta} > 2$,

$$g \le 4 + \frac{2\log(|V|/2)}{\log(\overline{\delta} - 1)} \le 4 + \frac{2\log(|E|/2)}{\log(\overline{\delta} - 1)}, \tag{13}$$

where, for the sake of concreteness, $\log$ denotes the natural logarithm.

Since codewords in $\mathcal{C}(\mathcal{G})$ correspond to cycles in $\mathcal{G}$, we see that the minimum distance of $\mathcal{C}(\mathcal{G})$ equals the girth g of the graph $\mathcal{G}$. Furthermore, if $\mathcal{G}$ is connected, then (as mentioned in Section 2) the rank of its vertex-edge incidence matrix is $|V| - 1$, and hence, $\dim(\mathcal{C}(\mathcal{G})) = |E| - (|V| - 1)$. Thus, the rate of $\mathcal{C}(\mathcal{G})$ is $1 - (|V| - 1)/|E| \approx 1 - 2/\overline{\delta}$. Consequently, if a family of graphic codes has dimension growing linearly with codelength n, then by (13) their minimum distance grows as $O(\log n)$, which implies that graphic codes are not asymptotically good. This argument is formalized in the proof given below.

Lemma C.1. *The family of graphic codes is not asymptotically good.*

Proof. We will in fact prove a stronger statement: for $r \in (0, 1)$, let $\Gamma_r = \{\mathcal{C} \in \Gamma : \mathcal{C} \text{ has rate} > r\}$; then, for any code $\mathcal{C} \in \Gamma_r$ with length $n \ge 2$, we have

$$d(\mathcal{C}) \le \frac{4 \log n}{\log(1 + r)}. \tag{14}$$

Consider first an $[n, k, d]$ code $\mathcal{C} \in \Gamma_r$ with $n > 2/r$. Without loss of generality (WLOG), we can assume that $\mathcal{C} = \mathcal{C}(\mathcal{G})$ for some connected graph $\mathcal{G} = (V, E)$ [18, Proposition 1.2.8]. Therefore, $k/n = 1 - (|V| - 1)/|E| > r$, or equivalently, $(|V| - 1)/|E| < 1 - r$. Furthermore, since $n > 2/r$, we have that $|V|/|E| < 1 - r/2$. Therefore, the average degree, $\overline{\delta}$, of $\mathcal{G}$ is larger than $2/(1 - r/2)$, which is in turn larger than $2(1 + r/2) = 2 + r$. Hence, by (13),

$$d \le 4 + \frac{2\log(n/2)}{\log(1 + r)} \tag{15}$$

We claim that for $n > 2/r$, we have $4 + \frac{2\log(n/2)}{\log(1+r)} \le \frac{4 \log n}{\log(1+r)}$. Indeed, note that this last inequality is equivalent to $\frac{\log(4n^2)}{\log(1+r)} \ge 4$. A simple calculation will confirm that $\frac{\log(4(2/r)^2)}{\log(1+r)} \ge 4$ for any $r \in (0, 1)$, and hence, for $n > 2/r$, we have $\frac{\log(4n^2)}{\log(1+r)} > \frac{\log(4(2/r)^2)}{\log(1+r)} \ge 4$. Thus, (15) can be replaced by the simpler inequality $d \le \frac{4 \log n}{\log(1+r)}$.

[8] This approach to solving the Eulerian subgraph problem was conveyed to the author by Adrian Vetta.

Therefore, without the assumption $n > 2/r$, we have

$$d(\mathcal{C}) \le \max\left\{2/r,\ \frac{4\log n}{\log(1+r)}\right\}.$$

However, it is straightforward to verify that $(2/r)\log(1+r) \le 2$ for any $r > 0$, from which we obtain that for $n \ge 2$, $\frac{4\log n}{\log(1+r)} > 2/r$, and (14) follows. □

Let $\mathfrak{D}$ be a finite collection of codes, and let $\Gamma + \mathfrak{D}$ be the set of all codes that can be expressed as $\mathcal{C}_1 \oplus_2 \mathcal{C}_2$ or $\mathcal{C}_1 \overline{\oplus}_3 \mathcal{C}_2$, with $\mathcal{C}_1 \in \Gamma$ and $\mathcal{C}_2 \in \mathfrak{D}$. The following result should not be surprising.

Lemma C.2. *For any finite collection of codes $\mathfrak{D}$, the family $\Gamma + \mathfrak{D}$ is not asymptotically good.*

Proof. Define $d_{\max}(\mathfrak{D}) = \max\{d(\mathcal{C}') : \mathcal{C}'$ is a minor of some code in $\mathfrak{D}\}$. We will show that if $\mathcal{C}$ is a code in $\Gamma + \mathfrak{D}$ with rate larger than r, then

$$d(\mathcal{C}) \le \max\left\{d_{\max}(\mathfrak{D}),\ \frac{4\log n}{\log(1+r)}\right\} \tag{16}$$

So, let $\mathcal{C}$ be an $[n,k]$ code in $\Gamma + \mathfrak{D}$ with $k/n > r$. Now, $\mathcal{C} = \mathcal{C}_1 \oplus_2 \mathcal{C}_2$ or $\mathcal{C} = \mathcal{C}_1 \overline{\oplus}_3 \mathcal{C}_2$ for some $\mathcal{C}_1 \in \Gamma$ and $\mathcal{C}_2 \in \mathfrak{D}$. In particular, note that $\mathcal{C}$ must have length at least 4. Suppose first that $\dim(\mathcal{C}_2) \le 2$. Then, $\mathcal{C}_2$ cannot contain a minor equivalent to any of the codes $\mathcal{H}_7$, $\mathcal{H}_7^\perp$, $\mathcal{C}(K_5)^\perp$ and $\mathcal{C}(K_{3,3})^\perp$, since each of these codes has dimension at least 3. So, by Theorem 2.1, $\mathcal{C}_2$ is graphic. Since the family of graphic codes is closed under the operations of 2-sum and $\overline{3}$-sum (see *e.g.* [26, Chapter 8]), it must be that $\mathcal{C}$ is graphic. Therefore, (16) holds by the bound in (14).

If $\dim(\mathcal{C}_2) > 2$, then by Propositions 4.1(b) and 4.8(b), we have that $d(\mathcal{C}) \le d(\mathcal{C}')$ for some minor $\mathcal{C}'$ of $\mathcal{C}_2$. So, once again, (16) holds, this time by definition of $d_{\max}(\mathfrak{D})$. □

Given an almost-graphic code family $\mathfrak{C}$, let $\mathfrak{D}$ be the finite sub-family of codes with the properties guaranteed by Definition 6.1. WLOG, we may assume that $\mathfrak{D}$ is minor-closed.

For $r \in (0,1)$, define N_r to be the least positive integer such that for all $n > N_r$,

$$0 < \frac{1}{\log(1+r-2/n)} < \frac{2}{\log(1+r)}.$$

Note that since $\lim_{n\to\infty} 1/\log(1+r-2/n) = 1/\log(1+r)$, such an N_r does exist. Now, define

$$d_{\max}(r,\mathfrak{D}) = \max\{d(\mathcal{C}) :\ \mathcal{C} \in \mathfrak{C} \text{ and has length at most } N_r, \text{ or } \mathcal{C} \in \mathfrak{D}\}.$$

Note, in particular, that since $\mathfrak{D}$ is taken to be minor-closed, we have $d_{\max}(r,\mathfrak{D}) \ge d_{\max}(\mathfrak{D})$, where $d_{\max}(\mathfrak{D})$ is as defined in the proof of Lemma C.2.

We now have the definitions needed to state the next result, which shows that codes in $\mathfrak{C}$ cannot have both dimension and minimum distance growing linearly with codelength. It is clear that Theorem 6.8 follows directly from this result.

Lemma C.3. *Let $\mathfrak{C}$ be an almost-graphic family of codes. For any $r \in (0,1)$, if $\mathcal{C} \in \mathfrak{C}$ is an $[n,k,d]$ code with $k/n > r$, then*

$$d \le \max\left\{d_{\max}(r,\mathfrak{D}),\ \frac{8\log n}{\log(1+r)}\right\}. \tag{17}$$

Proof. From the definition of $d_{\max}(r,\mathfrak{D})$, and the bounds in (14) and (16), it is obvious that the statement of the lemma holds for all codes in $\Gamma \cup \mathfrak{D} \cup (\Gamma + \mathfrak{D})$. The proof that the statement holds for all codes in $\mathfrak{C}$ is by induction on codelength for a fixed $r \in (0,1)$.

So, fix an $r \in (0,1)$. If n_0 is the smallest length of a non-trivial code in $\mathfrak{C}$, then a length-n_0 code in $\mathfrak{C}$ cannot be decomposed into smaller codes, and so must be in $\Gamma \cup \mathfrak{D}$. Therefore, the statement of the lemma holds for the base case of length-n_0 codes.

Now, suppose that for some $n > n_0$, (17) holds for all codes $\mathcal{C}' \in \mathfrak{C}$ of length $n' \le n-1$ and rate larger than r. Let $\mathcal{C} \in \mathfrak{C}$ be an $[n,k,d]$ code with $k/n > r$. If $\mathcal{C} = \mathcal{C}_1 \oplus \mathcal{C}_2$ for some (non-empty) codes $\mathcal{C}_1$ and $\mathcal{C}_2$ in $\mathfrak{C}$, then at least one of $\mathcal{C}_1$ and $\mathcal{C}_2$ has rate larger than r, and so (17) holds for $\mathcal{C}$ by the induction hypothesis. We may thus assume that $\mathcal{C}$ is 2-connected.

If $\mathcal{C} \in \Gamma \cup \mathfrak{D} \cup (\Gamma + \mathfrak{D})$, there is nothing further to be proved; so we will henceforth assume that this is not the case. So, either $\mathcal{C} = \pi(\mathcal{C}_1 \oplus_2 \mathcal{C}_2)$ or $\pi(\mathcal{C}_1 \,\overline{\oplus}_3\, \mathcal{C}_2)$ for $\mathcal{C}_1, \mathcal{C}_2, \pi$ as in Definition 6.1. WLOG, we may take π to be the identity permutation, so that $\mathcal{C}$ is either $\mathcal{C}_1 \oplus_2 \mathcal{C}_2$ or $\mathcal{C}_1 \,\overline{\oplus}_3\, \mathcal{C}_2$, for some $[n_1, k_1]$ code $\mathcal{C}_1 \in \Gamma \cup \mathfrak{D}$, and some $[n_2, k_2]$ code $\mathcal{C}_2 \in \mathfrak{C}$. Furthermore, $\mathcal{C}_2 \notin \Gamma$, since $\mathcal{C} \notin \Gamma \cup (\Gamma + \mathfrak{D})$. In particular, this means that $k_2 \geq 3$, since if $k_2 \leq 2$, then it would follow from Theorem 2.1 that $\mathcal{C}_2$ is graphic.

We consider the case $\mathcal{C} = \mathcal{C}_1 \oplus_2 \mathcal{C}_2$ first. In this case, Definition 6.1 gives us $k_1 \geq 2$. We have shown above that $k_2 \geq 3$, and so by Proposition 4.1(b), $d \leq \min\{d(\mathcal{C}_1'), d(\mathcal{C}_2')\}$, where $\mathcal{C}_1' = \mathcal{C}_1 \setminus \{n_1\}$ and $\mathcal{C}_2' = \mathcal{C}_2 \setminus \{1\}$. Note that for $i = 1, 2$, $\mathcal{C}_i'$ is an $[n_i', k_i']$ code, where $n_i' = n_i - 1$ and $k_i' = k_i - 1$. Thus, $n = n_1' + n_2'$, and from Proposition 4.1(a), we also have $k = k_1 + k_2 - 1 = k_1' + k_2' + 1$.

Now, if $k_i'/n_i' > r$ for some $i \in \{1, 2\}$, then the statement of the lemma holds for $\mathcal{C}$ by the induction hypothesis. So, we are left with the situation when $k_i'/n_i' \leq r$ for $i = 1, 2$. But in this case, since $k/n = (k_1' + k_2' + 1)/(n_1' + n_2') > r$, we must have $k_1'/n_1' > r - 1/n_1'$; otherwise, we would have $k_2' > (n_1' + n_2')r - 1 - k_1' \geq (n_1' + n_2')r - 1 - (rn_1' - 1) = rn_2'$, which would mean that $k_2'/n_2' > r$. Note that since $\mathcal{C}_1 \in \Gamma \cup \mathfrak{D}$, and $\mathcal{C}_1'$ is a minor of $\mathcal{C}_1$, we have that $\mathcal{C}_1' \in \Gamma \cup \mathfrak{D}$. If $\mathcal{C}_1' \in \mathfrak{D}$ or $n_1' \leq N_r$, then $d(\mathcal{C}_1') \leq d_{\max}(r, \mathfrak{D})$; otherwise, $\mathcal{C}_1'$ is graphic with $n_1' > N_r$, and so, by (14) and the definition of N_r,

$$d(\mathcal{C}_1') \leq \frac{4 \log n_1'}{\log(1 + (r - 1/n_1'))} \leq \frac{4 \log n_1'}{\log(1 + (r - 2/n_1'))} \leq \frac{8 \log n_1'}{\log(1 + r)}.$$

In any case, $d(\mathcal{C}_1') \leq \max\left\{d_{\max}(r, \mathfrak{D}),\ \frac{8 \log n_1'}{\log(1+r)}\right\} \leq \max\left\{d_{\max}(r, \mathfrak{D}),\ \frac{8 \log n}{\log(1+r)}\right\}$. Since $d \leq d(\mathcal{C}_1')$, we have that (17) holds for $\mathcal{C}$.

Finally, we deal with the case when $\mathcal{C} = \mathcal{C}_1 \,\overline{\oplus}_3\, \mathcal{C}_2$. The approach is essentially the same as that in the 2-sum case. This time, we define $\mathcal{C}_1' = \mathcal{C}_1 \setminus \{n_1 - 2, n_1 - 1, n_1\}$ and $\mathcal{C}_2' = \mathcal{C}_2 \setminus \{1, 2, 3\}$. For $i = 1, 2$, we now find that $\mathcal{C}_i'$ is an $[n_i', k_i']$ code, where $n_i' = n_i - 3$ and $k_i' = k_i - 2$. Thus, $n = n_1' + n_2'$, and via Proposition 4.8(a), $k = k_1' + k_2' + 2$. Furthermore, $k_1 \geq 3$ (by Definition 6.1) and $k_2 \geq 3$ (shown above), and so, by Proposition 4.8(b), we have $d(\mathcal{C}) \leq \min\{d(\mathcal{C}_1'), d(\mathcal{C}_2')\}$. If either k_1'/n_1' or k_2'/n_2' is larger than r, then (17) holds for $\mathcal{C}$ by the induction hypothesis. So suppose that $k_i'/n_i' \leq r$ for $i = 1, 2$. Since $k/n = (k_1' + k_2' + 2)/(n_1' + n_2') > r$, we must have $k_1'/n_1' > r - 2/n_1'$; otherwise, we would obtain $k_2'/n_2' > r$. If $\mathcal{C}_1' \in \mathfrak{D}$ or $n_1' \leq N_r$, then $d(\mathcal{C}_1') \leq d_{\max}(r, \mathfrak{D})$; otherwise, $\mathcal{C}_1'$ is graphic with $n_1' > N_r$, and so, by (14) and the definition of N_r,

$$d(\mathcal{C}_1') \leq \frac{4 \log n_1'}{\log(1 + (r - 2/n_1'))} \leq \frac{8 \log n_1'}{\log(1 + r)}.$$

In any case, $d(\mathcal{C}_1') \leq \max\left\{d_{\max}(r, \mathfrak{D}),\ \frac{8 \log n}{\log(1+r)}\right\}$, and since $d \leq d(\mathcal{C}_1')$, we see that (17) holds for $\mathcal{C}$. The proof of the lemma is now complete. □

Acknowledgments

The author would like to thank Jim Geelen for patiently answering the author's questions pertaining to the structure theory of matroids, Adrian Vetta for the polynomial-time algorithm for solving the minimum-weight Eulerian subgraph problem, and Alexander Vardy for discussions leading up to this work. Finally, the author is grateful to the anonymous reviewers for suggesting several improvements to the presentation, and especially for catching an error in an earlier proof of Theorem 6.8.

REFERENCES

[1] C. Greene, “Weight enumeration and the geometry of linear codes,” *Studia Appl. Math.*, vol. 55, pp. 119–128, 1976.

[2] P.J. Cameron, “Polynomial aspects of codes, matroids and permutation groups,” lecture notes, March 2002.

[3] A. Barg, “The matroid of supports of a linear code,” *Appl. Algebra Engrg. Comm. Comput.*, vol. 8, no. 2, pp. 165–172, 1997.

[4] R. Dougherty, C. Freiling and K. Zeger, “Networks, matroids, and non-Shannon information inequalities,” *IEEE Trans. Inform. Theory*, vol. 53, no. 6, pp. 1949–1969, June 2007.

[5] J. Feldman, M.J. Wainwright and D.R. Karger, “Using linear programming to decode binary linear codes,” *IEEE Trans. Inform. Theory*, vol. 51, no. 3, pp. 954–972, 2005.

[6] P.D. Seymour, “Decomposition of regular matroids,” *J. Combin. Theory, Series B*, vol. 28, pp. 305–359, 1980.

[7] P.D. Seymour, “Matroids and multicommodity flows,” *Europ. J. Combin.*, vol. 2, pp. 257–290, 1981.

[8] M. Grötschel and K. Truemper, “Decomposition and optimization over cycles in binary matroids,” *J. Combin. Theory Series B*, vol. 46, pp. 306–337, 1989.

[9] 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.

[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] D. Jungnickel and S.A. Vanstone, “Graphical codes revisited,” *IEEE Trans. Inform. Theory*, vol. 43, pp. 136–146, Jan. 1997.

[12] G.D. Forney, Jr., “Dimension/length profiles and trellis complexity of linear block codes,” *IEEE Trans. Inform. Theory*, vol. 40, no. 6, pp. 1741–1752, Nov. 1994.

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

[14] N. Kashyap, “On codes of bounded trellis complexity,” *Proc. 2007 IEEE Inform. Theory Workshop (ITW 2007)*, Lake Tahoe, CA, USA, pp. 168–173, Sept. 2007.

[15] A.J. Hoffman and J.B. Kruskal, “Integral boundary points of convex polyhedra,” in *Linear Inequalities and Related Systems*, H.W. Kahn and A.W. Tucker (eds.), pp. 223–246, Princeton Univ. Press, Princeton, NJ, 1956.

[16] P.O. Vontobel and R. Koetter, “Graph-cover decoding and finite-length analysis of message-passing iterative decoding of LDPC codes,” preprint submitted to *IEEE Trans. Inform. Theory*, Dec. 2005. ArXiv e-print cs.IT/0512078.

[17] F.J. MacWilliams and N.J.A. Sloane, *The Theory of Error-Correcting Codes*, North-Holland, Amsterdam, 1977.

[18] J.G. Oxley, *Matroid Theory*, Oxford University Press, Oxford, UK, 1992.

[19] H. Whitney, “On the abstract properties of linear dependence,” *Amer. J. Math.*, vol. 57, pp. 509–533, 1935.

[20] J.G. Bredeson and S.L. Hakimi, “Decoding of graph theoretic codes,” *IEEE Trans. Inform. Theory*, vol. IT-13, pp. 348–349, 1967.

[21] W.T. Tutte, “Matroids and graphs,” *Trans. Amer. Math. Soc.*, vol. 90, pp. 527–552, 1959.

[22] R.E. Bixby and W.H. Cunningham, “Converting linear programs to network problems,” *Math. Oper. Res.*, vol. 5, pp. 321–357, 1980.

[23] R.E. Bixby and D.K. Wagner, “An almost linear-time algorithm for graph realization,” *Math. Oper. Res.*, vol. 13, no. 1, pp. 99–123, 1988.

[24] S. Fujishige, “An efficient PQ-graph algorithm for solving the graph realization problem,” *J. Comput. System Sci.*, vol. 21, pp. 63–86, 1980.

[25] W.T. Tutte, “An algorithm for determining whether a given binary matroid is graphic,” *Proc. Amer. Math. Soc.*, vol. 11, pp. 905–917, 1960.

[26] K. Truemper, *Matroid Decomposition*, Academic Press, San Diego, 1992.

[27] B. Bollobás, *Modern Graph Theory*, Springer, New York, 1998.

[28] W.T. Tutte, “Connectivity in matroids,” *Canad. J. Math.*, vol. 18, pp. 1301–1324, 1966.

[29] G.D. Forney, Jr., “Coset codes II: Binary lattices and related codes,” *IEEE Trans. Inform. Theory*, vol. 34, no. 5, pp. 1152–1187, Sept. 1988.

[30] G.D. Forney, Jr. and M.D. Trott, “The dynamics of group codes: State spaces, trellis diagrams and canonical encoders,” *IEEE Trans. Inform. Theory*, vol. 39, no. 5, pp. 1491–1513, Sept. 1993.

[31] G.B. Horn and F.R. Kschischang, “On the intractability of permuting a block code to minimize trellis complexity,” *IEEE Trans. Inform. Theory*, vol. 42, no. 6, pp. 2042–2048, Nov. 1996.

[32] F.R. Kschischang and V. Sorokine, “On the trellis structure of block codes,” *IEEE Trans. Inform. Theory*, vol. 41, pp. 1924–1937, Nov. 1995.

[33] W.H. Cunningham, *A combinatorial decomposition theory*, Ph.D. thesis, Univ. of Waterloo, 1973.

[34] J. Edmonds, “Matroid intersection,” in *Discrete Optimization I*, P.L. Hammer, E.L. Johnson and B.H. Korte (eds.), Ann. Discrete Math., vol. 4, pp. 39–49, North-Holland, Amsterdam, 1979.

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

[36] 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.

[37] J. Geelen, B. Gerards and G. Whittle, “Towards a matroid-minor structure theory,” in *Combinatorics, Complexity and Chance. A Tribute to Dominic Welsh*, G. Grimmett and C. McDiarmid, eds., Oxford University Press, 2007.

[38] W.T. Tutte, “A homotopy theorem for matroids, I, II,” *Trans. Amer. Math. Soc.*, vol. 88, pp. 144–174, 1958.

[39] K. Truemper, “A decomposition theory for matroids. V. Testing of matrix total unimodularity,” *J. Comb. Theory, Series B*, vol. 49, pp. 241–281, 1990.

[40] P.H. Siegel and M.H. Taghavi, “Adaptive linear programming decoding,” *Proc. 2006 IEEE Int. Symp. Inform. Theory (ISIT 2006)*, Seattle, WA, pp. 1374–1378, July 2006.

[41] F. Barahona and M. Grötschel, “On the cycle polytope of a binary matroid,” *J. Comb. Theory Ser. B*, vol. 40, pp. 40–62, 1986.

[42] K. Truemper, “A decomposition theory for matroids. I. General results,” *J. Comb. Theory, Series B*, vol. 39, pp. 43–76, 1985.

[43] K. Truemper, “A decomposition theory for matroids. IV. Decomposition of graphs,” *J. Comb. Theory, Series B*, vol. 45, pp. 259–292, 1988.

[44] S.C. Ntafos and S.L. Hakimi, “On the complexity of some coding problems,” *IEEE Trans. Inform. Theory*, vol. IT-27, no. 6, pp. 794–796, Nov. 1981.

[45] J. Edmonds, “Paths, trees and flowers,” *Canad. J. Math*, vol. 17, pp. 449–467, 1965.

[46] J. Edmonds and E.L. Johnson, “Matching, Euler tours and the Chinese postman,” *Math. Programming*, vol. 5, pp. 88–124, 1973.

[47] A. Vardy, “The intractability of computing the minimum distance of a code,” *IEEE Trans. Inform. Theory*, vol. 43, no. 6, pp. 1757–1766, Nov. 1997.

[48] A. Itai and M. Rodeh, “Finding a minimum circuit in a graph,” *SIAM J. Computing*, vol. 7, no, 4, pp. 413–423, 1978.

[49] P.O. Vontobel and R. Koetter, “On the relationship between linear programming decoding and min-sum algorithm decoding,” *Proc. ISITA 2004*, Parma, Italy, pp. 991-996, October 10–13, 2004.

[50] J. Geelen, B. Gerards and G. Whittle, “Branch-width and well-quasi-ordering in matroids and graphs,” *J. Combin. Theory Ser. B*, vol. 84, pp. 270–290, 2002.

[51] P. Hliněný and G. Whittle, “Matroid tree-width,” *Europ. J. Comb.*, vol. 27, pp. 1117–1128, 2006.

[52] N. Robertson and P.D. Seymour, “Graph Minors. XIII. The disjoint paths problem,” *J. Combin. Theory, Series B*, vol. 63, pp. 65–110, 1995.

[53] N. Robertson and P.D. Seymour, “Graph Minors. XX. Wagner’s Conjecture,” *J. Combin. Theory, Series B*, vol. 92, pp. 325–357, 2004.

[54] L. Lovasz, “Graph minor theory,” *Bull. Amer. Math. Soc.*, vol. 43, pp. 75–86, Jan. 2006.

[55] A. Schrijver, *Combinatorial Optimization: Polyhedra and Efficiency*, Springer-Verlag, Berlin, 2003.

[56] N. Alon, S. Hoory and N. Linial, “The Moore bound for irregular graphs,” *Graphs Combin.*, vol. 18, pp. 53–57, 2002.