Algebraic Algorithms for Vector Network Coding

Javad Ebrahimi and Christina Fragouli
EPFL, Lausanne, Switzerland.
Email: {javad.ebrahimi,christina.fragouli}@epfl.ch

***Abstract*—We develop new algebraic algorithms for scalar and vector network coding. In vector network coding, the source multicasts information by transmitting vectors of length L, while intermediate nodes process and combine their incoming packets by multiplying them with $L \times L$ coding matrices that play a similar role as coding coefficients in scalar coding. We start our work by extending the algebraic framework developed for multicasting over graphs in [1] to include operations over matrices and to admit both graphs and linear deterministic networks (which approximate wireless channels) as special cases. Our algorithms build on this generalized framework, and provide as special cases unicast and multicast algorithms for deterministic networks, as well as network code designs using structured matrices.**

I. Introduction

In the seminal paper [1], Koetter and Medard translated the problem of network code design to an algebraic problem. This algebraic framework provided the theoretical foundation for the development of most of the finite length coding results and practical algorithms for network coding today. In this paper, we show that this framework naturally extends to incorporate vector communication, and can be applied not only on graphs, but also on linear deterministic networks, which form a generalization of regular graphs and model wireless channels. Building on this extended framework, we propose new algebraic code designs for scalar and vector network coding, that apply both over networks represented as graphs, as well as arbitrary linear deterministic networks.

In scalar network coding, when multicasting to N receivers at rate h, the source transmits h scalar values over some finite field. The size of the employed finite field is a design parameter, that for complexity considerations we typically desire to minimize. Intermediate network nodes linearly combine their received symbols by multiplying them with scalar coefficients, called coding coefficients in the literature. The network code design consists of selecting the coding coefficients, and the finite field of operation, so that each receiver has a full rank set of equations to solve and can thus retrieve the source symbols [1].

In vector network coding, the source transmits h vectors of length L, where the elements of the vectors are over a fixed finite field $\mathbb{F}_q$, for example, the binary field $\mathbb{F}_2$. Intermediate network nodes perform coding operations over vectors, namely, multiply their incoming vectors with $L \times L$ coding matrices and then add them to create the new vectors that they propagate towards the destinations. That is, intermediate nodes linearly combine their incoming vectors using coding matrices, where these matrices play a similar role as scalar coding coefficients in traditional algebraic network coding. The code design consists in selecting the length L and the $L \times L$ coding matrices so that each receiver receives information at rate h. Scalar network coding over a field of size $\mathbb{F}_q$ can be viewed as a special case of vector network coding with $L = 1$.

Vector network coding offers a natural generalization of network coding, and thus offers a larger space of choices for optimizing cost parameters, such as the operational complexity, or the communication block length. We show through specific examples in Section VI that using vector coding we can achieve coding solutions not possible with scalar network coding. An alternative motivation to study it is that, scalar network coding may require operations over a finite field $\mathbb{F}_q$ of size q that grows with the number of receivers; however, in some networks, operations may be restricted to be over a fixed field, while the number of receivers grows arbitrarily. This can occur in practical networks, where intermediate nodes might need to be equipped in advance with the capability of operations over a fixed field; moreover, it naturally occurs over deterministic networks, where the finite field of operation is predetermined and dictated by the network model itself.

Our code designs offer finite length designs for multicasting in deterministic networks. Deterministic network models have recently attracted a significant interest in the literature, in particular in the context of serving as approximate characterizations for wireless networks [9]–[11]. The work in [9], [10] established an information theoretical min-cut max-flow theorem for deterministic networks, showing that the unicast capacity equals the minimum rank of the transfer matrices of cuts separating the source from the destination. In the same work, the authors also extended to deterministic networks the multicasting theorem in network coding [7], and showed that in the presence of N receivers, if the mincut to each receiver equals at least h, it is possible to multicast information to all receivers at this rate. The proof uses information theoretical arguments and does not directly lead to polynomial time designs. The vector coding algorithms we develop in this paper allow to achieve the min-cut capacity when multicasting over arbitrary deterministic networks, using finite block lengths and with polynomial complexity.

We start our work by extending the algebraic framework developed for multicasting over graphs in [1] in two ways: (i) to include operations over matrices and (ii) to admit both graphs and linear deterministic networks as special cases. Building on this generalized framework, our contributions include:

1) We provide a polynomial time algorithm for the design

of coding matrices of vector network coding when multicasting to N receivers. Our metric of optimization is the smallest size L such that there exist $L \times L$ coding matrices that allow all N destinations to successfully decode the source information. The size of these matrices plays the same role as the size of the finite field in traditional network coding. Our algorithm reduces the problem of finding a small size L to the problem of finding a small degree coprime factor of an algebraic polynomial, and leads to solutions not possible with using scalar network coding, as illustrated through examples in Sections V and VI.

2) Both for the case of graphs and deterministic networks, we show that $L \approxeq \log(N(\log N - 1)h\Lambda)$ is always sufficient, where Λ is a network parameter, and we can find such matrices in polynomial time. We also provide probabilistic guarantees, and show for example that in a fraction $\frac{1023}{1024}$ of polynomials derived from transfer functions, we will be able to find in polynomial time binary coding matrices of size at most 3×3 that lead to a valid code.
3) For the case of deterministic networks, our approach offers the first code design for multicasting over arbitrary linear deterministic networks. Moreover, as a special case, it gives an alternative algorithm for a unicast connection over a deterministic network, adding to the existing algorithms [12]–[16].
4) Our approach also gives a new algorithm for scalar network code design, that operates in polynomial time. This new algorithm jointly minimizes the employed field size while selecting the coding coefficients. In contrast, existing algorithms [1]–[4], [6], [31] first select a fixed finite field and then proceed to design the network codes over this predetermined field. As a consequence, these algorithms would operate over a field of size N, i.e., the worst case guarantee. However, our algorithm, due to jointly optimizing the effective field size as well as the coefficients, can result in a much smaler field while still being a polynomial time algorithm. A theoretical side-result of our work is establishing a connection between the problem of identifying the minimum field size required for network coding and finding the smallest coprime factor of algebraic polynomials.

The paper is organized as follows. Section II formalizes our notation and problem statement, and briefly reviews linear deterministic networks. Section III presents the generalized algebraic framework that includes vector coding and linear deterministic networks as special cases. Section IV presents our proposed code designs for vector and scalar network coding, while Section V illustrates through examples the application of vector network coding and of our developed design tools. Section VI defines a notion of equivalence between scalar and vector coding, and offers a comparison between the two. Section VII discusses related work, while finally Section VIII concludes the paper.

II. Linear Deterministic Networks and Problem Formulation

In this paper, we consider the problem of multicasting a common rate h to N receivers, over linear deterministic networks. These networks, as we will argue, form generalizations of graphs. Although in this paper we focus on the problem of multicasting, we believe that the tools we develop might be useful for other traffic scenaria as well. Trivially, we get as a special case algorithms for a unicast connection over deterministic networks, as we discuss in Section IV-D.

A. Graphs and Linear Deterministic Networks

Let $G = (V, E)$ denote a directed graph with unit capacity edges. We can think of each edge of this graph as a channel *orthogonal* to all other channels, where each channel (edge) has a single input and a single output, and can be used to send a single bit from the input to the output (unit capacity). We can then depict a node with multiple incoming and outgoing edges as having multiple inputs and multiple outputs, as determined by its adjacent edges, where inputs and outputs can be arbitrarily connected to each other within the node. For example, Fig. 1(a) depicts a node in a directed graph, and Fig. 1(b) the equivalent representation of this node.

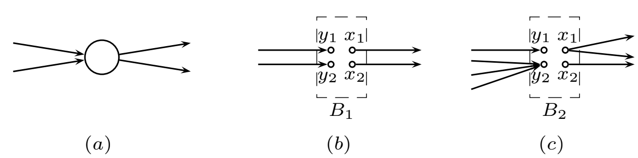

Fig. 1. (a) A node in a directed graph, (b) equivalent representation through orthogonal channels, and (c) a node in a network of deterministic channels.

Deterministic network models have recently attracted a significant interest in the literature, in particular in the context of serving as approximate characterizations for wireless networks [9]–[11]. Wireless relay networks cannot be represented as graphs, due to the inherently shared nature of the wireless medium that causes complex signal interactions. In the wireless medium, transmissions are broadcasted, and may be received by multiple receivers at different signal strengths depending on path loss parameters. Moreover, there is interference between transmissions, and the signal from different nodes in the network can be received superimposed at very different power at a given receivers. To take into account broadcasting and interference, recent work in the literature proposes to use linear binary deterministic networks that take into account the signal interactions and represents the noise by a deterministic threshold rather than a random variable [9]–[11]. The argument is that for high Signal-to-Noise-Ratio (SNR), it is the signal interactions that will dominate the performance, and thus the capacity of the deterministic could be very close to that of the noisy network. Thus networks of deterministic channels could be used as approximate models for wireless networks.

An example of a linear deterministic network is depicted in Fig. 2. Each deterministic network has a set of nodes V, a

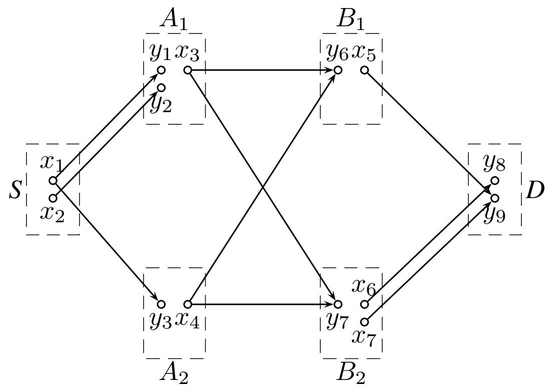

Fig. 2. A deterministic network where a source S transmits information to a single receiver D.

set of edges E, and is associated with a fixed finite field of operation $\mathbb{F}_q$. Each node in the deterministic network consists of two sets, the set of inputs and the set of outputs of the node. We will generally denote inputs and outputs using the variables x and y, respectively. Each channel is an edge $e = (x, y)$ between an input of a node and an output at a different node. With edge e is associated a fixed nonzero coefficient c_e that takes values over the finite field of operation. For deterministic networks over the binary field ($\mathbb{F}_2$), each edge is associated with a coefficient of value one, i.e., $c_e = 1$. For deterministic networks over the ternary field ($\mathbb{F}_3$), an edge e is associated with a value c_e in the set $\{1, 2\}$.

To model broadcasting, when transmitting an input x, all channel outputs y connected to x receive the same value, multiplied by the edge coefficient. For example, in Fig. 2, that depicts a binary deterministic network, input x_1 is received by outputs y_1 and y_3. Interference is modeled through addition over the finite field of operation $\mathbf{F}_q$. In Fig. 2 the bit y_6 equals the binary addition (xor) of bits x_3 and x_4.

A cut between a source S and destination D is a partition of the nodes into parts $\mathcal{C}$ and $\bar{\mathcal{C}}$ with $S \in \mathcal{C}$ and $D \in \bar{\mathcal{C}}$. Let $U = \{(x, y) : \ x \in \mathcal{C},\ y \in \bar{\mathcal{C}}\}$ be the set of edges that cross the cut (cut edges). For example, in Fig. 2, if we consider the cut with $\mathcal{C} = \{S, A_1, A_2\}$ and $\bar{\mathcal{C}} = \{B_1, B_2, D\}$ then the set $\mathcal{U} = \{(x_3, y_6),\ (x_3, y_7),\ (x_4, y_6),\ (x_4, y_7)\}$ collects the edges that cross the cut. If we collect all inputs in the cut in a row vector $\mathbf{x}$ and the outputs in the cut in a vector $\mathbf{y}$ they are related through a linear transformation, $\mathbf{y} = \mathbf{T}\mathbf{x}$. The transformation matrix $\mathbf{T}$ associated with the set of edges U is a binary matrix with rows corresponding to the inputs $\mathbf{x}$, columns corresponding to the outputs $\mathbf{y}$, and value c_e at position (i, j) if and only if edge $e = (x_i, y_j)$ exists. In our example in Fig. 2, for $\mathcal{U} = \{(x_3, y_6),\ (x_3, y_7),\ (x_4, y_6),\ (x_4, y_7)\}$, the transfer matrix equals to

$$\mathbf{T} = \begin{array}{ccc} & y_6 & y_7 \\ x_3 & 1 & 1 \\ x_4 & 1 & 1 \end{array}, \text{ and mincut=rank}(\mathbf{T}) = 1.$$

Indeed, $y_6 = y_7 = x_3 + x_4$, and thus one only independent bit can be transmitted through this cut to the destination.

In [9], [10] it is shown that the information theoretic value [32] of a cut $\mathcal{C}$ between a source-destination pair equals the rank of the transformation matrix $\mathbf{T}$ associated with the edges crossing the cut (measured in symbols over the finite field of the deterministic model). The min-cut value equals the min such value among all possible cuts.

In [9], [10] it is also proved that information rate equal to the min-cut value is achievable using information theoretical tools, that is, infinite block processing. In the same work, the authors also extended to deterministic networks the multicasting theorem in network coding [7], [8], and showed that in the presence of N receivers, if the mincut to each receiver equals at least h, it is possible to multicast information to all receivers at this rate. The proof again uses information theoretical tools. For deterministic networks, polynomial time algorithms that allow unicast information transmission were developed in [12]–[16].

Directed graphs with unit capacity edges form a special case of linear deterministic networks, where each input and each output are associated with exactly one edge, and each edge is associated with value equal to one. In this case, it is straightforward to see that the transfer matrix has rank equal to the number of edges that cross the cut.

B. Problem Formulation

We consider a deterministic network, a source S, and N receivers, where the min-cut value from the source to each receiver equals at least h. We are interested in designing scalar and vector network coding algorithms that allow to multicast rate h to all receivers over linear deterministic networks. In this paper we present our designs for binary vector coding and binary deterministic networks; the extension over arbitrary fields is straightforward, and we briefly discuss it in Section IV-A.

Both in the cases of graphs and deterministic networks, we can in general perform arbitrary operations inside the network nodes; i.e., each network node can collect its outputs, and send an abritrary function of these outputs through each input. However, in this paper we will restrict our attention to linear operations, namely, each network code can perform linear combining with scalar coefficients for scalar network coding, and linear combining with matrix coefficients for vector network coding.

Multicasting algorithms over graphs typically require, as a first step, to identify for each destination a set of h edge-disjoint paths, where h is the multicasting rate [31]. These paths may overlap, and linear coding for multicasting effectively resolves what linear combination is sent through these common edges where the overlap occurs. We will here for simplicity also adopt this approach, although, as we will argue in Section IV-D, it is not necessary.

Thus, similarly to graphs, for multicasting over deterministic networks, we assume as a first step that an algorithm for unicasting identifies, for each receiver, what are inputs and the outputs used in each layer [12]–[16], and ignore the remaining

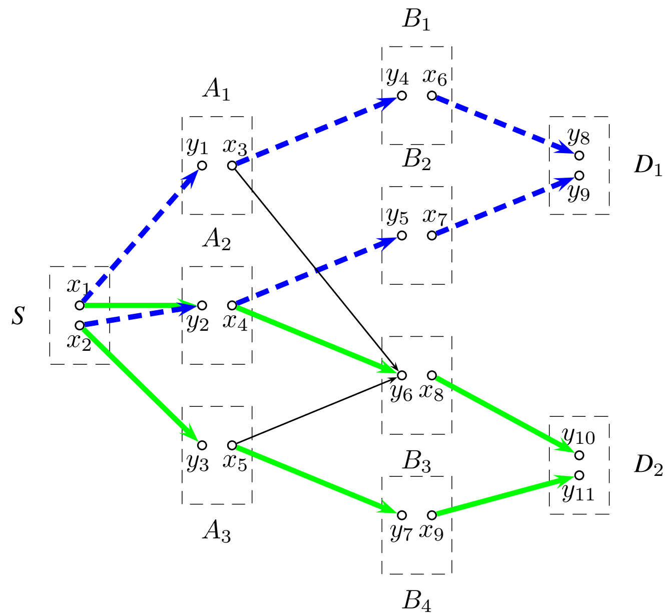

Fig. 3. A deterministic network with a source S multicasting information at rate $h = 2$ to $N = 2$ receivers. The paths to the first receiver are depicted in dashed lines, and the paths to the second receiver in bold.

inputs and outputs. For example, in Fig. 3, we depict with dashed lines the paths from the source to the receiver D_1. Our multicasting algorithm uses this information as input, and is independent of the exact algorithm that has been used to identify the unicast connections (namely, any of the algorithms in [12]–[16] can be used). Similarly to the case of graphs, when multiple paths share inputs (edges), we will transmit coded information.

Note that in the case of deterministic networks, even if different unicast connections do not use the same edges (inputs-outputs), due to broadcasting may still destructively interfere with each other, as the following example illustrates. This example also shows that, when multicasting to two receivers over deterministic networks, it is not always sufficient to perform coding operations over the binary field; in contrast, in graphs, when multicasting to two receivers, binary operations are always sufficient [22]. This difference stems exactly from the fact that in deterministic networks the network itself imposes coding operations.

Example II.1. *Consider the network in Fig. 3, where a source S multicasts information at rate $h = 2$ to $N = 2$ receivers. The paths to the two receivers are depicted in bold and dashed respectively. Let u_1 and u_2 denote the information bits, and assume we attempt to use one bit processing. The source creates and transmits the bits x_1 and x_2 as linear combinations of u_1 and u_2.*

Notice that $x_3 = y_1 = x_1$, $x_4 = y_2 = x_1 + x_2$ and $x_5 = y_3 = x_2$. Node A_1 is obliged to transmit x_3, otherwise D_1 will only receive one bit. Similarly, node A_2 also needs to transmit x_4, otherwise node D_1 will only receive one bit. Finally, node A_3 needs to transmit x_5, otherwise node D_2 will receive only one bit. But then, $y_6 = x_3 + x_4 + x_5 = x_1 + x_1 + x_2 + x_2 = 0$, i.e., is always identically zero.

We conclude that in this network, it is impossible to transmit rate two to both receivers, with one-bit processing. We provide a solution with vector coding for this network in Example III.3. □

For multicasting over graphs, a class of algorithms developed for polynomial time code design, impose a partial-order and sequentially visit the edges of the network (see for example [2]). Such algorithms cannot be directly applied over deterministic networks. This is because, partial order algorithms make decisions taking into account the coding operations only in the already visited edges; however, in deterministic networks, we also need to take into account coding operations by the network itself, and subsequent network coding operations can affect what are current viable decisions.

III. Algebraic Framework

We here first review the algebraic framework for scalar network coding over graphs in [1]; we then proceed to discuss how it can be extended to incorporate vector network coding; finally, we discuss how it can be applied over linear deterministic networks.

A. Scalar coding over graphs

In scalar coding, the source has h symbols $\{u_1, \ldots, u_h\}$ in a field $\mathbb{F}_q$ to multicast to N receivers. Intermediate nodes linearly combine their received information using coding coefficients $\{X_k\}$ from the same field $\mathbb{F}_q$.

Consider a directed acyclic graph $G = (V, E)$, and each edge of the graph as a memory element that stores an intermediate information symbol. Assume that the source node has also h auxiliary incoming edges, each bringing one of the h symbols $\{u_1, \ldots, u_h\}$. Then, if we consider a specific receiver j, our network acts as a linear system with h inputs (the h source symbols), h outputs (that the receiver observes), and $m := |E|$ memory elements. This system is described by the following set of finite-dimensional state-space equations:

$$\begin{aligned} \mathbf{s}_{k+1} &= \mathbf{A}\mathbf{s}_k + \mathbf{B}\mathbf{u}_k \\ \mathbf{y}_k &= \mathbf{C}_j\mathbf{s}_k + \mathbf{D}_j\mathbf{u}_k, \end{aligned} \tag{1}$$

where $\mathbf{s}_k$ is the $m \times 1$ state vector, $\mathbf{y}_k$ is the $h \times 1$ output vector, $\mathbf{u}_k$ is the $h \times 1$ input vector, and $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}_j$, and $\mathbf{D}_j$ are matrices with appropriate dimensions which we discuss in the following.

Matrix $\mathbf{A}$ of dimension $m \times m$ is common for all receivers and reflects the network topology, the way the edges (memory elements) are connected. An element $\mathbf{A}_{ij}$ is nonzero if and only if the directed edge j points at a node v, and edge i starts from the same node v. In particular, $\mathbf{A}_{ij}$ is equal to 1 if node v has only one incoming edge (the edge j) since in this case the information from edge j has to flow in edge i. If there are multiple incoming edges in node v, then $\mathbf{A}_{ij}$ is equal to an unknown variable in the set $\{X_k\}$, the coefficient that will multiply the information symbol from edge j as it flows through edge i. We will assume that we have in total ν such coefficients; clearly, $\nu \leq m^2$. Matrix $\mathbf{B}$ is also common for all receivers and reflects the way the inputs (sources) are connected to our graph. Matrices $\mathbf{C}_j$ and $\mathbf{D}_j$ respectively express how the outputs receiver j observes depend upon the state variables and the inputs. Matrices $\mathbf{B}$, $\mathbf{C}_j$, and $\mathbf{D}_j$ can be chosen to be binary matrices by possibly introducing auxiliary vertices and edges. Without loss of generality, we will assume that $\mathbf{D}_j = \mathbf{0}$ (this we can again do by possibly adding auxiliary edges and increasing the size m of the state space).

A standard result in linear system theory gives us the transfer matrix $\mathbf{G}_j(\mathcal{D})$:

$$\mathbf{G}_j(\mathcal{D}) = \mathbf{C}_j(\mathcal{D}^{-1}\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}, \tag{2}$$

where $\mathcal{D}$ is the indeterminate delay operator. Using unit delay, we obtain the $h \times h$ transfer matrix for receiver j:

$$\mathbf{M}_j = \mathbf{C}_j(\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}. \tag{3}$$

By accordingly ordering the elements of the state space vector, matrix $\mathbf{A}$ becomes strictly upper triangular for acyclic graphs, and therefore, nilpotent, that is, $A^n = 0$ for some positive integer n. Let Λ denote the length of the longest path between the source and a receiver. Then $\mathbf{A}^{\Lambda+1} = 0$. In other words,

$$(\mathbf{I} - \mathbf{A})^{-1} = \mathbf{I} + \mathbf{A} + \mathbf{A}^2 + \ldots + \mathbf{A}^{\Lambda}. \tag{4}$$

This equation immediately implies that the elements of the transfer matrices $\mathbf{M}_j$ are multivariate polynomials in the unknown variables $\{X_k\}$, of degree at most Λ in each variable, and also of total degree at most Λ. By total degree we refer to the maximum sum of the degrees across the variables in each monomial; for example the polynomial $f(X_1, X_2) = X_1^2X_2 + X_1 + X_1X_2^4$ has three monomials, total degree 5, degree 2 in the variable X_1 and degree 4 in the variable X_2.

Network code design amounts to selecting values in $\mathbf{F}_q$ for the variable entries $\{X_k\}$ in $\mathbf{A}$, so that all matrices $\mathbf{M}_j$, for $j = 1 \ldots N$, are simultaneously full rank. Equivalently, if the determinant of the matrix $\mathbf{M}_j$ is the multivariate polynomial $f_j(X_1, X_2, \ldots, X_\nu)$, we want to select values for the variables $\{X_k\}$ in $\mathbf{F}_q$ so that all the polynomial f_j evaluate to a nonzero value. Note that the degree of this polynomial in each variable equals at most $h\Lambda$, and the total degree as well equals $h\Lambda$.

The matrices $\mathbf{M}_j$, for $j = 1 \ldots N$, are simultaneously full rank if and only if the matrix

$$\mathbf{M} \triangleq \mathbf{M}_1 \cdot \mathbf{M}_2 \cdots \; \ldots \; \cdots \mathbf{M}_N \tag{5}$$

is full rank. Thus, we can equivalently select values for the variables $\{X_k\}$ so that the polynomial

$$f(X_1, X_2, \ldots, X_\nu) = \det\mathbf{M} \tag{6}$$

of total degree at most $Nh\Lambda$ evaluates to a nonzero value.

It can be shown in [3], [4] that $\mathbf{M}_j$ is full rank if and only if the matrix

$$\bar{\mathbf{M}}_j = \begin{bmatrix} \mathbf{C}_j & \mathbf{0} \\ \mathbf{I} - \mathbf{A} & \mathbf{B} \end{bmatrix} \tag{7}$$

is full rank. Let

$$\bar{\mathbf{M}} \triangleq \bar{\mathbf{M}}_1 \cdot \bar{\mathbf{M}}_2 \cdots \; \ldots \; \cdots \bar{\mathbf{M}}_N \tag{8}$$

and

$$\bar{f}(X_1, \ldots, X_\nu) = \det(\bar{\mathbf{M}}) \tag{9}$$

be the corresponding determinant. The polynomial $\bar{f}_j$ has degree at most N in each of the variables; however it may have a total degree as large as Nm, where $m = |E|$ is the number of edges in the graph. The algebraic code design can equivalently be expressed as selecting values for the variables so that the polynomial $\bar{f}$ evaluates to a nonzero value. Three algebraic formulations for the multicasting problem are summarized in the following table. All these formulations are equivalent.

Scalar Algebraic Formulations [1]:

(1) Select a finite field $\mathbb{F}_q$ and values for the variables $\{X_k\}$ from the field $\mathbf{F}_q$ so that all matrices $\mathbf{M}_j$ become simultaneously full rank.

(2) Select a finite field $\mathbb{F}_q$ and values for the variables $\{X_k\}$ from the field $\mathbf{F}_q$ so that the polynomial $f(X_1, \ldots, X_\nu)$ in (17) of degree at most $h\Lambda N$ in each variable and of total degree at most $h\Lambda N$, evaluates to a nonzero value.

(3) Select a finite field $\mathbb{F}_q$ and values for the variables $\{X_k\}$ from the field $\mathbf{F}_q$ so that the polynomial $\bar{f}(X_1, \ldots, X_\nu)$ in (9) of degree at most N in each variable and of total degree at most mN, evaluates to a nonzero value.

From the sparse zero lemma [4], [28], we can assign to the variables X_k values in a finite field $\mathbb{F}_q$ of size larger than N so that all transfer matrices $\mathbf{M}_j$ are simultaneously invertible. Provided that $q > N$, we can find such values deterministically in polynomial time, for example using the methods [1]–[4], [6]. The algorithms we will develop in this paper differ from the existing algorithms in the literature in that they jointly optimize for the finite field of operation and the specific values for the coding parameters.

B. Vector coding over graphs

In vector coding, the source simultaneously conveys h vectors of length L to the destination, where L is a design parameter. We will denote these vectors as $\{\mathbf{u}_1, \ldots, \mathbf{u}_h\}$. These vectors take values over a predetermined field $\mathbb{F}_q$. For example, in most of this paper we will focus on binary vector coding, where $\mathbb{F}_q = \mathbb{F}_2$. The intermediate network nodes collect vectors of length L, linearly process them by multiplying them with coding matrices with values in the field $\mathbb{F}_q$, and then further propagate them. We will denote the $L \times L$ coding matrices as $\{X_k\}$. Note that to convey a binary vector of length L from an input x to an output y over the binary deterministic network, we need to use the input h times, each time conveying a single bit, and accordingly, collect h bits from the output y.

Exactly similar to before, we can associate a state variable with every edge of the network, where now each state variable is a vector of length L, and write the state-space equations for receiver j as

$$\begin{aligned} \mathbf{s}_{k+1}^B &= \mathbf{A}^B \mathbf{s}_k^B + \mathbf{B}^B \mathbf{u}_k^B \\ \mathbf{y}_k^B &= \mathbf{C}_j^B \mathbf{s}_k^B + \mathbf{D}_j^B \mathbf{u}_k^B. \end{aligned} \tag{10}$$

If the network has $m = |E|$ edges, in the above equations, $\mathbf{u}_k^B$ is the $Lh \times 1$ input vector that contains the h vectors $\{\mathbf{u}_1, \ldots, \mathbf{u}_h\}$, $\mathbf{s}_k^B$ is the $Lm \times 1$ vector that contains the m state vectors, and $\mathbf{y}_k^B$ is a $Lh \times 1$ output vector. Matrices $\mathbf{A}^B$, $\mathbf{B}^B$, $\mathbf{C}_j^B$, and $\mathbf{D}_j^B$ are now block matrices of appropriate dimension, that contain blocks of size $L \times L$. Without loss of generality, we can assume that $\mathbf{D}_j^B$ is the all zero matrix. Matrices $\mathbf{B}^B$ and $\mathbf{C}_j^B$ are fixed block matrices, that have as elements either the $L \times L$ identity matrix $\mathbf{I}$ or the $L \times L$ all zero matrix $\mathbf{0}$, at the same positions where matrices $\mathbf{B}$ and $\mathbf{C}_j$ in (1) had 1 and 0, respectively. Matrix $\mathbf{A}^B$ has also the same structure as matrix $\mathbf{A}$ in (1) with the difference that where matrix $\mathbf{A}$ had a variable entry X_k, now matrix $\mathbf{A}^B$ has at the same entry the $L \times L$ coding matrix X_k.

Exactly similar to before, the $hL \times hL$ transfer matrix for receiver j can be calculated as

$$\mathbf{M}_j^B = \mathbf{C}_j^B (\mathbf{I} - \mathbf{A}^B)^{-1} \mathbf{B}^B. \tag{11}$$

and $\mathbf{M}_j^B$ is full rank if and only if the matrix

$$\bar{\mathbf{M}}_j^B = \begin{bmatrix} \mathbf{C}_j^B & \mathbf{0} \\ \mathbf{I} - \mathbf{A}^B & \mathbf{B}^B \end{bmatrix} \tag{12}$$

is full rank. Let also

$$\mathbf{M}^B \triangleq \mathbf{M}_1^B \cdot \mathbf{M}_2^B \cdot \ \ldots \ \cdot \mathbf{M}_N^B \tag{13}$$

and

$$\bar{\mathbf{M}}^B \triangleq \bar{\mathbf{M}}_1^B \cdot \bar{\mathbf{M}}_2^B \cdot \ \ldots \ \cdots \bar{\mathbf{M}}_N^B \tag{14}$$

We observe that the dimensions of matrices $\mathbf{M}_j^B$ and $\bar{\mathbf{M}}_j^B$ depend upon the size parameter L. The multicasting code design problem is to select the size parameter L and the $L \times L$ coding matrices $\{X_k\}$ so that all matrices $\mathbf{M}_j^B$ for $j = 1 \ldots N$ are simultaneously full rank. We will denote the set of $L \times L$ matrices with elements over a field $\mathbb{F}_q$ as $M_L(\mathbb{F}_q)$. Thus, the matrices $\{X_k\}$ take values in $M_L(\mathbb{F}_q)$.

Since all the previous matrices have the same structure, we will omit the superscript B, and refer for example to a matrix $\mathbf{A}$ that has as elements variables $\{X_i\}$ that take values in an algebraic structure, either a finite field $\mathbb{F}_q$ or $M_L(\mathbb{F}_q)$.

Example III.1. *For the butterfly network [7], [8], the transfer matrices to the two receivers are*

$$\mathbf{M}_1 = \begin{bmatrix} X_1 & X_2 \\ \mathbf{I} & \mathbf{0} \end{bmatrix}, \quad \mathbf{M}_2 = \begin{bmatrix} X_1 & X_2 \\ \mathbf{0} & \mathbf{I} \end{bmatrix}.$$

In vector coding the transfer matrices are block matrices, and each element is an $L \times L$ matrix, where $\mathbf{0}$ is the all-zero and $\mathbf{I}$ the identity matrix. For the butterfly network it is sufficient to use $L = 1$.

The following theorem helps relate the problem of vector code design to the problem of a polynomial evaluation. A proof of this theorem can be found in [25].

Theorem III.2. *Let $\mathbf{M}$ be an $hL \times hL$ matrix over a field. Suppose $\mathbf{M}$ is subdivided into h^2 blocks $\mathbf{M}_{i,j}, 1 \leq i, j \leq h$ each of which is an $L \times L$ matrix. Moreover, suppose that for all numbers $1 \leq i, i', j, j' \leq n$ we have $\mathbf{M}_{i,j} \cdot \mathbf{M}_{i',j'} = \mathbf{M}_{i',j'} \cdot \mathbf{M}_{i,j}$. Then det$(\mathbf{M}) =$ det$(f(\mathbf{M}_{1,1}, \mathbf{M}_{1,2}, \ldots, \mathbf{M}_{n,n}))$ where $f(x_{1,1}, x_{1,2}, \ldots x_{n,n}) =$ det$([x_{i,j}])$.*

Thus, if the matrices we chose for the variables $\{X_i\}$ are pairwise commuting, then, from Theorem III.2, $\det(\mathbf{M}_j) = \det(f_j(X_1, \ldots, X_\nu))$, $\det(\mathbf{M}) = \det(f(X_1, \ldots, X_\nu))$ and $\det(\bar{\mathbf{M}}_j) = \det(\bar{f}_j(X_1, \ldots, X_\nu))$. We will call, in this case, the polynomial $f_j(X_1, \ldots, X_\nu) : M_L(\mathbb{F}_2) \times \ldots \times M_L(\mathbb{F}_2) \rightarrow M_L(\mathbb{F}_2)$ a *matrix* polynomial, to indicate that its evaluation results in an $L \times L$ matrix (and similarly for f and $\bar{f}$). The vector code design problem can be cast as selecting the length L and the commutative $L \times L$ matrices $\{X_k\}$ so that the matrix polynomials $f(X_1, \ldots, X_\nu)$ and $\bar{f}(X_1, \ldots, X_\nu)$ evaluate to an invertible matrix, as summarized in the following table.

Vector Algebraic Formulations:
(1) Select length L and $L \times L$ matrices $\{X_k\}$ in $M_L(\mathbb{F}_q)$ so that all matrices $\mathbf{M}_j$ become simultaneously full rank.
(2) Select length L and $L \times L$ commutative matrices $\{X_k\}$ in $M_L(\mathbb{F}_q)$ so that the the matrix polynomial $f(X_1, \ldots, X_\nu)$ of degree at most $h\Lambda N$ in each variable and of total degree at most $h\Lambda N$, evaluates to an invertible matrix.
(3) Select length L and $L \times L$ commutative matrices $\{X_k\}$ in $M_L(\mathbb{F}_q)$ so that the the matrix polynomial $\bar{f}(X_1, \ldots, X_\nu)$ of degree at most N in each variable and of total degree at most mN, evaluates to an invertible matrix.

Note that although formulations (2) and (3) are equivalent, they lead only to a subset of the possible solutions, since they require the use of commutative matrices. Formulation (1) does not impose this assumption and leads to solutions not possible with (2) and (3), as we will also see through examples in Section VI.

C. Linear Deterministic Networks

Consider a linear deterministic network over a field $\mathbb{F}_q$. If restrict our attention to linear operations inside each node of the network, we can substitute each node with the gadget shown in Fig. 4(b). In this gadget, we have associated each input and each output with a single edge, to emphasize that each input (output) receives (sends) one symbol, and then the inputs are connected to the output through a complete bipartite graph, to allow for arbitrary linear coding inside the node. Each edge of the bipartite graph is associated with a coding coefficient (or coding matrix). These are the coefficients that form the object of the network code design. Using this gadget, we can now treat our deterministic network as a directed graph, where some coding operations (corresponding to the interference) are predetermined by the network, and some coding operations (inside the network nodes) can be selected.

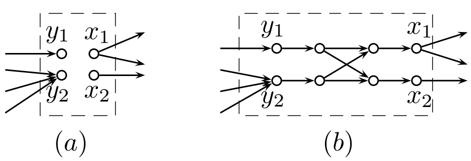

Fig. 4. (a) A node in a deterministic network and (b) equivalent representation, where the edges of the bipartite graph are each associated with a coding matrix, while the remaining edges with identity.

In the special case where the finite field of the deterministic network has size larger than N, it is sufficient for intermediate nodes to perform coding operations on a single symbol of the finite field. However, when the field size is smaller than N there does not always exist a valid coding solution when operations are restricted to a single symbol. For example, even for two receivers, there might not exist a valid solution over the binary field (see Example II.1). In this case, we will need to employ vector network coding, and use coding matrices to perform the linear combining inside the network nodes.

In both cases, we can write exactly as before the state-space equations in (1) and (10) and use the same algebraic formulations as in the previous cases. The only difference from an algebraic point of view between linear deterministic networks and graphs, is that in linear deterministic networks, the network itself can perform linear operations. In graphs, each row of matrix $\mathbf{A}$ contains either one 1 ($\mathbf{I}$ for vector coding) and zero everywhere else; or alternatively, each row contains two or more coding variables, and zero everywhere else. In deterministic networks some rows of matrix $\mathbf{A}$, corresponding to channel inputs, may have several constant terms that represent the interference. Deterministic networks simply result in more possibilities for the matrix $\mathbf{A}$ in (1), which does not affect our algorithms.

Example III.3. *For the network in Fig. 2, assume for simplicity that nodes A_1, A_2 and A_3 employ the coding matrices X_1, X_2 and X_3, while no other node employs coding. Then*

$$\mathbf{M}_1 = \begin{bmatrix} X_1 & \mathbf{0} \\ X_2 & X_2 \end{bmatrix}, \quad \mathbf{M}_2 = \begin{bmatrix} X_1 + X_2 & X_2 + X_3 \\ \mathbf{0} & X_3 \end{bmatrix}.$$

We thus require that the matrices X_1, X_2, X_3 and $X_1 + X_2$ are full rank. This is not possible for $L = 1$, as we discussed in Example II.1. For $L = 2$, we can employ

$$X_1 = \mathbf{I}, \quad X_2 = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \text{ and } X_3 = \mathbf{I}. \tag{15}$$

Consider now the algebraic formulations (2) *and* (3)*. We are seeking commutative matrices X_1, X_2 and X_3, so that the polynomial*

$$f(X_1, X_2, X_3) = X_1 X_2 X_3 (X_1 + X_2)$$

evaluates to an invertible $L \times L$ matrix. Note that the solution in (15) employs such matrices.

IV. Deterministic Code Design

In this section we develop our algebraic algorithms for vector and scalar network coding that allow multicasting over graphs and linear deterministic networks. For simplicity, we restrict our attention to binary vector network coding, where the vectors and the matrices have elements in the binary field, but the extension over arbitrary fields is straightforward, and we briefly discuss it at the end of Section IV-A.

Both for vector and scalar network coding, we start from the algebraic formulation described in Section III. That is, we construct the transfer matrices $\mathbf{M}_j$, $1 \le j \le N$, $\bar{\mathbf{M}}_j$, $1 \le j \le N$, and $\mathbf{M}$. As we are interested in polynomial time algorithms, we do not explicitly calculate the multivariate polynomials $f(X_1, \ldots, X_\nu)$ and $\bar{f}(X_1, \ldots, X_\nu)$. Our algorithms for scalar network coding seek to assign scalar values to the variables $\{X_i\}$, over a field of size q as small as possible. Similarly, for vector network coding, they seek to assign to these variables $L \times L$ binary coding matrices in $M_L(\mathbb{F}_2)$, of a size L as small as possible.

The code design consists of two basic steps:

- *Step 1:* we express each variable X_i as a polynomials of a single variable X, and we carefully select these polynomials in a manner that ensures the polynomial $f(X_1, \ldots, X_\nu)$ does not become identically zero;
- *Step 2:* for scalar network coding we select a scalar value for the variable X from a finite field of size as small as possible, and for vector network coding we select a matrix for the variable X of size as small as possible, so that the polynomials evaluate to a nonzero value for scalar coding, and to an invertible matrix for vector coding.

The second step is what distinguishes our algorithms from the algebraic code designs in the literature: our algorithms are the first, as far as we know, that jointly attempt to minimize the field size (or the size of the coding matrices) while identifying valid solutions.

In the following, we first describe the code design for vector network coding, we then proceed to the design for scalar network coding, and finally explore the connections between the two.

A. Code design for vector coding

We start by describing our algorithm, and then analyze its performance. The complexity calculation is provided by Lemma IV.5.

Step 1: Assignment of polynomials to $\{X_i\}$

1) Assume that the variables $\{X_i\}$ take scalar values. Using the matrix completion methods in [3], we can find an assignment of values to the variables $\{X_i = \alpha_i\}$, with $\{\alpha_i\}$ in a finite field $\mathbb{F}_q$ of size $q > 2^{\lceil \log N \rceil}$, so that all matrices $\bar{\mathbf{M}}_j$ become invertible, i.e., $\det(\bar{\mathbf{M}}_j) \neq 0$, $j = 1 \dots N$. Note that this assignment of values $\{X_i = \alpha_i\}$ also makes $\det(\mathbf{M}_j) \neq 0$, for $j = 1 \dots N$, and $\det(\mathbf{M}) \neq 0$. That is,
$$f(X_1 = \alpha_1, \dots, X_\nu = \alpha_\nu) \neq 0 \qquad (16)$$
2) Assume that the field $\mathbb{F}_q$, where the values $\{\alpha_i\}$ belong, has size $q = 2^k$ with $k = \lceil \log N \rceil + 1$. Using a standard representation of extension fields [24], we can express each value $\alpha_i \in \mathbb{F}_{2^k}$, identified in the previous step, as a binary polynomial $p_i(X)$ of degree at most $k-1$ in an indeterminate X. For example, we could have that α_1 is expressed as $p_1(X) = X^2 + 1$, and α_ν is expressed as $p_\nu(X) = X^3$. We substitute these polynomials in place of the variables $\{X_i\}$ in the transfer matrices $\mathbf{M}_j$ and the transfer matrix $\mathbf{M}$.
3) We calculate the determinant of the transfer matrix $\mathbf{M}$. Note that the entries of $\mathbf{M}$ are polynomials in a single variable X, and thus the determinant can be calculated efficiently, for example through interpolation as discussed in Lemma VIII.3 in the Appendix. We then get a single variable polynomial $f(X)$, that equals
$$f(X) \triangleq f(X_1 = p_1(X), \dots, X_\nu = p_\nu(X)). \qquad (17)$$
For example,
$$f(X) = f(X_1 = X^2 + 1, \dots, X_\nu = X^3).$$
We know from (16) that the polynomial $f(X)$ in (17) is not identically zero. Moreover, from Section III, it has degree at most $N(k-1)h\Lambda$ in the variable X, where Λ is the longest path length from the source to a receiver.

We underline that in the above procedure we never explicitly calculate the polynomial $f(X_1, \dots, X_\nu)$, as this calculation is not polynomial time; instead, we directly calculate the polynomial $f(X)$ in (17).

Now consider the variables $\{X_i\}$ as $L \times L$ matrices, and assume we express each such matrix as the polynomial $p_i(X)$ we have previously identified, of an $L \times L$ matrix X. This assignment ensures that the resulting matrix polynomial $f(X)$ in (17) is not identically zero. Our code design problem is now reduced to selecting the size parameter L and a single matrix $X = \mathbf{A}$ so that the matrix $f(\mathbf{A})$ is invertible.

B. Step 2: Assignment of value to X

1) Find a polynomial $g(X)$ that is co-prime with $f(X)$, of degree m as small as possible. We will prove in the analysis of our algorithm (Theorem IV.3) that we can always find such a $g(X)$ of degree $m \leq \log(Nh\Lambda \log(N))$ in polynomial time.[1]
2) If $g(X)$ has degree m, create an $m \times m$ matrix $\mathbf{A}$ so that $g(\mathbf{A}) = 0$, using for example the well known construction in Lemma IV.2.
3) Select $L = m$ and $X = \mathbf{A}$. The following Lemma IV.1 proves that for this selection, $f(\mathbf{A})$ is an invertible $m \times m$ matrix. Thus, each coding matrix X_i is assigned the $L \times L$ matrix $p_i(\mathbf{A})$.

Lemma IV.1. *Let $f(x)$, $g(x)$ be two relatively co-prime polynomials in $\mathbb{F}_q[x]$ for some field $\mathbb{F}_q$. If $\mathbf{A}$ is a matrix in $M_L(\mathbb{F}_q)$ and $g(\mathbf{A}) = 0$, then $f(\mathbf{A})$ is an inverible matrix.*

Proof: Since $\gcd(f(x), g(x)) = 1$, there exist polynomials $h_1(x), h_2(x)$ so that $f(x)h_1(x) + g(x)h_2(x) = 1$. If we set $x = \mathbf{A}$ we get $f(\mathbf{A})h_1(\mathbf{A}) + g(\mathbf{A})h_2(\mathbf{A}) = \mathbf{I}$. Since $g(\mathbf{A}) = 0$, $f(\mathbf{A})h_1(\mathbf{A}) = \mathbf{I}$. ■

Lemma IV.2. *[26] The $m \times m$ matrix*
$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & a_0 \\ 1 & 0 & 0 & \dots & 0 & a_1 \\ 0 & 1 & 0 & \dots & 0 & a_2 \\ 0 & 0 & 1 & \dots & 0 & a_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & a_{m-1} \end{bmatrix} \qquad (18)$$
has the characteristic polynomial $g(x) = a_{m-1}x^{m-1} + \dots + a_0$ and thus satisfies $g(\mathbf{A}) = 0$.

Algorithm Analysis

In this section, we analyze the performance of our algorithm. Lemma IV.5 provides the complexity calculation of our algorithm, and shows that it is polynomial time, thus providing a polynomial time algorithm for multicasting over arbitrary linear deterministic networks.

Our algorithm also attempts to minimize the size L of the employed coding matrices, which is equal to the smallest degree polynomial $g(X)$ coprime to $f(X)$ that we can find in polynomial time. In Theorem IV.3 we provide an upper bound on the degree m of $g(X)$ that we will need to employ (hard guarantees). In Lemma IV.4 we show that the fraction of polynomials that have a coprime factor of degree at most m, converges doubly exponentially (with m) to one. This strongly indicates that our algorithm will in the majority of cases result in a size much smaller than the upper bound in Theorem IV.3.

[1] In fact, we can always use the polynomial $h(X)$ that generates the field 2^k over which we made the assignment in step 1, which guarantees there exists a choice of degree $\log N$. However, we also independently prove our alternative upper bound, as it is independent of the employed technique to identify the polynomials $p_i(X)$.

Theorem IV.3. *If $f(x)$ is a nonzero binary polynomial of degree n, then there exists a coprime polynomial $g(x)$ of degree at most $\log(n+1)-1$, and we can identify it in polynomial time.*

Proof: As candidates for the polynomials $g(x)$, we are going to consider irreducible polynomials. Given that $f(x)$ has a finite degree, it cannot have as factors an arbitrary number of irreducible polynomials. In particular, if there does not exist a polynomial $g(x)$ of degree at most m (for some m) co-prime with $f(x)$, then the product of all the irreducible polynomials of degree at most m must divide the polynomial $f(x)$. That is, if we let $g_1, g_2, \dots g_K$ be all the irreducible binary polynomials of degree at most m then $\Pi_{j+1}^{K} g_j(x)$ divides $f(x)$, otherwise at least one of the g_i's is coprime with f.

In Appendix A, Lemmas VIII.1 and VIII.2, we prove that the summation of the degrees of all the irreducible binary polynomials of degree at most m is $(1-2\epsilon)2^{m+1}$ for some small ϵ. Then $f(x)$ must have degree larger than this summation, i.e., $2^{m+1} \leq n$, and the result follows.

It is also easy to see that we can find such a co-prime $g(x)$ in polynomial time. Indeed, the total number of such polynomials is at most $n+1$, and thus exhaustive search would suffice. ■

In the previous theorem, we argued that given a polynomial $f(x)$ of degree n, we can always find a coprime polynomial $g(x)$ of degree $m = O(\log n)$. We next show that, although $m = O(\log n)$ is always sufficient, our algorithm will in many cases find a co-prime polynomial of much smaller degree than $O(\log n)$. The proof of Lemma IV.4 is provided in Appendix A.

Lemma IV.4. *The fraction of polynomials that have a coprime factor of degree at most m, converges doubly exponentially (with m) to one. In particular, this fraction is at least as large as*

$$1 - \prod_{i=1}^{m} \frac{1}{2^{i\zeta(i)}},$$

where $\zeta(i)$ is the number of irreducible binary polynomials of degree i and can be approximated by $\frac{2^i}{i}$.

For example, if we take $g_1(x) = x$ and $g_2(x) = x+1$, then $\frac{3}{4}$ of all polynomials will be co-prime with either g_1 and/or g_2. For the case of $g_1(x) = x$, $g_2(x) = x+1$, $g_3(x) = x^2+x+1$, the fraction increases to $\frac{15}{16}$, and if we consider all the irreducible polynomials of degree at most 3, the fraction becomes $\frac{1023}{1024}$. That is, if we are given a polynomial any degree $n \geq 3$, a fraction of $\frac{1023}{1024}$ of these polynomials have a co-prime polynomial of degree $m \leq 3$, and thus a binary matrix of size 3×3 would lead to valid code.

Lemma IV.5. *The complexity of the algorithm is*

$$O(N(\nu+h)^3 \log(\nu+h) + \nu(\nu+h)^2 + (N \log Nh\Lambda)^3).$$

Proof: To calculate the determinant of matrix $\mathbf{M}$ and thus the polynomial $f(X_1, \dots, X_\nu)$, for each evaluation point β_i we can compute the value $D(\beta_i)$ as the determinant of the matrix $\mathbf{M}(\beta_i)$ in $O(h^3)$. So, in total we will have $O(h^3 N \log Nh\Lambda)$ operations for the determinant and we need $O((N \log Nh\Lambda)^3)$ operations to solve for $\{c_i\}$. For identifying the values $\{\alpha_i\}$, the matrix completion algorithm requires $O(N((\nu+h)^3 \log(\nu+h) + \nu(\nu+h)^2))$, where ν is the number of variables [3]. For a polynomial f of degree $n = Nh\Lambda \log N$, to find a co-prime factor of degree at most $\log n$, we need to perform approximately n polynomial divisions, where each division can be achieved using complexity $n \log n$. ■

Vector coding over a field of size $\mathbb{F}_q$, with $q > 2$.

When the ground field is $\mathbb{F}_q$ for $q > 2$ the same algorithm and analysis can be applied. The only important difference is that the number of irreducible polynomials of degree m and coefficients in $\mathbb{F}_q$ can be approximated by $\frac{1}{m}q^m$. As a result, the value L we need to employ reduces as q increases.

C. Code design for scalar coding

Step 1: Assignment of polynomials to $\{X_i\}$

Same as in Step 1 in Section IV-A, we create the not-identically zero polynomial $f(X)$. We thus reduce the code design problem to the problem of finding a value $X = \alpha$ so that $f(\alpha) \neq 0$.

Step 2: Assignment of value to X

1) Similarly to Step 2 in Section IV-A, we find an irreducible polynomial $g(X)$ that is co-prime with $f(X)$ of degree at most $m = \log n$.
2) We consider the finite field of size $\mathbb{F}_{2^m}$ generated by the polynomial $g(X)$. We make the assignment

$$X_i = p_i(X) \mod g(X).$$

Thus, each X_i is assigned a value in the field $\mathbb{F}_{2^m}$. The polynomial $f(X)$ evaluates to the nonzero value $f(X) \mod g(X)$.

That is, we assign to X the value α in the finite field generated by $g(X)$, corresponding to the indeterminate X.

Analysis

The analysis is the same as in Section IV-A. For example, for 75% of polynomials, employing a binary alphabet for scalar network coding is sufficient.

Alphabet Size in Network Coding

It is interesting to note that our algorithm reduces the problem of minimizing the alphabet size (finite field of operation) in scalar network coding, to the problem of finding a reduction of the polynomial $f(X_1, \dots, X_\nu)$ to a single variable polynomial $f(X)$ that has a co-prime factor of degree as small as possible.

It was shown in [22], [23] that the problem of finding the minimum alphabet size is NP-hard, through a reduction to the graph coloring problem. The hardness in our equivalent algebraic formulation is expressed through the manner the reduction to a single variable polynomial is performed. This reduction can be performed in multiple ways, and what is the optimal way is not clear. For example, a polynomial $f_1(X)$ can have larger degree than a polynomial $f_2(X)$, however, $f_1(X)$ may have a smaller degree coprime factor than $f_2(X)$, leading to a smaller field of operation. Our algorithm does not guarantee to find the optimal alphabet size, but still, provides a method to reduce the employed alphabet in a large fraction of cases.

D. An application in Deterministic Networks: from Multicast to Unicast

As we have discussed, a deterministic network is distinguished by the fact that the field of operation $\mathbb{F}_q$ is predetermined. Our vector code design allows to identify a vector coding solution over the desired field of operation while employing finite block processing of length L.

Note that in our multicast problem formulation for deterministic networks, we assumed that a unicast algorithm identifies in advance the min-cut values, and the inputs-outputs that we need employ towards each receiver. Here we show that this is not necessary. In fact, we observe that our code design, when restricted to the case of a single receiver, leads to a simple algorithm for determining the intermediate node operations for a single unicast connection, adding one more to the proposed approaches in the literature [12]–[16].

The main idea is to assume that all inputs and outputs in the network are employed, consider the associated coding matrices $\{X_k\}$, and calculate the transfer matrices as usual. The only challenge is in determining the min-cut value h. We procceed as follows.

- We calculate the value $\mathcal{H}$ of any cut in the network; we know for the min-cut h that $h \leq \mathcal{H}$.
- We perform $\log(\mathcal{H})$ iterations, where in each iteration, we assume that the mincut has a value H.
  - Initially, we assume that $H = \mathcal{H}$.
  - By adding virtual edges to the source and sink we create an $H \times H$ transfer matrix, in the unknowns $\{X_k\}$. We note that the method in [3] guarantees to find a full rank completion for this matrix, if this is possible, and over a field of size $q \geq 2$.
  - If it is possible, then $h = \mathcal{H}$ and the algorithm has identified a solution. If not, we examine the value $H = \frac{\mathcal{H}}{2}$.
  - We continue examining values H using any standard log-time search algorithm in the literature. For example, if there does exist a solution for $H = \frac{\mathcal{H}}{2}$, we then examine if there exists a solution for $H = \frac{3\mathcal{H}}{2}$, otherwise we examine if there exists a solution for $H = \frac{\mathcal{H}}{4}$, etc.

Once we identify the min-cut value, the algorithm procceeds in the same manner as in the case of multicasting.

V. Applications

In this section, we show how we can apply the vector network coding design framework to use specific types of structured matrices that lead to low complexity implementations. Use of structured matrices has already been proposed in the literature, for permutation matrices in [19], and very recently for rotation matrices in [20]. Our designs give a different viewpoint in the utilization of such matrices, and also allow to develop alternative designs.

The main idea that we apply in the following is that structured matrices have associated characteristic polynomials that are also structured; we can find a solution using our algorithms, provided at least one such structured polynomial is co-prime with our transfer function.

A. Rotation Matrices over $\mathbb{F}_q$

Rotation matrices over the binary field are proposed in [20] to achieve low complexity encoding at network nodes. We here consider a more general form of rotation matrices, over an arbitrary field, and show how our developed framework can be used in order to design network codes that employ such matrices.

Definition 1. *An $L \times L$ matrix $\mathbf{A}$ over a field $\mathbb{F}_q$ is called t-th order rotation matrix with respect to the L-tuple $(c_1, c_2, \ldots, c_L)$ in $\mathbb{F}_q^L$ and denoted by rot$(t; c_1, c_2, \ldots, c_L)$ if its entries are defined as following*

$$A[i,j] = \begin{cases} c_i & \text{if } j = (i+t) \mod L \\ 0 & \text{otherwise.} \end{cases}$$

Observe that the product of two rotation matrices is again a rotation matrix. More preciely we have the following lemma (the proof is straightforward and we omit it).

Lemma V.1. *rot$(t_1; c_1, c_2, \ldots, c_L) \times$ rot$(t_2; b_1, b_2, \ldots, b_L) =$ rot$(t_1+t_2; c_1 b_{j_1}, c_2 b_{j_2}, \ldots, c_L b_{j_L})$, where j_i is such that $(j_i + t_2) \mod L = i$.*

As a result, taking the powers of a rotational matrix leads to a rotational matrix as well.

Recall that in our algorithms, we substitute the coding coefficients $\{X_i\}$ to be in general polynomials of an indeterminate X. From the previous lemma, if we would like the matrices $\{X_i\}$ to be rotation matrices, we can simply select the variable X to be a rotation matrix, and then have the variables $\{X_i\}$ be powers of X; namely,

$$X_i = X^{m_i} \tag{19}$$

for some exponent m_i. We then need to select the exponents $\{m_i\}$ so that the polynomial $f(X_1, \ldots, X_\nu)$ does not become identically zero. We can easily do this by slightly modifying Step 1 in our algorithms.

Alternative substitution for Step 1: As before we employ the algorithm in [3] to find an assignment $X_i = \alpha_i$ with $f(\alpha_1, \alpha_1, \ldots, \alpha_\nu)$ invertible. Next, let α be a primitive element of the finite field $\mathbb{F}_q$, we can then express each α_i in $\mathbb{F}_q$ as $\alpha_i = \alpha^{m_i}$ for some m_i. Therefore we have:

$$0 \neq f(\alpha_1, \alpha_2, \ldots, \alpha_\nu) = f(\alpha^{m_1}, \alpha^{m_2}, \ldots, \alpha^{m_\nu}) = f(\alpha).$$

Selecting these values for the exponents in (19) concludes the first step.

Another statement about the rotation matrices is the following lemma, which is easy to prove.

Lemma V.2. *The characteristic polynomial of the matrix rot$(1; c_1, c_2, \ldots, c_L)$ is $X^L - \prod_{i=1}^{L} c_i$.*

Rotational coding may not always lead to a solution; the following theorem helps explore when it is possible.

Theorem V.3. *Consider a polynomial $f(X)$ which is not identically zero, resulting from the transfer polynomial of a network $f(X_1, \ldots, X_\nu)$ by substituting $X_i = X^{m_i}$. If $f(X)$ is coprime with any of the polynomials $g(X) = X^L - c$ for some $c \in \mathbb{F}_q$, then we can solve the network coding problem in polynomial time using rotational coding.*

Proof: If such a coprime factor exists, use $X = \text{rot}(t;\ 1 \ldots 1\ c)$. ■

Using this theorem, the following corollary characterizes when it is possible to have a solution with rotational matrices.

Corollary 1. *If there exists an assignment for the variables of the polynomial f with powers of a single variable X so that $f(X)$ is co-prime with a polynomial $g(X) = X^k - c$ for some $c \in \mathbb{F}_q$, then the multicast network code design is solvable with rotational coding.*

B. Permutation Matrices over $\mathbb{F}_q$

Another class of matrices that can be employed is the class of all matrices that are associated with the powers of a fixed permutation matrix $\mathbf{A}$. Randomized designs for binary permutations were presented in [19]. Our approach allows us to develop finite length deterministic designs.

Each permutation can be written as the product of cycles; thus the matrix associated with every permutation can be written as a block diagonal matrix so that the block diagonals are rotation matrices.

Definition 2. *A permutation matrix is a block diagonal matrix where each block is a rotation matrix.*

This implies that the same analysis and approach as in the case of rotation matrices can be applied .That is, we can substitute the coding variables as powers of a single variable, and check if the resulting polynomial is coprime with the characteristic polynomial of a permutation matrix.

C. Fourier Transform Matrices

Another special type of the matrices we can use is discrete Fourier transform matrices. Such matrices again have low implementation complexity.

Definition 3. *A discrete Fourier transform matrix $\mathbf{A}$ over a finite field $\mathbb{F}_q$ has elements*

$$\mathbf{A}[i, j] = \alpha^{(i-1)\cdot(j-1)}$$

where α is a primitive element in $\mathbb{F}_q$.

Next, it is well known that the transform Fourier matrices satisfy the equation $\mathbf{A}^4 - \mathbf{I} = 0$. We thus have the following result.

Lemma V.4. *If there exists an assignment of the uni-variable polynomials to the variables of the transfer matrix of a network in such a way that the resulting polynomial is co-prime with the polynomial $X^4 - 1$, then there exists a network coding for the network so that the coding matrices are polynomials of the matrix $\mathbf{A}$.*

D. Circulant matrices

Circulant matrices, that we define in the following, have also a well studied structure, and several algorithms have been developed for example for efficiently calculating the powers of circulant matrices [29], [30]. In Lemma IV.2 we have shown that given a polynomial $g(x)$ of degree L we can always construct an $L \times L$ matrix so that $g(\mathbf{A}) = 0$. We here prove that in fact, without loss in terms of the size L, we can always construct a matrix $\mathbf{A}$ that satisfies $g(\mathbf{A}) = 0$ and additional is circulant. We can thus leverage properties of circulant matrices for our algorithms.

Definition 4. *An $L \times L$ matrix over a field $\mathbb{F}_q$ is called a circulant matrix if there exists an L-tuple $(c_0, c_1, \ldots, c_{L-1}) \in \mathbb{F}_q^L$ so that the (i, j)-th element of $\mathbf{A}$ is equal to c_{j-i}, with indices taken modulo L. In this case, the matrix $\mathbf{A}$ is called a circulant matrix generated by $(c_0, c_1, \ldots, c_{L-1})$ and is denoted by circ$(c_0, c_1, \ldots, c_{L-1})$.*

For a proof of the following lemma see, for example, [29].

Lemma V.5. *The eigenvalues of circ$(c_0, c_1, \ldots, c_{L-1})$ are of the form $\sum_{j=0}^{L-1} c_j \cdot \alpha^j$, where α is any L-th root of unity in some extension field of $\mathbb{F}_q$.*

Using this lemma we are able to prove the following theorem.

Theorem V.6. *For every polynomial $f(x) \in \mathbb{F}_q[x]$ with deg$(f) = L$, there exists a circulant matrix whose characteristic polynomial is equal to $f(x)$.*

Proof: Let $r_1, r_2, \ldots, r_L$ be the roots of $f(x)$ with possibly repetation. Consider the following system of linear equations.

$$\sum_{j=0}^{L-1} c_j.\alpha^j = r_1$$
$$\sum_{j=0}^{L-1} c_j.\alpha^{2j} = r_2$$
$$\ldots$$
$$\sum_{j=0}^{L-1} c_j.\alpha^{Lj} = r_L$$

where α is a primitive L-th root of unity and c_j's are unkowns. Notice that the coefficient matrix of this set of linear eauations is a Vandermond matrix and therefore there exists a unique solution to this equation. Now, using the previous Lemma it is clear that circ$(c_0, c_1, \ldots, c_{L-1})$ has eigenvalues $r_1, r_2, \ldots, r_L$. Therefore, the characteristic polynomial of circ$(c_0, c_1, \ldots, c_{L-1})$ is equal to $f(x)$. ■

VI. Scalar vs. vector operation: a comparison

We start this section by defining an equivalence between scalar and vector network coding solutions, and a set of solutions that are according to our definition equivalent; we then procceed to identify the benefits vector code designs can offer as compared to scalar designs, and summarize these benefits at the end of the section.

Definition 5. *In a multicasting network, we say that a vector coding solution using binary vectors of length L, and a scalar solution that employs a finite field of size $\mathbb{F}_{2^m}$ are equivalent, iff $m = \log L$.*

It is clear that, if our proposed algorithm for vector network coding identifies a solution of size $L = m$, then the algorithm for scalar network coding will identify a solution over a finite field of size $\mathbf{F}_{2^m}$. Thus these algorithms lead to equivalent solutions. The next two theorems make this equivalence more general, and up to some degree independent of the method employed to identify the vector or scalar solution.

The first Theorem VI.1 implies that, if we can solve the scalar network coding problem for a network over a field $\mathbb{F}_{2^m}$, with any algorithm that we desire, then we are able to solve the vector network coding problem over the same network using binary matrices of dimension $m \times m$. Therefore, it guarantees that binary matrices are at least as useful as finite fields, since, every solution over a finite field can be translated to an equivalent solution using matrices. This is a well known result in algebraic coding, but we repeat it here for completeness.

Theorem VI.1. *For a non-zero polynomial $f(x_1, x_2, \ldots, x_n)$, assume that there are values $\alpha_1, \alpha_2, \ldots, \alpha_n \in \mathbb{F}_{2^m}$ with $f(\alpha_1, \alpha_2, \ldots, \alpha_n) \neq 0$. Then there are pairwise commuting matrices $A_1, A_2, \ldots, A_n \in M_m(\mathbb{F}_2)$ such that $f(A_1, A_2, \ldots, A_n)$ is an invertible matrix.*

Proof: Consider the natural one-to-one ring homomorphism $\phi : \mathbb{F}_{2^{\alpha(f)}} \to GL_{\alpha(f)}(\mathbb{F}_2)$ as defined in [24]. Let $\alpha_1, \alpha_1, \ldots, \alpha_n \in \mathbb{F}_{2^m}$ with $g(\alpha_1, \alpha_1, \ldots, \alpha_n) \neq 0$. Let $A_i := \phi(\alpha_i)$ for $1 \leq i \leq n$. Since ϕ is a ring homomorphism we have: $f(A_1, A_2, \ldots, A_n) = f(\phi(\alpha_1), \phi(\alpha_1), \ldots, \phi(\alpha_n)) = \phi(f(\alpha_1, \alpha_1, \ldots, \alpha_n)) = \phi(\alpha)$. Since $\alpha \neq 0$, $\phi(\alpha)$ is invertible and hence, $f(A_1, A_2, \ldots, A_n)$ is invertible. ∎

For example, we can always find a vector coding solution of size $m = \log N$, by using any of the polynomial time network code design algorithms in the literature, and translate this solution to vectors. This would lead to always operating using the worst case size $\log N$.

The second Theorem VI.2, shows that, for the vector algebraic formulations (2) and (3), if for vector coding we find a matrix $\mathbf{A}$ of size $m \times m$ such that $f(\mathbf{A})$ is invertible, and thus can solve the vector coding problem using size m, no matter how we identified this matrix $\mathbf{A}$, we can translate this to scalar solution over a field of size $\mathbb{F}_{2^m}$.

Theorem VI.2. *Consider a polynomial $f(x)$ with binary coefficients, and assume that there exists an $m \times m$ matrix $\mathbf{A}$ so that $f(\mathbf{A})$ is invertible. Then, there exists a scalar value α in a finite field of size at most 2^m so that $f(\alpha) \neq 0$.*

Proof: If $f(\mathbf{A})$ is invertible for some $\mathbf{A} \in M_m(\mathbb{F}_2)$, then any eigenvalue a of $\mathbf{A}$ also satisfies $f(a) \neq 0$. On the other hand, since the characteristic polynomial of $\mathbf{A}$ has degree m, the degree of the characteristic polynomial $h(x)$ of a over $\mathbb{F}_2$ is at most m [24]. If we take the field generated by α, it is of size at most 2^m and it contains α. ∎

We underline that Theorem VI.2, shows that we can translate a vector solution to a scalar solution, under two basic assumptions:

1) The variables $\{X_k\}$ are pairwise commuting, in order to be able to write the matrix polynomial, and
2) The multivariate polynomial is reduced to a single variable polynomial.

However, if we do not impose these assumptions, searching over the set of matrices offers a much larger set of choices than restricting the search to finite fields. The following three examples VI.3, VI.4 and VI.5 illustrate this point. Additionally, as we illustrated in section V, vector coding allows to leverage well-studied structure and properties of matrices to create designs that optimize criteria such as operational complexity.

Our first example assumes that we are restricted to operate using vectors and matrices of size exactly equal to $L = m = 5$. For example, for practical reasons we have a fixed packet length. It is possible that we can achieve this using vector network coding, while there does not exist a scalar solution in a field of size $\mathbb{F}_{32}$, i.e., we cannot translate a scalar solution to a vector solution of the desired length.

Example VI.3. *We here present an example of a binary polynomial $f(x)$ that has as roots all elements of the field $\mathbb{F}_{32}$, while there exists a matrix A in $M_5(\mathbb{F}_2)$ so that $g(A)$ is invertible. Let $f(x) = x^{32} - x$. Clearly $f(\alpha) = 0$ for every $\alpha \in \mathbb{F}_{32}$. It can be shown that $f(x)$ has 8 irreducible factors over the binary field, namely x, $x - 1$ and 6 more factors each of which is of degree 5. Thus the polynomial $g(x) = (x^2 + x + 1)(x^3 + x + 1)$ is a polynomial of degree 5 and is coprime with $f(x)$. Now, take any binary matrix with characteristic polynomial $g(x)$ and use Lemma IV.1 to construct the vector coding solution.*

In the previous example we leverage the fact that if there exist vector solutions of length k_1 and k_2, there exists a vector solution of length $k_1 + k_2$. In contrast, a solution over fields of size 2^{k_1} and 2^{k_2} does not necessarily imply a solution over a field of size $2^{k_1+k_2}$. This is because a field of size $\mathbb{F}_{2^k}$ is a subfield of $\mathbb{F}_{2^l}$ only if k divides l.

The next example demonstrates that it is possible to have a valid vector code that employs length $L = 10$, while there does not exist a valid scalar code in *any* field of size smaller or equal to 2^{10}.

Example VI.4. *Define $f_i(x)$ to be the product of all the binary irreducible polynomials of degree i. Let*

$$h(x, y) = F(x, y)G(x, y)$$

with

$$F(x,y) = f_4(x)\cdot f_6(x)\cdot f_7(x)\cdot f_8(x)\cdot f_9(x)\cdot f_{10}(x)$$

and

$$G(x,y) = ((x^4+x)(x^8+x)+(y^4+y)(y^8+y))(x^{32}+x+y^{32}+y).$$

We first show that the evaluation of h at any element of the fields $\mathbb{F}_2, \mathbb{F}_4, \ldots, \mathbb{F}_{1024}$ is zero. If $x, y \in \mathbb{F}_4$ then both x^4+x and y^4+y are zero and therefore $h(x,y)=0$. Similar argument shows that if $x, y \in \mathbb{F}_8$ or $x, y \in \mathbb{F}_{32}$ then $h(x,y)=0$. If $x, y \in \mathbb{F}_{16}$ then either both of x or y will be in $\mathbb{F}_4$ or at least one of them is not in that field. If both of x and y are in $\mathbb{F}_4$ then both x^4+x and y^4+y are zero and therefore $h(x,y)=0$. If one of them, say x, is in $\mathbb{F}_{16}$ but not in $\mathbb{F}_4$ then the characteristic polynomial of x will be an irreducible polynomial of degree 4 and therefore $f_4(x)=0$. Thus, in both cases, $h(x,y)=0$. Similar argument shows that if $x, y \in \mathbb{F}_{2^i}$ for $i=1,2,\ldots,10$ then $h(x,y)=0$.

Next, we will show that $h(x=X_1, y=X_2)$ is an invertible matrix, where

$$X_1 = \begin{bmatrix} A_1 & 0 & 0 \\ 0 & A_2 & 0 \\ 0 & 0 & A_3 \end{bmatrix}, \text{ and } X_2 = \begin{bmatrix} A_3 & 0 & 0 \\ 0 & A_1 & 0 \\ 0 & 0 & A_2 \end{bmatrix}$$

where

$$A_1 = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

and

$$A_3 = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

First notice that the characteristic polynomials of both matrices X_1, X_2 is the product of the characteristic polynomials of the matrices A_1, A_2, and A_3 which is $c(x) = (x^2+x+1)(x^3+x+1)(x^5+x^2+1)$. All the three factors of $c(x)$ are irreducible polynomials. Since $c(X_1)=c(X_2)=0$ and $c(x)$ is co-prime with the polynomials $f_4, f_6, f_7, f_8, f_9, f_{10}$, then $F(x,y)$ will be invertible if evaluated at $x=X_1, y=X_2$. So, we only need to check the invertiblity of the factors $((x^4+x)(x^8+x)+(y^4+y)(y^8+y))$ and $(x^{32}+x+y^{32}+y)$ of $G(x,y)$. We can evaluate each of these factors to verify that all of them become invertible matrices. Therefore $h(X_1, X_2)$ is a 10×10 invertible binary matrix.

The next simple example makes the point that, the set of vector solutions can be larger than the set of scalar solutions.

Example VI.5. *Consider again the network in Fig. 2, that was examined in Example III.3. The solution provided in (15) employs commutative matrices. However, we can also employ non-commutative matrices such that X_1, X_2, X_3 and X_1+X_2 are full rank, for example*

$$X_1 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}, \quad X_2 = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \text{ and } X_3 = \mathbf{I}. \tag{20}$$

We will count the number of assignments for the variables X_1, X_2, X_3 using 2×2 binary matrices so that the matrix $X_1X_2X_3(X_1+X_2)$ is invertible. In fact it suffices to choose the value of X_3 to be an arbitrary invertible matrix and values of X_1, X_2 so that X_1, X_2, X_1+X_2 are all invertible. Clearly there are 6 choices for X_3. The only way we can find two invertible binary matrix whose sum is also invertible is to choose both matrices from any of the following two sets.

$$\{\mathbf{I}, \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}\}, \{\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}\} \tag{21}$$

This shows that there are 12 possibilities to chose the ordered pair X_1, X_2 satisfying the properties mentioned above. So, in total there are 72 different assignments for X_i's which make $X_1X_2X_3(X_1+X_2)$ an invertible matrix. Now if we want to make sure that all the matrices X_1, X_2, X_3 are commuting we can still choose X_1, X_2 the same as before but then having X_1, X_2 there are only 3 possibilities for X_3. Therefore there will be 36 different assignments. We can thus double the solution space if we do not restrict our attention to commutative matrices.

Benefits of vector coding

To summarize, we have shown that designing vector coding solutions, as compared to scalar solutions, has the following benefits:

- All scalar solutions over a field of size $q=2^k$ can be translated to a vector solution of length k (see Theorem VI.1).
- There may exist vector solutions of length k, while there exists no scalar solution in a field of size $q \le 2^k$ (see example VI.4). In general, if we are not restricted to use commutative matrices that are polynomials of the same matrix, there exists a larger set of solutions with vector as compared to scalar coding (see example VI.5).
- If there exist vector solutions of length k_1 and k_2, there exists a vector solution of length k_1+k_2. In contrast, a solution over fields of size 2^{k_1} and 2^{k_2} does not necessarily imply a solution over a field of size $2^{k_1+k_2}$ (see example VI.3).
- Vector coding allows to leverage well-studied structure and properties of matrices (see section V).

VII. Related Work

Algebraic network coding was introduced in the seminal paper [1]. We here extend this framework to admit vector coding and linear deterministic networks as special cases. This develops on the ideas we have presented in [17], [18]. Indepedently from our work, the authors in [21] also extended the algebraic model to linear deterministic networks, and the authors in [20] considered designs for deterministic networks that use a specific type of structured matrices.

Polynomial time designs for scalar network coding over graphs were proposed for example in [1]–[4]. The novelty in our scalar code design algorithm is that it jointly optimizes the effective field size as well as the coding coefficients, and can thus result in a much smaler field while still being a polynomial time algorithm.

The potential of vector coding was already indicated through simple examples in several papers, see for example [5]. Designs specifically targeted to vector coding were first proposed in [19], where vector coding was termed block network coding, and where the coding matrices were restricted to be binary interleaving matrices, leading to permute-and-add network codes. The approach in [19] examines this specific set of structured matrices is probabilistic, and does not provide deterministic designs for finite length. Very recently, the authors in [20] have proposed the use of rotation matrices, and also proposed designs specifically targetted to this set of matrices. Our work generalizes these results and provides a unifying framework which leads to additional structured network code designs.

VIII. Conclusions

In this paper we developed new algebraic algorithms for the problem of vector and scalar network code design. We start our work by extending the algebraic framework developed for multicasting over graphs in [1] to include operations over matrices and to admit both graphs and linear deterministic networks as special cases. The main idea in our approach is to reduce the problem of code design to an algebraic problem of finding coprime factors of a given polynomal. Based on this, we provided algorithms for scalar coding that attempt to minimize the alphabet size and show a doubly exponential convergence to a solution. We also provided algorithms for vector coding that allow to use finite lengths and systematically design vector coding solutions. We illustrated how our approach can effectively utilize structured matrices to further reduce the encoding complexity. We also explored a notion of equivalence between vector and scalar network coding and identified potential benefits of vector network coding. Our algorithms lead as a special case to algorithms for multicast and unicast communication over arbitrary linear deterministic networks. We believe that the approach we have developed may be useful for network optimization problems that allow coding, and for deriving additional algorithms for network code design.

References

[1] R. Koetter and M. Médard, "Beyond routing: an algebraic approach to network coding", *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 782-796, October 2003.

[2] S. Jaggi, P. Sanders, P. Chou, M. Effros, S. Egner, K. Jain and L. Tolhuizen, "Polynomial time algorithms for multicast network code construction", *IEEE Transactions on Information Theory,* vol. 51, no. 6, pp. 1973–1982, 2005.

[3] N. Harvey, "Deterministic network coding by matrix completion", MS Thesis 2005.

[4] T. Ho, R. Koetter, M. Médard, M. Effros, J. Shi, and D. Karger, "A random linear network coding approach to multicast," *IEEE Transactions on Information Theory*, vol. 52, iss. 10, pp. 4413-4430, October 2006.

[5] C. Fragouli and E. Soljanin, "A connection between network coding and convolutional codes," *IEEE International Conference on Communications (ICC)*, pp. 661–666, vol.2, June 2004.

[6] C. Fragouli and E. Soljanin, "Decentralized network coding", *Information Theory Workshop*, 2004.

[7] R. Ahlswede, N. Cai, S-Y. R. Li, and R. W. Yeung, "Network information flow", *IEEE Trans. Inform. Theory,* vol. 46, pp. 1204–1216, July 2000.

[8] S-Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Trans. Inform. Theory,* vol. 49, pp. 371–381, Feb. 2003.

[9] S. Avestimehr, S N. Diggavi and D N C. Tse, "Wireless network information flow", *Proceedings of Allerton Conference on Communication, Control, and Computing*, Illinois, September 2007. See http://licos.epfl.ch/index.php?p=research_projWNC

[10] S. Avestimehr, S N. Diggavi and D N C. Tse, "A deterministic approach to wireless relay networks", *Proceedings of Allerton Conference on Communication, Control, and Computing*, Illinois, September 2007.

[11] S. Avestimehr, S N. Diggavi and D N C. Tse, "Wireless netwokr information flow: a deterministic approach", $arXiv:0906.5394$, June 2009.

[12] A. Amaudruz and C. Fragouli, "Combinatorial algorithms for wireless information flow", *SODA*, January 2009.

[13] J. Ebrahimi and C. Fragouli, "Combinatorial algorithms for wireless information flow", arXiv:0909.4808, under submission, Sept. 2009.

[14] M. Goemans, I. Satoru and R. Zenklusen, "An algorithmic framework for wireless information flow", *Allerton*, September 2009.

[15] S. Tabatabaei Yazdi and S. Savari, "A combinatorial study of linear deterministic relay networks", *SODA*, January 2010.

[16] C. Shi and A. Ramamoorthy, "Fast algorithms for finding unicast capacity of linear deterministic wireless relay networks", arXiv:0909.5507v1, under submission, 2010.

[17] J. Ebrahimi and C. Fragouli, "Multicasting algorithms for deterministic networks", *IEEE Information Theory Workshop (ITW)*, Cairo, January 2010.

[18] J. Ebrahimi and C. Fragouli, "Vector network coding", *EPFL Technical Report*, January 2010, under submission.

[19] S. Jaggi, Y. Cassuto, M. Effros, "Low complexity encoding for network codes," *ISIT* 2006.

[20] M. Khojastepour, A. Keshavarz-Haddad, "Rotational coding achieves multicast capacity of deterministic wireless networks", *IEEE Allerton*, September 2009.

[21] MinJi Kim, Muriel Medard "Algebraic network coding approach to deterministic wireless relay networks", $http://arxiv.org/pdf/1001.4431$.

[22] C. Fragouli and E. Soljanin, "Information flow decomposition for network coding," *IEEE Transactions on Information Theory*, vol. 52, iss. 3, pp. 829–848, March 2006.

[23] A. Lehman and E. Lehman, "Complexity classification of network information flow problems", *ACM SODA*, 2004.

[24] P. Morandi, "Field and Galois Theory", *Springer*, 1996.

[25] Istvan Kovacs, Daniel S. Silver and Susan G. Williams, "Determinants of commuting-block matrices", *The American Mathematical Monthly*, vol. 106, no. 10, pp. 950-952, December 1999.

[26] Horn and Johnson, "Linear Algebra", *Cambdrige University Press*.

[27] I. Chaparlinski, "On finding primitive roots in finite fields", *Theoretical computer science*, vol. 157, no 2, pp. 273-275, May 1996.

[28] J. T. Schwartz, "Fast probabilistic algorithms for verification of polynomial identities", *Journal of the ACM*, vol. 27, no. 4, pp. 701-717, 1980.

[29] P.J. Davis, "Circulant Matrices", *Chelsea Publishing*, New York 1994.

[30] J. Rimas, "On computing of arbitrary positive integer powers for one type of even order symmetric circulant matrices", *Applied Mathematics and Computation*, vol. 172, pp. 8690, 2006.

[31] C. Fragouli and E. Soljanin, "Network coding: Fundamentals," *Foundations and Trends in Networking*, vol. 2, pp. 1–133, 2007.

[32] Cover and Thomas, "Elements of Information Theory", *Wiley*, 2006.

Appendix A

Lemmas used in the proof of Theorem IV.3

For the proof in Theorem IV.3, we need to find a good approximation for the summation of the degrees of all irreducible polynomials of degree at most m. To do so, we use the following well-known result on the number of irreducible binary polynomials of certain degree.

Lemma VIII.1. *The number of irreducible polynomials of degree m is equal to $\frac{1}{m}\sum_{d|m}\mu(\frac{m}{d}2^d)$ where $\mu(d)$ is the mubius function (see [?]). Moreover, for every $\epsilon > 0$, there exists a number N_ϵ so that if $m > N_\epsilon$ then the number of irreducible polynomials of degree m is at least $\frac{(1-\epsilon)}{m}2^m$.*

Lemma VIII.2. *Let ϵ be a positive real number. If $m > 2N_\epsilon$ then the summation of the degrees of all the irreducible binary polynomials of degree at most m is at least $(1-2\epsilon)2^{m+1}$.*

Proof: For $k = N_\epsilon, N_\epsilon + 1, \ldots, m$, the number of irreducible polynomials of degree k is at least $\frac{(1-\epsilon)}{k}2^k$ by the previous lemma. Each such polynomial has degree k, and therefore, the summation of all the irreducible polynomials of degree k, equals k times their number, which is at least $(1-\epsilon)2^k$. Summing up this quantity for $k = N_\epsilon, N_\epsilon+1, \ldots, m$, we see that the summation of the degrees of all the irreducible polynomials of degree at most m is at least $(1-2\epsilon)2^{m+1}$. ■

We observe that the numbers N_ϵ are not large. In fact the upper bound $\frac{1}{m}2^m$ for the number of irreducible polynomials of degree m is very close to the actual number. We will use in this paper this approximation for the number of irreducible polynomials, as it does not affect the order arguments. We will thus dispense of ϵ and N_ϵ and say that the summation of all the irreducible polynomials of degree at most m is approximately 2^{m+1}.

Proof of Lemma IV.4

Proof: For a fixed polynomial $g(x)$ not identically zero $g(x)$ is a factor of $f(x)$ for a fraction of $\frac{1}{2^m}$ of all polynomials $f(x)$ of degree n. This follows by observing that the remainder after dividing $f(x)$ with $g(x)$, can be any of the 2^m binary polynomials of degree smaller or equal to $m-1$. Moreover, we can divide the polynomials of degree n to 2^m mutually-exclusively and equally-sized sets, one corresponding to each possible remainder.

Let $g_1(x), g_2(x), \ldots g_k(x)$ be pairwise co-prime polynomials. Therefore $f(x)$ is divisible by all of them if and only if it is divisible by their product. Therefore, the fraction of non-zero polynomials f that have none of the g_i's as a factor is at least $1 - \prod_{i=1}^{k}\frac{1}{2^{m_i}}$, where m_i is the degree of $g_i(x)$. ■

Efficient time calculation of a polynomial determinant

Lemma VIII.3. *Consider a $k \times k$ matrix whose elements are either binary variables or polynomials in the same variable x of degree at most p. Then we can calculate the determinant by solving a $(pk+1)\times(pk+1)$ set of linear equations.*

Proof: Let $g(x)$ denote the polynomial resulting from the determinant, we know that the degree ν of the polynomial $g(x)$ is at most $\nu \le pk$. Since we know that

$$g(x) = c_\nu x^\nu + \ldots + c_1 x + c_0$$

for some unknown coefficients $\{c_i\}$, we can simply evaluate the value of the polynomial at $\nu+1$ points $\{\beta_i\}$ and solve a system of linear equations to retrieve $\{c_i\}$. We select β_i from a finite field $\mathbf{F}_{q'}$ of size q' much larger than ν we take the $\nu+1$ distinct nonzero elements β_j of $\mathbf{F}_{q'}$, and for each one, compute det($\mathbf{M}(\beta_j)$). Notice that in the system of linear equations, the corresponding matrix is a Vandermonde matrix and in particular, it is invertible. So, there exists a unique solution for this system of linear equation.

Note that if we expand the determinant as in the definition of the determinant of a matrix, since all the entries are binary polynomials, the determinant will be a binary polynomial as well. Thus we know that $\{c_i\}$ are binary coefficients, and, although we solve linear equations over the finite field where the β_i's belong, the unique solution of this system of equations will be binary. ■