Vector Network Coding Algorithms

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. Our algorithms for scalar network jointly optimize the employed field size while selecting the coding coefficients. Similarly, for vector coding, our algorithms optimize the length L while designing the coding matrices. These algorithms apply both for regular network graphs as well as linear deterministic networks.**

I. Introduction

In this paper we consider the problem of network code design for multicasting common information at rate h to N receivers using vector communication. 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 the same role as scalar coding coefficients in traditional network coding [1], [6], [7]. 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. For example, the authors in [13] propose probabilistic designs that employ permutation matrices for the coding matrices. Our work provides a unifying framework for deterministic designs. Additionally, we have shown in [11] that vector coding can be directly deployed in linear deterministic networks that have proposed as approximate characterizations for wireless networks [8]–[10].

In [11], we have extended the algebraic framework developed for multicasting over graphs in [1] in two ways: (i) to include operations over matrices and (ii) to accept both graphs and linear deterministic networks as special cases. Independently from our work, the framework in [1] was also extended over deterministic networks in [14].

In this paper, we build on this algebraic framework to develop new algorithms for vector and scalar coding, that can be employed both over graphs and deterministic networks. Our contributions in this paper include:
• We provide a polynomial time algorithm for the design of the $L \times L$ coding matrices used in vector network coding when multicasting to N receivers. Our algorithm reduces the problem of finding a small size L to the problem of finding a small degree co-prime factor of an algebraic polynomial, and leads to solutions not possible with using scalar network coding, as illustrated through examples in Section IV.
• 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, we will be able to find in polynomial time binary coding matrices of size at most 3×3 that lead to a valid code.
• Our approach provides 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] 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 employed field size as well as the coefficients, can result in a much smaller 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 co-prime factor of algebraic polynomials.

The paper is organized as follows. Section II provides our notation and the algebraic framework generalization; Section III develops our code design algorithms; Section IV compares scalar and vector network coding; and Section V concludes the paper.

II. Algebraic Framework

We here review the algebraic framework in [1], [11].

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 as in the case of scalar coding [1], 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

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

If the network has $m = |E|$ edges, in the above equations, $\mathbf{u}_k$ is the $Lh \times 1$ input vector that contains the h vectors $\{\mathbf{u}_1, \ldots, \mathbf{u}_h\}$, $\mathbf{s}_k$ is the $Lm \times 1$ vector that contains the m state vectors, and $\mathbf{y}_k$ is a $Lh \times 1$ output vector. Matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}_j$, and $\mathbf{D}_j$ are block matrices of appropriate dimension, that contain blocks of size $L \times L$. Without loss of generality, we can assume that $\mathbf{D}_j$ is the all zero matrix. Matrices $\mathbf{B}$ and $\mathbf{C}_j$ 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 $\mathbf{0}$. Matrix $\mathbf{A}$ is common for all receivers and reflects the network topology, that is, the way the edges (memory elements) are connected. The entries of this matrix are either constant, or the unknown coding matrices $\{X_k\}$, and we assume we have ν such unknowns.

The $hL \times hL$ transfer matrix for receiver j can be calculated as

$$\mathbf{M}_j = \mathbf{C}_j(\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}. \tag{2}$$

Also, let

$$\mathbf{M} \triangleq \mathbf{M}_1 \cdot \mathbf{M}_2 \cdot \ \ldots \ \cdot \mathbf{M}_N \tag{3}$$

We observe that the dimensions of matrices $\mathbf{M}_j$ 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 $\mathbf{M}_j$ 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 $M_L(\mathbb{F}_q)$.

The algebraic formulation for vector network coding is exactly the same as that for scalar network coding up to this point; the only difference is that for scalar network coding $\{X_k\}$ take values in $\mathbb{F}_q$, while for vector network coding in $M_L(\mathbb{F}_q)$.

Scalar Formulations: For scalar network coding, let

$$f(X_1, \ldots, X_\nu) = \det(\mathbf{M}) \tag{4}$$

be the determinant of matrix $\mathbf{M}$. The following two 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 $\mathbb{F}_q$ 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 $\mathbb{F}_q$ so that the polynomial $f(X_1, \ldots, X_\nu)$ evaluates to a nonzero value.

From the sparse zero lemma [4], [19], 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]. The algorithms for scalar coding we will develop in this paper differ in that they jointly optimize for the finite field of operation and the specific values for the coding parameters.

Vector Formulations: The following theorem helps relate the problem of vector code design to the problem of a polynomial evaluation (for a proof see [17]).

Theorem II.1. *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 II.1, $\det(\mathbf{M}_j) = \det(f_j(X_1, \ldots, X_\nu))$, $\det(\mathbf{M}) = \det(f(X_1, \ldots, X_\nu))$. We will call, in this case, the polynomial $f_j(X_1, \ldots, X_\nu) : M_L(\mathbb{F}_2) \times \ldots \times M_L(\mathbb{F}_2) \to M_L(\mathbb{F}_2)$ a *matrix* polynomial, to indicate that its evaluation results in an $L \times L$ matrix. 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 polynomial $f(X_1, \ldots, X_\nu)$ evaluates 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 $f(X_1, \ldots, X_\nu)$ evaluates to an invertible matrix.

Note that formulation (2) leads only to a subset of the possible solutions, since it requires the use of commutative matrices. Formulation (1) does not impose this assumption and leads to solutions not possible with (2), as we will also see through examples in Section IV.

III. Code Design Algorithm

In this section we develop our algorithms. Both for vector and scalar network coding, we start from the algebraic formulation described in Section II. That is, we construct the transfer matrices $\mathbf{M}_j$, $1 \leq j \leq N$, and $\mathbf{M}$. As we are interested in polynomial time algorithms, we do not explicitly calculate the multivariate polynomials $f(X_1, \ldots, X_\nu)$. 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 q as small as possible, and for vector network coding we select a $L \times L$ matrix in $M_L(\mathbb{F}_2)$ for the variable X of size L 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 while identifying valid solutions.

A. Code design for vector coding

We start by describing our algorithm, and then analyze its performance.

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 $\mathbf{M}_j$ become invertible, i.e., det$(\mathbf{M}_j) \neq 0$, for $j = 1 \ldots N$, and det$(\mathbf{M}) \neq 0$. That is,

$$f(X_1 = \alpha_1, \ldots, X_\nu = \alpha_\nu) \neq 0. \tag{5}$$

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 [16], 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. 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. We then get a single variable polynomial $f(X)$, that equals

$$f(X) \triangleq f(X_1 = p_1(X), \ldots, X_\nu = p_\nu(X)). \tag{6}$$

We know from (5) that the polynomial $f(X)$ in (6) is not identically zero. Moreover, it is easy to see that 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 [1], [11], [12].

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 (6) 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.

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 III.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 III.2.

3) Select $L = m$ and $X = \mathbf{A}$. The following Lemma III.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 III.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 invertible 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 III.2. *[18] The $m \times m$ matrix*

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & \ldots & 0 & a_0 \\ 1 & 0 & 0 & \ldots & 0 & a_1 \\ 0 & 1 & 0 & \ldots & 0 & a_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 1 & a_{m-1} \end{bmatrix} \tag{7}$$

has the characteristic polynomial $g(x) = a_{m-1}x^{m-1} + \ldots + a_0$ and thus satisfies $g(\mathbf{A}) = 0$.

Algorithm Analysis

Our algorithm attempts to minimize the size L of the employed coding matrices, which is equal to the smallest degree polynomial $g(X)$ co-prime to $f(X)$ that we can find in polynomial time. In Theorem III.3 we provide an upper bound on the degree m of $g(X)$ that we will need to employ (hard guarantees). In Lemma III.4 we show that the fraction of polynomials that have a co-prime 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 III.3.

Theorem III.3. *If $f(x)$ is a nonzero binary polynomial of degree n, then there exists a co-prime 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, let $g_1, g_2, \ldots g_K$ be all

[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 $p_i(X)$.

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 co-prime with f. In [12], 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, since 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 co-prime 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.

Lemma III.4. *The fraction of polynomials that have a co-prime 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 $\frac{2^i}{i}$.

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. Thus $f(x)$ 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 $g_i(x)$. ■

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, a fraction of $\frac{1023}{1024}$ of 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.

The proof of the following lemma can also be found in [12].

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

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

B. Code design for scalar coding

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

Same as in Step 1 in Section III-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) Similar to Step 2 in Section III-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 III-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 $f(X_1, \ldots, X_\nu)$ to a single variable polynomial $f(X)$ that has a co-prime factor of degree as small as possible. The challenge in this formulation is 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 co-prime 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.

IV. Scalar vs. Vector Operation: A Comparison

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 $\mathbb{F}_{2^m}$. 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 IV.1 implies that, if we can solve the scalar network coding problem for a network over a field $\mathbb{F}_{2^m}$, then we are able to solve the vector network coding problem using binary matrices of dimension $m \times m$. Therefore, it guarantees that binary matrices are at least as useful as finite fields. This is a well known result in algebraic coding [16].

Theorem IV.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.*

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 IV.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, we can translate this to a scalar solution over a field of size $\mathbb{F}_{2^m}$.

Theorem IV.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 [16]. If we take the field generated by α, it is of size at most 2^m and it contains α. ∎

We underline that Theorem IV.2 holds under two basic assumptions: (i) the variables $\{X_k\}$ are pairwise commuting, in order to be able to write the matrix polynomial, and (ii) 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 larger set of choices than restricting the search to finite fields, as the following examples illustrate.

Example IV.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 co-prime with $f(x)$. Now, take any binary matrix with characteristic polynomial $g(x)$ and use Lemma III.1 to construct the vector coding solution.*

In the next example, we describe a polynomial that is zero for all the assignments to its variables of scalar values from the field $\mathbb{F}_{2^i}$ for $i = 1, 2, \ldots, 10$ while there exists an assignment to the variable of 10×10 binary matrices which makes the polynomial evaluate to an invertible matrix.

Example IV.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 $G(x, y) = ((x^4 + x)(x^8 + x) + (y^4 + y)(y^8 + y))(x^{32} + x + y^{32} + y)$. It is not difficult to see that h is zero over the fields $\mathbb{F}_2, \mathbb{F}_4, \ldots, \mathbb{F}_{1024}$. Now we construct two matrices X_1 and X_2 so that $h(x = X_1, y = X_2)$ is an invertible matrix. For example the following matrices can be used:*

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

where $A_1 = I_2$, $A_2 = I_3$ and

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

V. Conclusions

In this paper, we developed new algebraic algorithms for the problem of vector and scalar network code design. The main idea in our approach is to reduce the problem of code design to an algebraic problem of finding co-prime factors of a given polynomial. 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.

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] R. Ahlswede, N. Cai, S-Y. R. Li, and R. W. Yeung, "Network information flow", *IEEE Trans. Inform. Theory,* vol. 46, 2000.

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

[8] 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.

[9] 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.

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

[11] J. Ebrahimi and C. Fragouli, "Multicasting algorithms for deterministic networks", *IEEE ITW*, Cairo, January 2010.

[12] J. Ebrahimi and C. Fragouli, "Vector network coding", *EPFL Technical Report*, $http://infoscience.epfl.ch/record/$144144, 2010.

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

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

[15] 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.

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

[17] 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.

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

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

[20] J. T. Schwartz, "Fast probabilistic algorithms for verification of polynomial identities",