On Benefits of Vector Network Coding

Javad Ebrahimi B.
School of Computer and Communication Sciences
EPFL
Lausanne, Switzerland.
Email: javad.ebrahimi@epfl.ch

Christina Fragouli
School of Computer and Communication Sciences
EPFL
Lausanne, Switzerland.
Email: christina.fragouli@epfl.ch

***Abstract*—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. Vector network coding generalizes scalar coding, and thus offers a wider range of solutions over which to optimize. This paper starts exploring the new possibilities vector network coding can offer along two directions.**

First, we propose a new randomized algorithm for vector network coding. We compare the performance of our proposed algorithm with the existing randomized algorithms in the literature over a specific class of networks. Second, we explore the use of structured coding matrices for vector network coding. We present deterministic designs that allow to operate using rotation coding matrices and thus result in reduced encoding complexity.

I. Introduction

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. 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. This paper starts exploring the new possibilities vector network coding can offer along two directions: (i) design of randomized algorithms and (ii) use of structured matrices. Both directions build on the deterministic algorithms for vector code design we proposed in [11]–[13].

Randomized network coding is perhaps the most popular approach for network code design as it offers a simple, decentralized operation [4]. The standard approach has each node select uniformly at random scalar coefficients over a finite field. In this paper we propose an alternative two stage approach, where nodes first randomly select coefficients that are polynomials of a single variable and then assign a value to the variable. We analytically evaluate the performance of our proposed algorithm for a specific class of networks. This approach is inspired from and developed for vector network coding, however we show that it can also be translated to a new design for randomized scalar coding.

Use of structured matrices for vector coding has been explored in the literature to achieve reduced encoding complexity. For example, the authors in [14] propose probabilistic designs that employ permutation matrices for the coding matrices, while the authors in [15] employ rotation matrices. We here illustrate through several examples how we can employ our developed framework in [11]–[13] for deterministic designs employing structured matrices. We present sufficient conditions under which we can select the coding matrices to be a generalized form of rotational matrices. This technique can be applied to other families of structured matrices as well.

The paper is organized as follows. Section II provides our notation and reviews previous work; Section III develops our randomized code design algorithm and compares the error probabilities in the vector and scalar cases for a specific network; Section IV looks at the use of structured matrices for the design of network codes; and finally Section V summarizes our results and discusses directions for further studies on this subject.

II. Background

We here introduce the notation we will use in the paper, and review first the algebraic framework in [1], [11] and then the deterministic algorithms for vector network coding in [12]. We review in some detail these results to provide a more complete image as they form the foundation on which we develop this work.

A. Algebraic Framework

We here review the algebraic framework with emphasis on the vector coding formulation. 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 $M_L(\mathbb{F}_q)$. 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], [23], we can assign to the variables X_k values randomly in from a large enough field, and get a valid solution with probability that goes to one as the field size increases [4].

For vector network coding, we can apply a result which states that if the matrices $\{X_k\}$ are commuting, then $\det M_L(\mathbb{F}_q) = \det f(X_1, \ldots, X_\nu)$ [11]. We thus have the following formulations, which in the vector case are not equivalent (as for the second formulation we restrict our attention to commuting matrices).

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.

B. Deterministic Designs for Vector and Scalar Network Coding in [12].

Consider the transfer matrices $\mathbf{M}_j$, $1 \leq j \leq N$, and $\mathbf{M}$. The code design in [12] 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.

Step 1: Reducing the multivariate to a single variable polynomial.: 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.That is,

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

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 [19], 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 $\mathbf{M}$.

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

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

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 prove in [12] that we can always find such a $g(X)$ of degree $m \leq \log(N)$ in polynomial time.

2) If $g(X)$ has degree m, create an $m \times m$ matrix $\mathbf{A}$ so that $g(\mathbf{A}) = 0$.

3) Select $L = m$ and $X = \mathbf{A}$. We prove in [12] 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 $p_i(\mathbf{A})$.

We close this section by noting that there exists a well-known homomorphism that allows to translate scalar coding designs from a field of size 2^m to binary vector code designs using length $L = m$, summarized for completeness in the following theorem [19].

Theorem II.1. *For every positive integer n and every prime number p there exists a ring monomorphism (one-to-one homomorphism) from the finite field $\mathbb{F}_{p^n}$ into the ring $M_n(\mathbb{F}_p)$, the ring of all $n \times n$ matrices over the field $\mathbb{F}_p$.*

III. A Randomized Algorithm for Vector Coding

We are here interested in randomized algorithms for vector network coding. Clearly, a straightforward approach is to apply the randomized algorithm proposed in [4], using a field of size $q = p^n$, with p a prime number, and translate the randomly selected scalar values to coding matrices through the homomorphism described in Theorem II.1. The techniques we developed in [12], [13] allow to develop alternative randomized algorithms, that can offer different trade offs, in terms of probability of error, length of used vectors, and overhead to convey the selected random combinations. We provide such an algorithm in the following. We will denote by $\mathbb{F}_q[X]_{\leq n}$ the ring of all polynomials of degree at most n and with coefficients in the field $\mathbb{F}_q$.

Proposed Algorithm

1) Let $g(X)$ be an irreducible polynomial over the finite field F_q of degree $1 + \mu h(k-1)$, where h is the min-cut to each receiver, μ is the length of the longest source-destination path in the network, and k a parameter we will discuss. Let $\mathbf{A}$ be a matrix whose characteristic polynomial is $g(X)$. This matrix is communicated in advance to all network nodes.
2) Suppose that the node i that has d_i incoming edges, and receives from them the vectors $\mathbf{v}_1$, $\mathbf{v}_2$, ..., $\mathbf{v}_{d_i}$. For each outgoing edge, the node randomly selects the d_i coding coefficients $\{X_j\}$, by selecting d_i polynomials $\{p_j(X)\}$ of degree at most $k-1$, that is, $p_j(X) \in \mathbb{F}_q[X]_{\leq k-1}$. Each such polynomial is chosen uniformly at random from the polynomials in $\mathbb{F}_q[X]_{\leq k-1}$. It will then send the vector $p_1(\mathbf{A}) \cdot \mathbf{v}_1 + p_2(\mathbf{A}) \cdot \mathbf{v}_2 + \ldots + p_{d_i}(\mathbf{A}) \cdot \mathbf{v}_{d_i}$.

The intuition behind the proposed algorithm is as follows. Consider the traditional randomized code design that operates in a specific finite field of a given size $q = 2^k$, and think of the representation of the elements as polynomials of degree at most $k-1$, where operations are performed modulo a fixed irreducible polynomial $g(X)$. In the standard design, we select the polynomials $\{p_j(X)\}$ so that, once we make the substitution $f(X) = f(X_1 = p_1(X), \ldots, X_\nu = p_\nu(X))$, the resulting polynomial $f(X)$ *does not have* $g(X)$ *as a factor*. In contrast, in our approach, we make the substitution in a manner that ensures $f(X)$ is not identically zero. This is a milder condition we are aiming for, and as we will argue in the following, this is the only step that introduces randomness in our algorithm. Thus we hope to achieve a good performance in terms of probability of error.

Achieving a non-zero $f(X)$ implies that the transfer matrix towards each receiver $f_i(X)$, $i = 1 \ldots N$, where N is the number of receivers, is also nonzero. The degree of the transfer matrix towards each receiver has a degree that is upper bounded as $\mu h(k-1)$. Thus, if we select an irreducible polynomial $g(X)$ that has degree $1 + \mu h(k-1)$, $g(X)$ is definitely not a factor of any of the $f_i(X)$, for $i = 1 \ldots N$, and as a result, it is also not a factor of $f(X)$. In other words, as we discussed in Section II-B, the matrix $f(\mathbf{A})$ is invertible. In other words, if the polynomial $f(X)$ is nonzero, then we have ensured by the pre-selection of the matrix $\mathbf{A}$ that each receiver has a full rank set of equations to solve.

Application to Combination Network

The analysis of the error probability of the standard randomized algorithm [4], applies for any transfer matrix where the degree in each variable is the same. This is no longer the case for our algorithms, where the specific form the transfer matrix has might significantly affect the resulting probability of error. Part of our ongoing work is in evaluating what classes of transfer matrix polynomials are more relevant for arbitrary networks. Here, we analytically calculate the error probability for the class of combination networks, where the min-cut to each receiver equals $h = 2$. Such a network is depicted in Fig. 1. It comprises of a source, that is directly connected to a number of $M = 4$ bottleneck edges, and receivers each observing a subset of these edges. As a result, the transfer matrix polynomial to each receiver has the simple form $X_1X_2 - X_3X_4$. Note that the combination network with $h = 2$ offers a very general class of networks; for example, all graph coloring problems, over an arbitrary graph, can be reduced to the problem of network code design over a combination network with $h = 2$ [17]. Note that this network also models wireless one-hop networks, where a source transmits information to a set of M relays, while a mobile receiver is at any time connected to an arbitrary set of two such relays. The source does not know in advance the set of two relays that the receiver is connected to; randomized network coding is thus employed by the relays[1] to ensure that the receiver has a full rank of equations to solve.

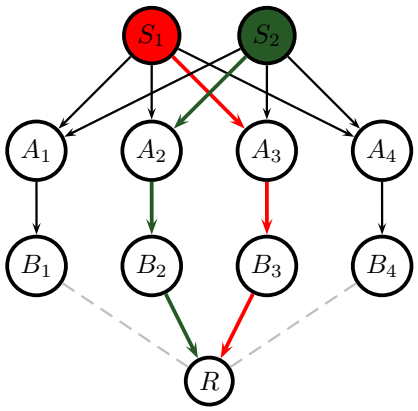

Fig. 1. A mobile receiver connects to a subset of bottleneck edges. The source has no information of the edges where the receiver is connected.

A practical consideration

The receiver, in order to be able to decode, needs to know what are the exact linear combination it received. If we employ randomized coding over a finite field of size $q = 2^k$, then each relay needs to send $hk = 2k$ bits to the receiver, i.e., k bits per coefficient. In our approach, we will instead have each relay select a binary polynomial of degree at most $k-1$: to convey this information, we still need to send $2k$ bits, i.e., k bits per polynomial. This is because, in both cases we are choosing values for variables from sets of the same sizes. In the scalar case we are choosing the values from $\mathbb{F}_{2^k}$ and in vector case we choose them from the set of all polynomials of degree at most $k-1$.

[1]If the number of relays M is known in advance, and there is perfect time synchronization, deterministic network code design methods can also be used for this type of networks. This is however not always the case in practical networks.

Error Probability Analysis

As we discussed, we will compute the probability of the event $X_1X_2 - X_3X_4 = 0$ in the following scenarios:

Randomized Scalar Network Coding [4]

Here we assume that the X_i's are i.i.d according to the uniform distribution on the finite field $\mathbb{F}_{2^k}$. In this scenario if X_1, X_2 and X_3 are taken randomly and $X_3 \neq 0$ then there exists at least one value for X_4 so that $X_1X_2 - X_3X_4 = 0$. So we have $P(X_1X_2 - X_3X_4 = 0) = P(X_3 = 0)P(X_1X_2 - X_3X_4 = 0 \Big| X_3 = 0) + P(X_3 \neq 0)P(X_1X_2 - X_3X_4 = 0 \Big| X_3 \neq 0) \geq \frac{1}{2^k}(1 - (\frac{2^k-1}{2^k})^2) + \frac{2^k-1}{2^k} \times \frac{1}{2^k} \approx \frac{1}{2^k}$.

Proposed Algorithm

Here the X_i's are selected i.i.d according to the uniform distribution on the set $\mathbb{F}[X]_{\leq k-1}$ of the binary polynomials with degree at most $k-1$. Let $p \triangleq Pr(X_1X_2 - X_3X_4 = 0 \Big| X_i \in \mathbb{F}[X]_{<k}, i = 1, \ldots, 4)$. To be able to approximate p, we need the following useful interpretation of error event. The error occurs when we assign random polynomials to the variables X_1, X_2, X_3, X_4 so that $X_1X_2 = f$ and $X_3X_4 = f$ for some polynomial f of degree at most $2k-2$. Therefore the error probability p is upper bounded by the following quantity:

$$
\begin{aligned}
& p \\
\overset{(a)}{\leq} & \sum_{f \in \mathbb{F}_{\leq 2k-2}} P(X_1X_2 = f)P(X_3X_4 = f) \\
\overset{(b)}{\leq} & \sum_{f \in \mathbb{F}_{\leq 2k-2}} P(X_1|f)P(X_2 = \frac{f}{X_1})P(X_3|f)P(X_4 = \frac{f}{X_3}) \\
\overset{(c)}{\leq} & \frac{1}{2^{2k}} \sum_{f \in \mathbb{F}[X]_{\leq 2k-2}} P(X_1|f)P(X_3|f) \\
\overset{(d)}{\leq} & \frac{1}{2^{2k}} \sum_{f \in \mathbb{F}[X]_{\leq 2k-2}} (\frac{M_{2k-2}}{2^k})^2 \\
\overset{(e)}{=} & \frac{1}{2}(\frac{M_{2k-2}}{2^k})^2
\end{aligned}
$$

where M_m is the maximum number of distinct divisors a polynomial of degree at most m can have.

The first inequality comes from the earlier interpretation of the error probability. (b) comes from the fact that for independent random variables X_i, X_j we have $P(X_iX_j = f) = P(X_i|f)P(X_j = \frac{f}{X_i})$. Inequality (c) is the consequence of the fact that for every fixed polynomial g, $P(X_i = g) < \frac{1}{2^k}$. Inequality (d) is because $P(X_i|f) \leq \frac{\text{number of divisors of } f \text{ of degree at most } k-1}{2^k}$. Finally, the last inequality comes from simple calculation.

So, in order to obtain an upper bound on $P(X_1X_2 - X_3X_4 = 0)$ in this scenario, we need to find an upper bound on the number of factors of a polynomial of a given degree $n = 2k-2$. More precisely, the error probability in this case is upper bounded by $p \leq \frac{1}{2}(\frac{M_n}{2^k})^2$ where $n = 2k-2$. Now we try to find an upper bound for this quantity, by upper bounding M_n.

Let $\phi(n)$ be the number of irreducible binary polynomials of degree n. From [27] we know that:

$$M_n \leq \prod_{i=1}^{s} (\frac{n + \sum_{j=1}^{s} j\phi(j)}{i \sum_{j=1}^{s} \phi(j)})^{\phi(i)} \tag{7}$$

in which s is upper bounded by $\log(n) + 1$. Since the right hand side of this inequality is an increasing function of s for $s \leq 1 + \log(n)$, we can replace s by $1 + \log(n)$.

Notice that for every positive integer n we have $\frac{2^n - 2.2^{n/2}}{n} \leq \phi(n) \leq \frac{2^n}{n}$. Combining these inequalities we obtain:

$$
\begin{aligned}
M_n & \overset{(f)}{\leq} \prod_{i=1}^{1+\log(n)} (\frac{n + \sum_{j=1}^{1+\log(n)} j\phi(j)}{i \sum_{j=1}^{1+\log(n)} \phi(j)})^{\phi(i)} \\
& \overset{(g)}{\leq} \prod_{i=1}^{1+\log(n)} (\frac{n + \sum_{j=1}^{1+\log(n)} 2^j}{i \sum_{j=1}^{1+\log(n)} \phi(j)})^{\phi(i)} \\
& \overset{(h)}{\leq} \prod_{i=1}^{1+\log(n)} (\frac{5n}{i \sum_{j=1}^{1+\log(n)} \phi(j)})^{\phi(i)} \\
& \overset{(i)}{\leq} \prod_{i=1}^{1+\log(n)} \Big(\frac{5n}{i \sum_{j=1}^{1+\log(n)} (\frac{2^j - 2.2^{j/2}}{j})}\Big)^{\phi(i)} \\
& \overset{(j)}{\leq} \prod_{i=1}^{1+\log(n)} \Big(\frac{5n}{i \frac{2^{1+\log(n)}}{2(\log(n)+1)}}\Big)^{\phi(i)} \\
& \overset{(k)}{\leq} \prod_{i=1}^{1+\log(n)} \Big(\frac{5n}{i \frac{n}{\log(n)}}\Big)^{\phi(i)} \\
& \overset{(l)}{=} \prod_{i=1}^{1+\log(n)} \Big(\frac{5\log(n)}{i}\Big)^{\phi(i)} \overset{(m)}{<} \prod_{i=1}^{1+\log(n)} (5\log(n))^{\phi(i)} \\
& \overset{(n)}{\leq} (5\log(n))^{\sum_{i=1}^{1+\log(n)} \phi(i)} \overset{(o)}{\leq} (5\log(n))^{\frac{n}{\log(n)-2}}
\end{aligned}
$$

The inequality f is due to (7). g is because $\phi(j) \leq 2^j$. h is deduced from g by simple calculation. i is the consequence of the lower bound we have for $\phi(i)$. j, k, l are trivial inequalities. To obtain m, we dropped the denominator of the fraction. Notice that this is a very rough estimation but even by this approximation still we can show that the error probability in this case is smaller than the error probability in scalar case. Finally, n, o are trivial inequalities.

Therefore $p \leq \frac{1}{8}\frac{(5\log(n))^{\frac{2n}{\log(n)-2}}}{2^n} = \frac{2^{\frac{2n\log(5)\log\log(n)}{\log(n)}-3}}{2^n}$.

Notice that for large n, the following upper bound of p is significantly smaller than $\frac{1}{2^k}$ which was the error probability in the scalar case. In fact, this upper bound of the error probability is not so informative for small k. However computer simulation confirms that even for small values of k, still our algorithm has smaller error probability compared to the randomized scalar network coding.

In figure 2, the error probability of our method is compared to the random vector codes using general matrices and also using the random matrices which are either zero or invertible. The size of the used matrices is $k \times k$. Notice that the for $k > 6$ the error probability for the case that we use zero or invertible binary matrices is not exhibited in the graph. This is due to the fact that in the simulation results, all the generated codes were correct.

In figure 3, we compared the error probability of scalar network codes with the error probability of our vector code and also with the theoretical upper bound of the error probability of our code, given in equation (e) and inequality (7).

We must mention that both of these simulation results are based on $100,000,000$ randomly chosen values for the variables X_1, X_2, X_3 and X_4.

IV. Special Types of Vector Network Codes

In this section we will introduce a familiy of matrices for which we have fast matrix multiplication algorithms (see [25])

Rotational matrices over the binary field are proposed in [15] 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.

An $L \times L$ matrix $\mathbf{A}$ over a field $\mathbb{F}_q$ is called t-th order rotational matrix with respect to the L-tuple $(a_1, a_2, \ldots, a_L)$ in $\mathbb{F}_q$ and denoted by $\text{rot}(t; a_1, a_2, \ldots, a_L)$ if its entries are defined as following:

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

Generally all the indices are taken modulo the size of $\mathbf{A}$.

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

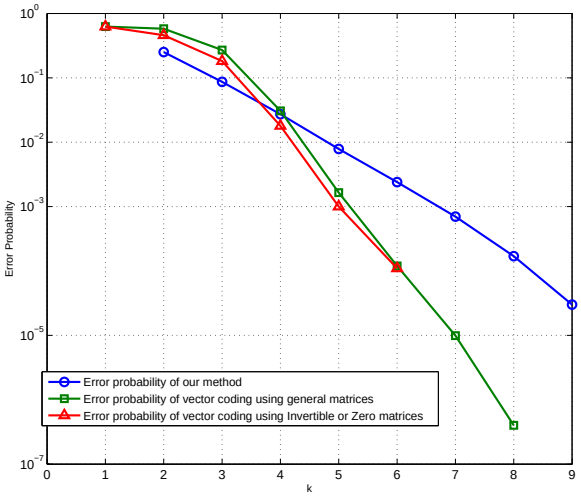

Fig. 2. Comparison between error probabilities of vector coding using general matrices, vector coding using nonzero non-invertible matrices and our method of vector coding.

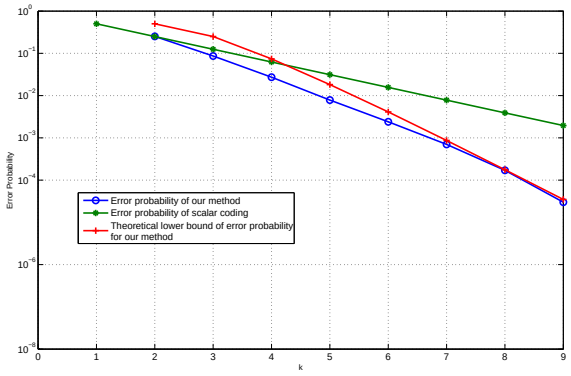

Fig. 3. Comparison between error probabilities in scalar coding, our method of vector coding and theoretical bound of the error probability in our method.

Lemma IV.1. *For every choice of the field elements a_i's and b_i's and any non-negative integers t_1, t_2 we have:* $\text{rot}(t_1; a_1, a_2, \ldots, a_L) \times \text{rot}(t_2; b_1, b_2, \ldots, b_L) = \text{rot}(t_1 + t_2; a_1 b_{t_1+1}, a_2 b_{t_1+2}, \ldots, a_L b_{t_1+L})$.

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 rotational matrices, we can simply select the variable X to be a rotational matrix,

and then have the variables $\{X_i\}$ be powers of X; namely,

$$X_i = X^{m_i} \tag{8}$$

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 $\mathbf{F}_q$, we can then express each α_i in $\mathbf{F}_q$ 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 (8) concludes the first step.

Another easy to show statement about the rotational matrices is the following lemma.

Lemma IV.2. *The characteristic polynomial of the matrix rot*$(1; a_1, a_2, \ldots, a_L)$ *is* $X^L - \prod_{i=1}^{L} a_i$

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

Theorem IV.3. *Consider a non-identically zero polynomial $f(X)$, 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 co-prime 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 co-prime factor exists, use $X = \text{rot}(t;\ 1, 1, \ldots, 1, c)$. ∎

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.*

V. Conclusions and Discussion

In this paper, we started exploring benefits vector network coding can offer. We proposed a randomized algorithm and we analyzed it for a simple combination network. For that particular case we showed that asymptotically our algorithm outperforms the scalar random network coding algorithm. We discussed the use of structured matrices and gave sufficient conditions for employing such matrices with the goal of reducing the encoding complexity.

Several research directions remain open. These include, analyzing the performance of our proposed randomized algorithm for general networks, and potentially refining such algorithms using some knowledge of the network topology. Additionally exploring benefits in terms for example of decoding complexity that use of structured matrices can offer.

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 Trans. Inform. 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 Trans. Inform. 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", *IEEE Allerton*, 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 B. and C. Fragouli, "Multicasting algorithms for deterministic networks", *IEEE ITW*, Cairo, January 2010.

[12] J. Ebrahimi B. and C. Fragouli, "Vector network coding algorithms",*ISIT*, 2010.

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

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

[15] M. Khojastepour, A. Keshavarz-Haddad, "Rotational coding achieves multicast capacity of deterministic wireless networks", *IEEE Allerton*, September 2009. URL : http://www.ece.rice.edu/ alireza/Mypublications/Alireza-DeterministicChannelRotationalCoding.pdf

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

[17] C. Fragouli and E. Soljanin, "Subtree Decomposition for Network Coding", *ISIT*, 2004.

[18] C. Fragouli and E. Soljanin, “Information flow decomposition for network coding,” *IEEE Trans. Inform. Theory*, vol. 52, iss. 3, pp. 829–848, March 2006.
[19] P. Morandi, “Field and Galois Theory”, *Springer*, 1996.
[20] R. M. Gray, “Toeplitz and Circulant Matrices: A Review ”, *Now Publishers*, 2006.
[21] I. Kovacs, D.S. Silver and S.G. Williams, ”Determinants of commuting-block matrices”, *The American Mathematical Monthly*, vol. 106, no. 10, pp. 950-952, December 1999.
[22] R.A. Horn and C.R. Johnson, “Matrix Analysis”, *Cambdrige University Press*.
[23] J.T. Schwartz, “Fast probabilistic algorithms for verification of polynomial identities”, *Journal of the ACM*, vol. 27, no. 4, 1980.
[24] P.J. Davis, “Circulant Matrices”, *Chelsea Publishing*, New York 1994.
[25] G.H. Golub, C.F. Van Loan, “Matrix Computations ”, *Johns Hopkins Studies in Mathematical Sciences*, 1996.
[26] K. Menger, “Zur allgemeinen Kurventheorie,” *Fund. Math.* vol. 10, pp. 95-115, 1927.
[27] Ph. Piret, “On the number of divisors of a polynomial over GF(2)”, *Lecture Notes in Computer Science*,vol 228, October 1984.