

On Linear Network Coding*

Sidharth Jaggi, Michelle Effros

Dept. of Electrical Engineering
California Institute of Technology
Pasadena, CA 91125
{jaggi, effros}@caltech.edu

Tracey C. Ho, Muriel Médard

Dept. of Electrical Engineering
Massachusetts Institute of Technology
Cambridge, MA 02139-4307
{trace, medard}@mit.edu

Abstract

Many recent papers study methods, bounds and limitations for *linear network coding*. We examine prior definitions of linearity in network coding and show their implications in terms of code design and performance. We demonstrate how codes with one notion of linearity can be used to build, in a distributed manner, codes with another notion of linearity. One such reduction provides a simple construction for *convolutional multicast network codes*. We also provide examples of networks for which convolutional network code design is significantly less complex than for any block code. Finally, we introduce the new class of *filter-bank network codes* of which all previous definitions of linear network codes are special cases. This work is motivated by discussions with Dr. Ralf Koetter.

Introduction

Much attention has focused on *network coding* following the results in [1]. A network code allows internal nodes of a network to forward on outgoing links functions of messages on incoming links. This is in contrast to the previous paradigm wherein nodes simply routed packets. As network coding matures, questions of code construction and of connections between network coding and traditional coding theory are gaining importance. Several authors explore design and implementation of algebraic, block and convolutional codes for multicast connections (e.g., [10], [7], [6], [11]). While the relations among these codes are well understood for single link channels, we here investigate relations in the networking context. In particular, we must consider not only issues of code existence. The interplay among the codes at different nodes, and whether choices can be made locally or require global coordination, are crucial to determining the consequences of code design choices on network management and operability.

We restrict ourselves to linear codes over finite fields, vectors or polynomials thereof, as does indeed the bulk of the network coding literature. We term such codes linear codes for brevity. The wealth of prior research in linear algebra provides affords a rich collection of tools for code design and analysis. Further, linear codes are known to be sufficient for some families of network coding problems, including multicast problems [10]. While linear codes may not be optimal for all network coding problems [2], it is conjectured that they are asymptotically optimal [11], as they are for lossless source coding and common coding for noisy channels.

*This research is supported in part by NSF Grant CCR-0325324 and a grant from the Lee Center for Networking

To begin studying linear network codes, we must first define linearity [8]. We study three definitions current in the literature, here called algebraic, block, and convolutional codes. Briefly, algebraic codes permit linear operations over the finite field \mathbb{F}_{2^m} , block codes permit linear operations over vectors of binary field elements, i.e. $(\mathbb{F}_2)^m$, and degree m convolutional codes perform convolutional coding with filters with m memory elements. We distinguish between FIR and IIR convolutional codes. FIR convolutional operations are equivalent to convolution by a polynomial of degree no more than m . IIR convolutional operations are equivalent to convolution by a degree m rational function, which is defined a ratio of two polynomials, neither of which has degree more than m . We discuss combinations and generalizations of these types of linearity in the last section.

We investigate reductions that enable us to transform one type of linear code into another in the context of network coding. Some of the proposed reductions apply only to a limited class of network coding problems (e.g., multicasting), while others to general network coding problems (multiple sources, multiple sinks).

Results currently in the literature deal with proving or disproving the existence of *global reductions* [3, 11, 9]. A global reduction for a class of networks implies that codes defined under one notion of linearity can be replaced at every node by codes, with identical rate regions, defined under another notion of linearity.

However, for practical codes on networks, distributed design and implementation is desirable. We therefore consider *local reductions*. A local reduction is a design algorithm for replacing codes of one type at every node of a network with codes of another type, independently of the global network structure, such that the new codes have identical rate regions.

Input-output nonequivalent local reductions are local reductions that do not necessarily preserve the input-out transfer function of every node. These are of interest because they preserve distributed design for a new type of code when distributed design is possible for the initial family of codes, which nonetheless maintain the feasibility of the network coding problem.

Input-output equivalent local reductions are local reductions that preserve the input-output transfer function of every node. This class is of interest since this would enable different types of linear codes to coexist in a network.

A summary of our and other previously known results is provided in Figure 1. Note that the existence of a local input-output equivalent reduction implies the existence of a local input-output nonequivalent local reduction, which in turn implies the existence of a global reduction. Conversely, a counter-example for global reductions for a class of networks implies that no local input-output nonequivalent reductions for that class of networks exist, which in turn implies that no input-output equivalent reductions for that class of networks exist. In a similar vein, the multicast network coding problem is a subset of the general network coding problem, and a counter-example of a reduction for a multicast network coding problem implies that no such reduction would exist for general network coding problems.

We then discuss complexity issues for different classes of network codes. A delay-free abstraction of network coding is only applicable for networks without cycles. For \mathbb{F}_{2^m} algebraic codes and m -length block codes the delay complexity is m because coding occurs in blocks of m . That of degree m convolutional codes also equals m , since that equals the delay introduced by the linear operation. We show a class of acyclic networks for which convolutional network code design has significantly lower delay

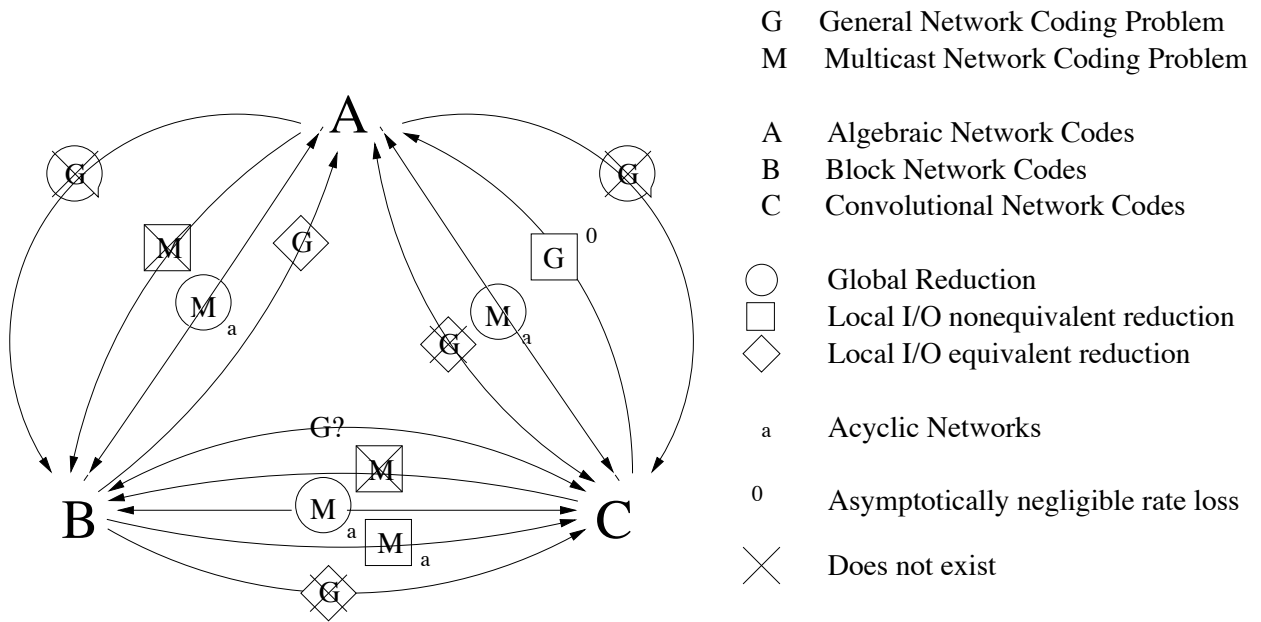


Figure 1: Diagrammatic representation of reductions between different notions of linearity.

complexity than the other two forms of network code design. We also note that another form of complexity is that of *description complexity*. Coefficients of \mathbb{F}_{2^m} algebraic network codes can be described in m bits each, those of FIR convolutional codes require m bits, those of IIR convolutional codes require $2m$ bits, and those of block network codes require m^2 bits.

Finally, we introduce the class of *filter bank network codes*, which subsume each of the three previously mentioned classes of linear network codes and which provide an immediate implementation. We show that under the assumptions of causality, finite memory, and L -shift-invariance (invariance of operations at nodes under shifts by an integer L) these are the most general possible linear codes.

Notation

A *network* is a directed graph $\mathcal{G} = (V, E)$ with unit capacity edges and specified sets of source nodes $S \subseteq V$ and sink nodes $T \subseteq V$. An edge e' is a *predecessor* of edge e , denoted by $e' \rightarrow e$, if the head of e' equals the tail of e . Each $s \in S$ generates a sequence of Bernoulli- $(1/2)$ random variables $\{X_i(s)\}_{i=0}^{\infty}$. Further, $X_i(s)$ is uncorrelated with $X_j(s')$ for all $(i, s) \neq (j, s')$. As defined in [7], a *network coding problem* for graph \mathcal{G} corresponds to an $|S| \times |T|$ binary matrix (denoted by $R_{\mathcal{G}} = \{r_{st}\}$) such that $r_{st} = 1$ if the data generated at source s is required at sink t , and 0 otherwise. Of particular interest is the *multicast network coding problem*, in which for a fixed s , r_{st} is either 0 or 1 for all $t \in T$.

We now define three classes of linear network operations previously discussed in the literature. Let $M(e_i(n))$ denote the message bit carried on edge $e_i \in E$ at the n th coding interval. We note the standard isomorphism between the field \mathbb{F}_{2^m} and equivalence classes of polynomials in $\mathbb{F}_2(z)$ modulo an irreducible polynomial $P_m(z)$ of degree m . Since such polynomials have m coefficients, any element of the finite field \mathbb{F}_{2^m} may be viewed as an m -length vector. We denote this bijection as $\mathcal{B}_{P_m} : \mathbb{F}_{2^m} \rightarrow$

$(\mathbb{F}_2)^m$, and the inverse as $\mathcal{B}_{P_m}^{-1} : (\mathbb{F}_2)^m \rightarrow \mathbb{F}_{2^m}$. In particular, we define the finite field element $\alpha(e_i(j)) = \sum_{n=0}^{m-1} M(e_i(jm+n))z^n$ for all $j \in \{0, 1, \dots, \}$. Further, we define the m -length vector $\vec{\alpha}(e_i(j)) = \{M(e_i(n))\}_{n=jm}^{j(m+1)-1}$ for all $j \in \{0, 1, \dots, \}$, and the polynomial $\alpha(e_i(z)) = \sum_{n=0}^{\infty} M(e_i(n))z^n$. A linear network code \mathcal{C} is said to *solve the network coding problem* $\mathcal{R}_{\mathcal{G}}$ if for all i and $s \in S, t \in T$ such $r_{st} = 1$, sink node t is able to reproduce $X_i(s)$ exactly. Let p be a prime number.

Definition 1. A network code for a network \mathcal{G} is said to be an \mathbb{F}_{p^m} -algebraic network code, denoted by $\mathcal{C}_A(\mathcal{G}, p, m)$, if for all $e_i \in E$ and all $j \in \{0, 1, \dots, \}$

$$\alpha(e_i(j)) = \sum_{i': e_{i'} \rightarrow e_i} \beta_{i', i} \alpha(e_{i'}(j))$$

where all $\alpha(e_i(j)), \alpha(e_{i'}(j)), \beta_{i', i} \in \mathbb{F}_{p^m}$.

Definition 2. A network code for a network \mathcal{G} is said to be an $(\mathbb{F}_p)^m$ -block network code, denoted by $\mathcal{C}_B(\mathcal{G}, p, m)$, if for all $e_i \in E$ and all $j \in \{0, 1, \dots, \}$

$$\vec{\alpha}(e_i(j)) = \sum_{i': e_{i'} \rightarrow e_i} [\beta_{i', i}] \vec{\alpha}(e_{i'}(j))$$

where all $\vec{\alpha}(e_i(j)), \vec{\alpha}(e_{i'}(j)) \in (\mathbb{F}_p)^m$, and $[\beta_{i', i}]$ is an $m \times m$ matrix over \mathbb{F}_p .

Definition 3. A network code for a network \mathcal{G} is said to be a degree m $\mathbb{F}_p(z)$ -convolutional network code, denoted by $\mathcal{C}_C(\mathcal{G}, p, m)$, if for all $e_i \in E$ and all $j \in \{0, 1, \dots, \}$

$$\alpha(e_i(z)) = \sum_{i': e_{i'} \rightarrow e_i} \beta_{i', i}(z) \alpha(e_{i'}(z))$$

where all $\beta_{i', i}(z)$ are rational functions in z of maximum degree m , over \mathbb{F}_p .

Algebraic versus block network codes

General networks

For general networks, there exists an input-output equivalent local reduction from algebraic to block network codes. Even global reductions in the opposite direction are impossible.

Lemma 1. For any network coding problem $\mathcal{R}_{\mathcal{G}}$ solved by an algebraic network code $\mathcal{C}_A(\mathcal{G}, p, m)$ there exists an input-output equivalent local reduction to a block network code $\mathcal{C}_B(\mathcal{G}, p, m)$.

Proof: Our construction uses \mathcal{B}_{P_m} . Let $\vec{u}_i, i \in \{1, \dots, m\}$ be unit vectors in \mathbb{F}_{p^m} . Given any $\beta_{i', i} \in \mathbb{F}_{p^m}$, we define the corresponding $[\beta_{i', i}]$ such that its i th row vector equals $\mathcal{B}_{P_m}(\beta_{i', i} \mathcal{B}_{P_m}^{-1}(\vec{u}_i))$. But this preserves the transfer function at every node, due to the following. For all $i \in \{1, \dots, m\}$, let $u_i \in \mathbb{F}_{p^m}$ equal $(z^{i-1}) \bmod (P_m(z))$. Then the linear span of $\{u_i(z)\}_{i=1}^m$ with scalars from \mathbb{F}_p is the set of polynomials of degree less than m . Hence any $\alpha \in \mathbb{F}_{p^m}$, denoted as polynomial $(\alpha(z)) \bmod (P_m(z))$, may be written as $\sum_{i=1}^m (a_i u_i(z)) \bmod (P_m(z))$, where $a_i \in \mathbb{F}_p$ for all i . Due to the linearity of \mathcal{B}_{P_m} , we are done. \square

Lemma 2 ([11],[9]). There exist network coding problems $\mathcal{R}_{\mathcal{G}}$ that are solved by block network codes $\mathcal{C}_B(\mathcal{G}, p, m)$, such that there are no algebraic network codes $\mathcal{C}_A(\mathcal{G}, p', m')$ which solve $\mathcal{R}_{\mathcal{G}}$ for any p' and m' .

Multicast networks

By Lemma 1 there is a local input-output equivalent reduction from algebraic network codes to block network codes for multicast networks. By [7],[6] algebraic network codes are optimal for acyclic multicast problems, which implies that there exists a global reduction from algebraic network codes to block network codes. Lemma 3 shows that local input-output non-equivalent reductions from algebraic network codes to block network codes are not possible for multicast networks.

Lemma 3. For all n consider a network code $\mathcal{C}_B(\mathcal{G}_n, 2, 2)$ an $(\mathbb{F}_2)^2$ -block network code for the network shown in Figure (a) be such that $[\beta_1] = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ and $[\beta_2] = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$. Then for any finite field \mathbb{F}_{p^m} there exists a network \mathcal{G}_N such that there does not exist an algebraic network code $\mathcal{C}_A(\mathcal{G}_N, p, m)$ which is local input-output non-equivalent to $\mathcal{C}_B(\mathcal{G}_N, 2, 2)$.

Proof: Code $\mathcal{C}_B(\mathcal{G}_N, 2, 2)$ achieves a rate of 1 bit per coding instant. We wish to replace each $[\beta_1]$ matrix on the left branch of Figure with an element, say β_1 , from a suitable finite field \mathbb{F}_{p^m} and replace each $[\beta_2]$ matrix on the right branch with another element, say β_2 . Since \mathbb{F}_{p^m} is a cyclic group of order p^m under multiplication, for any β_1, β_2 in \mathbb{F}_{p^m} , $\beta_1^{p^m} = \beta_2^{p^m} = 1$. Thus, if the network \mathcal{G}_n in Figure is such that $n = N - 1$, the messages from the two branches destructively interfere at the output, and the receiver receives 0 regardless of the input. \square

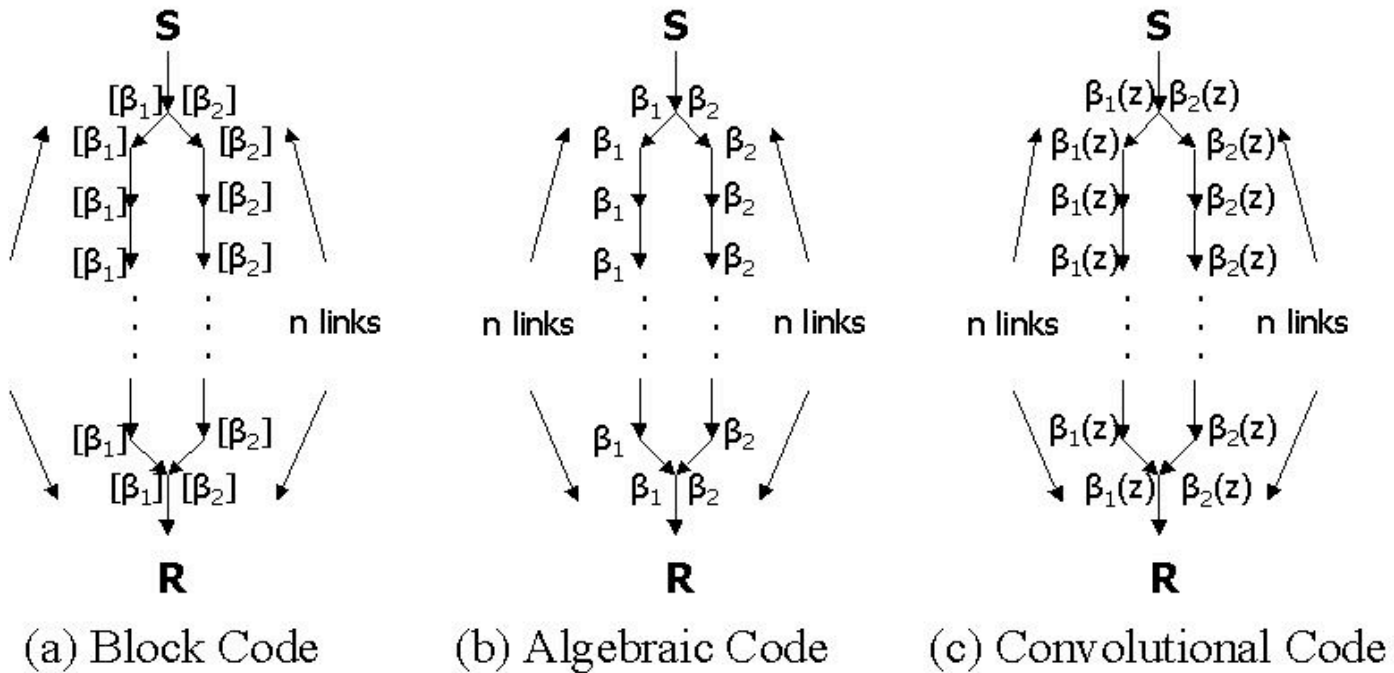


Figure 2: This figure shows a single-sender (S) single-receiver (R) network \mathcal{G}_n , such that both branches of the network have n edges. Sub-figures (a), (b) and (c) respectively show particular block, algebraic and convolutional network codes for \mathcal{G}_n .

Convolutional versus algebraic network codes

General networks

For general networks, there exists no input-output equivalent local reduction in either direction between algebraic and convolutional network codes. For general acyclic networks, we do not know of any other reductions in either direction. In Lemma 4 we distinguish between FIR and IIR convolutional codes.

- Lemma 4.** 1. For any algebraic code $\mathcal{C}_A(\mathcal{G}, 2, 2)$ that contains a single-input single-output node with $\beta_{i',i} = (z) \bmod (z^2 + z + 1)$, there exists no local input-output equivalent convolutional code $\mathcal{C}_C(\mathcal{G}, p, m)$ for any p, m .
2. For any convolutional code $\mathcal{C}_C(\mathcal{G}, 2, 1)$ that contains a single-input single-output node with $\beta_{i',i}(z) = 1/(z + 1)$, there exists no local input-output equivalent algebraic code $\mathcal{C}_A(\mathcal{G}, p, m)$ for any p, m .
3. For any convolutional code $\mathcal{C}_C(\mathcal{G}, 2, 1)$ that contains a single-input single-output node with $\beta_{i',i}(z) = z$, there exists no local input-output equivalent algebraic code $\mathcal{C}_A(\mathcal{G}, p, m)$ for any p, m .

Proof: 1. For the algebraic code, consider the input $M(e_{i'}(n)) = 1$ for all n . Therefore the output on edge i due to the incoming message equals $M(e_{i'}(n))$ for odd n , and $M(e_{i'}(n)) + M(e_{i'}(n - 1))$ for even n . It can be seen that no convolutional filter can mimic this behaviour.

2. For the convolutional code, consider the input $M(e_{i'}(n)) = 1$ for $n = 0$, and 0 otherwise. The corresponding output has infinite support. But this behaviour cannot be mimicked by algebraic codes.

3. Consider the sequence of inputs $M_j(e_{i'}(n)) = \delta(j)$ for all n , where $\delta(j)$ is the Kronecker- δ function. Therefore on input $\delta(j)$, the output is $\delta(j + 1)$. Let us assume that $\beta_{i',i}$, an element of some \mathbb{F}_{p^m} is input-output equivalent to z . But because the block-length of this $\beta_{i',i}$ equals m , therefore on being given the input $\delta(j)$, the output cannot equal $\delta(j + 1)$. \square

Multicast networks

For some networks with cycles, algebraic network codes which achieve the multicast capacity are not possible, but codes with memory achieve capacity [1], [7]. We show a local input-output nonequivalent reduction from algebraic convolutional network codes to convolutional network codes for multicast coding problems over acyclic graphs. We note that this reduction simplifies the arguments presented in [3], [5]. Further, by [7],[6] algebraic network codes are optimal for acyclic multicast problems, which implies that there exists a global reduction between algebraic and convolutional network codes for multicast problems on acyclic networks. Lemma 5 shows that local reductions from algebraic convolutional network codes are not possible for multicast networks.

Lemma 5. For all n let $\mathcal{C}_C(\mathcal{G}_N, 2, 2)$ (Figure (c)) be such that $\beta_1(z) = 1$ and $\beta_2(z) = z$. Then for any finite field \mathbb{F}_{p^m} there exists a network \mathcal{G}_N such that there does not exist an algebraic network code $\mathcal{C}_A(\mathcal{G}, p, m)$ which is local input-output non-equivalent to $\mathcal{C}_C(\mathcal{G}_N, 2, 2)$.

Proof: We note $\mathcal{C}_C(\mathcal{G}_N, 2, 2)$ as defined achieves the capacity of one bit per coding cycle. The remainder of the proof is identical to that of Lemma 3. \square

Lemma 6. For any algebraic network code $\mathcal{C}_A(\mathcal{G}, p, m)$ which solves any multicast network coding problem $\mathcal{R}_{\mathcal{G}}$ there exists an input-output nonequivalent local reduction to a convolutional network code $\mathcal{C}_C(\mathcal{G}, p, m)$.

Proof: Given any $\beta_{i',i} = (B(z)) \bmod (P_m(z))$ in $\mathcal{C}_A(\mathcal{G}, p, m)$ such that $B(z)$ is of degree less than m , we define the corresponding $\beta_{i',i}(z)$ in $\mathcal{C}_C(\mathcal{G}, p, m)$ as $B(z)$. We denote this mapping by $\mathcal{M} : \mathbb{F}_{2^m} \rightarrow \mathbb{F}_2(z)$.

Let $E_{c'} = \{e_{c'_i}\}_{i'}$ and $E_c = \{e_{c_i}\}_i$ be any two subsets of E such that both form cutsets of \mathcal{G} , and $E_{c'}$ is the set of all edges which are predecessors to E_c . Let $\vec{v}(E_{c'}, j)$ be the $|E_{c'}|$ -length vector over \mathbb{F}_{p^m} such that i th entry of $\vec{v}(E_{c'}, j)$ equals $\alpha(e_{c'_i}(j))$, and $\vec{v}(E_c, j)$ be the $|E_c|$ -length vector over \mathbb{F}_{p^m} such that i th entry of $\vec{v}(E_c, j)$ equals $\alpha(e_{c_i}(j))$. Similarly, let $\vec{v}(z)(E_{c'})$ be the $|E_{c'}|$ -length vector over $\mathbb{F}_2(z)$ such that i th entry of $\vec{v}(z)(E_{c'})$ equals $\alpha(e_{c'_i}(z))$, and $\vec{v}(z)(E_c)$ be the $|E_c|$ -length vector over $\mathbb{F}_2(z)$ such that i th entry of $\vec{v}(z)(E_c)$ equals $\alpha(e_{c_i}(z))$. Then for all $E_{c'}, E_c$ and j , any network code $\mathcal{C}_A(\mathcal{G}, p, m)$ is defined by a linear map $L_{E_{c'} \rightarrow E_c} : (\mathbb{F}_{p^m})^{|E_{c'}|} \rightarrow (\mathbb{F}_{p^m})^{|E_c|}$ from $\vec{v}(E_{c'}, j)$ to $\vec{v}(E_c, j)$. Given any such $L_{E_{c'} \rightarrow E_c}$, we define $L(z)_{E_{c'} \rightarrow E_c} : (\mathbb{F}_2(z))^{|E_{c'}|} \rightarrow (\mathbb{F}_2(z))^{|E_c|}$ as $L(z)_{E_{c'} \rightarrow E_c}(\beta(z)) = \mathcal{M}(L_{E_{c'} \rightarrow E_c}(\mathcal{M}^{-1}(\beta(z))))$. This linear transform $L(z)_{E_{c'} \rightarrow E_c}$ describes the linear transformation between $\vec{v}(z)(E_{c'})$ and $\vec{v}(z)(E_c)$ implemented by the convolutional network code $\mathcal{C}_C(\mathcal{G}, p, m)$. But the rank of $L(z)_{E_{c'} \rightarrow E_c}$ is at least that of $L_{E_{c'} \rightarrow E_c}$ (since the $(\cdot) \bmod (P_m(z))$ operation is linear) and therefore if $\mathcal{C}_A(\mathcal{G}, p, m)$ solved a particular network multicast problem, then so does $\mathcal{C}_C(\mathcal{G}, p, m)$. \square

Convolutional versus block network codes

General networks

For general networks, there does not exist any global reduction from block network codes to convolutional network codes [11],[9]. For some networks with cycles, block network codes which achieve capacity are not possible due to feedback, but convolutional network codes achieve capacity [1], [7]. If we weaken our requirement of reductions so that an asymptotically negligible rate-loss is allowed, we can demonstrate the existence of an input-output nonequivalent local reduction from convolutional network codes to block network codes.

Lemma 7 ([11],[9]). There exist network coding problems $\mathcal{R}_{\mathcal{G}}$ that are solved by block network codes $\mathcal{C}_B(\mathcal{G}, p, m)$, such that there are no convolutional network codes $\mathcal{C}_C(\mathcal{G}, p, m)$ which solve $\mathcal{R}_{\mathcal{G}}$ for any p' and m' .

Given $R_{\mathcal{G}} = \{r_{st}\}$, we define the ϵ -loss network coding problem $R_{\mathcal{G}}^{\epsilon} = \{r_{st}^{\epsilon}\}$ as a matrix such that $r_{st} = 1$ implies that $r_{st}^{\epsilon} = 1 - \epsilon$, and $r_{st} = 0$ implies that $r_{st}^{\epsilon} = 0$. An ϵ -loss network coding problem $R_{\mathcal{G}}^{\epsilon}$ is said to be solvable if there exists a block-length m such that for all $s \in S$ and $t \in T$ such that $r_{st}^{\epsilon} = 1 - \epsilon$, for all $i \in \{1, \dots, \lceil m(1 - \epsilon) \rceil\}$ sink t decodes $X_i(s)$ with no error. We also define the *decoding delay of a convolutional network code* d as the number of coding instants from the start of encoding operations before all sinks can start decoding their messages.

Lemma 8. Given network \mathcal{G} with decoding delay d , any convolutional network code which solves $R_{\mathcal{G}}$ and any $\epsilon > 0$, there exists a block network code which solves the $R_{\mathcal{G}}^{\epsilon}$ network coding problem with block-length $m = d/\epsilon$.

Note: The factor d/m decreases with n , and is asymptotically equal to 0.

Proof: Each s generates $\{X_0(s), X_i(s), \dots, X_{m-d-1}, 0, \dots, 0\}$ in the first m intervals. Each internal node mimics the convolutional code for m instants. Each sink is able to decode the first $m - d$ symbols of the desired source symbols, by the definition of decoding delay. The resulting $m \times m$ matrices at each node correspond to the truncated transfer functions of the convolutional network codes. \square .

Multicast networks

For some networks with cycles, algebraic network codes which achieve the multicast capacity are not possible due to feedback, but convolutional network codes achieve capacity [1], [7]. Input-output equivalent local reductions cannot exist in either direction, as the following arguments show.

- Lemma 9.** 1. For any block network code $\mathcal{C}_A(\mathcal{G}, 2, 2)$ that contains a single-input single-output node with $[\beta_{i',i}] = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$, there exists no local input-output equivalent convolutional network code $\mathcal{C}_C(\mathcal{G}, p, m)$ for any p, m .
2. For any convolutional network code $\mathcal{C}_C(\mathcal{G}, 2, 1)$ that contains a single-input single-output node with $\beta_{i',i}(z) = 1/(z + 1)$, there does not exist any local input-output equivalent block network code $\mathcal{C}_B(\mathcal{G}, p, m)$ for any p, m .
3. For any convolutional network code $\mathcal{C}_C(\mathcal{G}, 2, 1)$ that contains a single-input single-output node with $\beta_{i',i}(z) = z$, there does not exist any local input-output equivalent algebraic network code $\mathcal{C}_B(\mathcal{G}, p, m)$ for any p, m .

Proof: 1. For the block code, consider the input $M(e_{i'}(n)) = 1$ for all n . Therefore the output on edge i due to the incoming message equals $M(e_{i'}(n))$ for odd n , and 0 for even n . However, no convolutional filter can mimic this behaviour.

2., 3. The proofs of 2 and 3 are identical to that in Lemma 4. \square

Complexity

The previous sections focus on the feasibility of network coding problems under different notions of linearity. For some networks, codes of one sort may be of significantly lower complexity than codes of another sort. While this is true for some cyclic networks, we present a result which shows that convolutional network codes may offer lower implementation complexity for some acyclic networks than any block network codes. It has been shown that to achieve the multicast capacity of 2 for the network in Figure 3, any block code (including non-linear block codes) requires a block-length m of at least $\lceil 1/2 \log(|T|) \rceil$ [6],[4],[9]. However, degree- $\lceil 1/4 \log(|T|) \rceil$ convolutional codes achieve the multicast capacity. To prove this, we first state a lemma by Morrison [12].

Lemma 10 ([12], Section 3). *The number of pairs of coprime polynomials of degree at most m over a finite field of size q equals $q^{2m+2} - q^{2m+1} + q - 1$.*

Corollary 11. *The number of 2-length vectors of polynomials over \mathbb{F}_2 $v = (\beta_1(z), \beta_2(z))$ (where $\beta_1(z)$ and $\beta_2(z)$ are of degree at most m) such that any two v are linearly independent is at least $2^{2m+1} + 1$.*

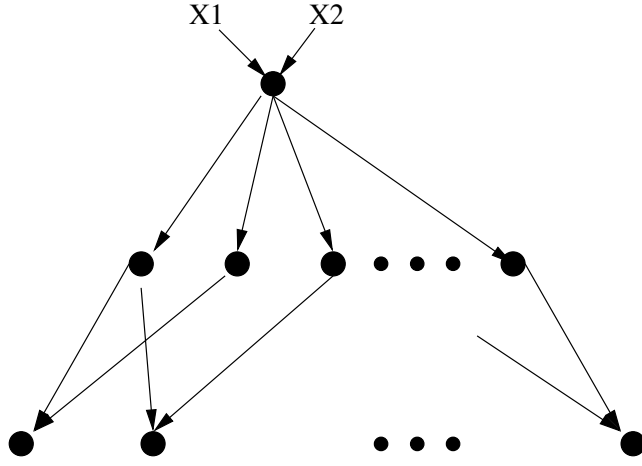


Figure 3: An example of a network where codes with memory require smaller block-lengths than codes without memory.

Proof: Given any two vectors $(\beta_1(z), \beta_2(z))$ and $(\beta'_1(z), \beta'_2(z))$ such that none of the components are zero, they are linearly independent if and only if $\beta_1(z)/\beta_2(z) \neq \beta'_1(z)/\beta'_2(z)$. But the number of distinct values that $\beta_1(z)/\beta_2(z)$ can take is at least the number of pairs of coprime polynomials. Calculating the value in Lemma 10 for $q = 2$ gives us the required result. \square

Therefore, we construct a convolutional code for the network in Figure 3 as follows. The message transmitted to every intermediate node has a distinct global coding vector from the set of vectors described in Lemma 11. Since any two distinct vectors span all of $(\mathbb{F}_2(z))^2$, therefore all the receiver nodes can decode everything. This implies the block-length m sufficiency claimed earlier for the network in Figure 3. We conjecture that degree IIR- $\lceil 1/8 \log(t) \rceil$ convolutional codes suffice for this network coding problem.

General Formulation for Linear Network Codes

In this section we give a general formulation for linear network codes under reasonable additional restrictions. Let us consider linear systems that are causal, have finite memory, and are L -shift invariant, i.e., operations at every node are periodic with period L . But this restricts the set of permissible encoding operations at intermediate nodes to be of the form

$$y(iL + j) = \sum_{k=1}^m c_{jk} y(iL + j - k) + \sum_{k=0}^m d_{jk} x(iL + j - k), \quad j \in \{0, \dots, L - 1\}$$

Note that the above formulation of network codes leads naturally to a state-space formulation for describing encoding operations at a node, and therefore for the entire network. These sets of operations can be implemented by filter-banks.

Conclusion

In this paper we analyze relationships between algebraic, block and convolutional network codes. When two different types of linear network codes can solve the same network coding problem we say that they are globally equivalent. We highlight examples in current literature of network coding problems where such global equivalence is

possible and when it is not. Since distributed design in networks is of value, we also define two local notions of equivalence - those of local input-output equivalent, and input-output non-equivalent. Both of these convert, in a localized manner, codes from one notion of linearity to another. In particular, the former preserves the input-output impulse response of each node, which enables nodes with different notions of linearity to co-exist in a network. We give examples of such reductions between all three types of linear network codes, and also point out when such reductions are not possible. We also present a result showing that the complexity of implementation for convolutional network codes can for some networks be significantly lower than that of any kind of block codes. Lastly, we present a new class of codes called filter-bank network code which subsume each of the three previously defined types of linear network codes, and show that under the reasonable extra conditions of causality, L -stationarity and finite memory, filter bank network codes are the most general possible class of network codes.

References

- [1] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, 2000.
- [2] R. Dougherty, C. Freiling, and K. Zeger. Insufficiency of linear coding in network information flow. submitted to the *IEEE Transactions on Information Theory*, 2004.
- [3] E. Erez and M. Feder. Convolutional network codes. In *IEEE International Symposium on Information Theory*, 2004.
- [4] M. Feder, D. Ron, and A. Tavorly. Bounds on linear codes for network multicast. In *Electronic Colloquium on Computational Complexity (ECCC) 10(033)*, 2003.
- [5] C. Fragouli and E. Soljanin. A connection between network coding and convolutional codes. In *2004 IEEE International Conference on Communications*, pages 661–666, 2004.
- [6] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain, and L. Tolhuizen. Polynomial time algorithms for multicast network code construction. submitted to the *IEEE Transactions on Information Theory*, 2003.
- [7] R. Koetter and M. Médard. Beyond routing: An algebraic approach to network coding. In *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOMM)*, volume 1, pages 122–130, 2002.
- [8] Ralf Koetter. What is linearity? personal communication.
- [9] A. Rasala Lehman and E. Lehman. Complexity classification of network information flow problems. In *Proceedings of the 41st Allerton Conference on Communication, Control and Computing*, 2003.
- [10] S.-Y. R. Li, R. W. Yeung, and N. Cai. Linear network coding. *IEEE Transactions on Information Theory*, 49(2):371–381, 2003.
- [11] M. Médard, M. Effros, T. Ho, and D. Karger. On coding for non-multicast networks. In *Proceedings of the 41st Allerton Conference on Communication, Control and Computing*, 2003.
- [12] K. E. Morrison. Random polynomials over finite fields. In *Combinatorics of Algebraic Structures*, <http://www.calpoly.edu/kmorrison/Research/RPFF.pdf>, 1999.