Localized Dimension Growth in Random Network Coding: A Convolutional Approach

Wangmei Guo, Ning Cai
The State Key Lab. of ISN,
Xidian University, Xi'an, China
Email: {wangmeiguo, caining}@mail.xidian.edu.cn

Xiaomeng Shi, Muriel Médard
Research Laboratory of Electronics,
MIT, Cambridge, USA
Email: {xshi, medard}@mit.edu

***Abstract*—We propose an efficient *Adaptive Random Convolutional Network Coding* (ARCNC) algorithm to address the issue of field size in random network coding. ARCNC operates as a convolutional code, with the coefficients of local encoding kernels chosen randomly over a small finite field. The lengths of local encoding kernels increase with time until the global encoding kernel matrices at related sink nodes all have full rank. Instead of estimating the necessary field size a priori, ARCNC operates in a small finite field. It adapts to unknown network topologies without prior knowledge, by locally incrementing the dimensionality of the convolutional code. Because convolutional codes of different constraint lengths can coexist in different portions of the network, reductions in decoding delay and memory overheads can be achieved with ARCNC. We show through analysis that this method performs no worse than random linear network codes in general networks, and can provide significant gains in terms of average decoding delay in combination networks.**

***Index Terms*—convolutional network code, random linear network code, adaptive random convolutional network code, combination networks**

I. Introduction

Since its introduction [1], network coding has been shown to offer advantages in throughput, power consumption, and security in both wireline and wireless networks. Field size and adaptation to unknown topologies are two of the key issues in network coding. Li et al. showed constructively that the max-flow bound is achievable by linear algebraic network coding (ANC) if the field is sufficiently large for a given deterministic multicast network [2], while Ho et al. [3] proposed a distributed random linear algebraic network code (RLNC) construction that achieves the multicast capacity asymptotically in field size. Because of its simplicity and the ability to adapt to unknown topologies, RLNC is often preferred over deterministic network codes. While the construction in [3] allows cycles, which leads to the creation of convolutional codes, it does not make use of the convolutional nature of the resulting codes to lighten bounds on field size, which may need to be large to guarantee successful decoding at all sinks. Both block network codes (BNC) [4], [5] and convolutional network codes (CNC) [6], [7] can mitigate the field size requirements. Médard et al. introduced the concept of vector, or block network codes (BNC) [4], and Xiao et al. proposed a deterministic binary BNC to solve the combination network problem [8]. BNC can operate on smaller finite fields, but the block length may need to be pre-determined. In discussing cyclic networks, both Li et al. and Ho et al. pointed out the equivalence between ANC in cyclic networks with delays, and CNC [2], [3]. Because of coding introduced across the temporal domain, CNC does not have a field size constraint. Even though degree growth of the encoding kernel may lead to high computation complexity during decoding when there are many coding nodes along a path to a sink, it is often possible to achieve the network coding advantage by coding at a subset of nodes only [9]. In addition, the structure of CNC allows decoding to occur symbol-by-symbol, thus offering gains in decoding delay. However, it may require long coding kernels when the network is unknown. As discussed by Jaggi et al. [10], there exists equivalence relationships between ANC, BNC, and CNC. All three schemes require some prior knowledge on the network topology. Overestimation for the worst case assumption can be wasteful, leading to high computation complexity, decoding delay, and memory overheads.

Our work extends the RLNC and CNC setup, allowing nodes to locally grow the dimensionality of the code until necessary. We propose an efficient adaptive random convolutional network code (ARCNC) for multicast networks, with local encoding kernels chosen randomly from a small field, and the code constraint length incremented locally at each node. Our scheme inherits the small field size property of CNC, while taking advantage of the distributive nature of RLNC. The gains offered by ARCNC are three-fold. First, it operates in a small finite field. Second, it adapts to unknown network topologies without prior knowledge. Last, the localized adaptation allows convolutional codes with different code lengths to coexist in different portions of the network, leading to reduction in decoding delay and memory overheads associated with using a pre-determined field size or code length.

The remainder of this paper is organized as follows: the ARCNC algorithm is proposed in Section II and its performance analyzed in Section III. As an example, the advantages of ARCNC is considered in a combination network in Section IV. Section V concludes the paper.

This work is funded by CSC and the National Natural Science Foundation of China (NSFC) under grant No.60832001, and partially presented in conference AEW 2010; also based upon work supported under subcontract # RA306-S1 issued by the Georgia Institute of Technology, by the Claude E. Shannon Research Assistantship awarded by the Research Laboratory of Electronics, MIT, and by the NSERC Postgraduate Scholarship (PGS) issued by the Natural Sciences and Engineering Research Council of Canada.

II. Adaptive Randomized Convolutional Network Coding Algorithm

A. Basic Model and Definitions

We first introduce definitions used throughout the paper. We model a communication network as a finite directed multigraph, denoted by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. An edge represents a noiseless communication channel on which one symbol is transmitted per unit time. In this paper, we consider the multicast case. The source node is denoted by s, and the set of d sink nodes is denoted by $T = \{r_1, \ldots, r_d\} \subset V$. For every node $v \in \mathcal{V}$, denote the sets of incoming and outgoing channels to v by $In(v)$ and $Out(v)$. An ordered pair (e', e) of channels is called an adjacent pair when there exists a node v with $e' \in In(v)$ and $e \in Out(v)$.

The symbol alphabet is represented by a base field, $\mathbb{F}_q$. Assume s generates a message per unit time, consisting of a fixed number m of symbols represented by an m-dim row vector $x \in \mathbb{F}_q^m$. We index time to start from 0, hence the $(t+1)$-th coding round occurs at time t. Messages transmitted by s is represented by a power series $x(z) = \sum_{t\geq 0} x_t z^t$, where $x_t \in F_q^m$ is the message generated at t and z denotes a unit-time delay. Data propagated over a channel $e \in Out(v)$ is $y_e(z)$, a linear function of the source message; $y_e(z) = x(z) f_e(z)$, where the m-dim column vector of rational power series, $f_e(z) = \sum_{t\geq 0} f_{e,t} z^t$, is called the global encoding kernel over e. Viewed locally, $y_e(z)$ is a linear combination of messages over all incoming channels to node v; $y_e(z) = \sum_{e' \in In(v)} k_{e',e}(z) y_{e'}(z)$, where $k_{e',e}(z) = \sum_{t\geq 0} k_{e',e,t} z^t$ is the local encoding kernel over the adjacent pair (e', e). Hence, $f_e(z) = \sum_{e' \in In(v)} k_{e',e}(z) f_{e'}(z)$. As discussed in [6], $k_{e',e}(z)$ and $f_e(z)$ are rational power series in the form of $\frac{p(z)}{1+zq(z)}$, where $p(z)$ and $q(z)$ are polynomials. Collectively, we call the $|In(v)| \times |Out(v)|$ matrix $K_v(z) = (k_{e',e}(z))_{e' \in In(v), e \in Out(v)}$ the local encoding kernel matrix at node v, and the $m \times |In(v)|$ matrix $F_r(z) = (f_e(z))_{e \in In(r)}$ the global encoding kernel matrix at sink r.

B. Algorithm Statement for Acyclic Networks

1) Encoding: at time 0, all local and global encoding kernels are set to 0. Source s generates a message $x = \sum_{t\geq 0} x_t z^t$, where x_t consists of m symbols $(x_{t,1}, x_{t,2}, \cdots, x_{t,m})$. Each intermediate node v, when a symbol is received on $e' \in In(v)$ at time t, stores it in memory as $y_{e',t}$, and chooses the $(t+1)$-th term $k_{e',e,t}$ of the local encoding kernel $k_{e',e}(z)$ uniformly randomly from $\mathbb{F}_q$ for $e \in Out(v)$. Node v assigns registers to store $k_{e',e,t}$, and forms the outgoing symbol as

$$y_{e,t} = \sum_{e' \in In(v)} \left(\sum_{i=0}^{t} k_{e',e,i} y_{e',t-i} \right) .$$

That is, the outgoing symbol is a random linear combination of symbols in the node's memory. The $(t+1)$-th term of $f_e(z)$, $f_{e,t}$, is placed in the header of the outgoing message.

2) Decoding: at each time instant t, each sink node r decides whether its global encoding kernel matrix is full rank. If so, it sends an ACK signal to its parent node. An intermediate node v which has received ACKs from all its children at a time t_0 will send an ACK to its parent, and set all subsequent local encoding kernel coefficients $k_{e',e,t}$ to 0 for all $t > t_0$, $e' \in In(v)$, and $e \in Out(v)$. In other words, the constraint length of the local convolutional code increases until it is sufficient for downstream sinks to decode. Such automatic adaptation eliminates the need for estimating the field size or the constraint length a priori. It also allows nodes within the network to operate with different constraint lengths as needed.

Once its global encoding kernel matrix $F_r(z)$ is full rank, a sink node r performs sequential decoding as introduced by Erez et al. [7] to obtain the source message symbol-by-symbol. If $F_r(z)$ is not full rank, r stores received messages and wait for more data to arrive. At time t, the algorithm is considered successful if all sink nodes can decode. At sink r, the local and global encoding kernels are $k_{e',e}(z) = k_{e',e,0} + k_{e',e,1} z + \cdots + k_{e',e,t} z^t$ and $f_e(z) = f_{e,0} + f_{e,1} z + \cdots + f_{e,t} z^t$ respectively, where $k_{d,e,i}$ and $f_{e,i}$ are the encoding coefficients at time i. Sink r can decode successfully if there exists at least m linear independent incoming channels, i.e., the determinant of $F_r(z)$ is a non-zero polynomial. At time t, $F_r(z)$ can be written as $F_r(z) = F_0 + F_1 z + \cdots + F_t z^t$, where F_i is the global encoding kernel matrix at time i. Computing the determinant of $F_r(z)$ at every time instant is complex, so we test instead the following conditions, introduced in [11], [12] to determine decodability at a sink r. The first condition is necessary, while the second is both necessary and sufficient.

1) $rank\begin{pmatrix} F_0 & F_1 & \cdots & F_t \end{pmatrix} = m$
2) $rank(M_t) - rank(M_{t-1}) = m$, where

$$M_i = \begin{pmatrix} F_0 & F_1 & \cdots & F_i \\ 0 & \ddots & \ddots & \vdots \\ 0 & \cdots & F_1 & F_1 \\ 0 & \cdots & 0 & F_0 \end{pmatrix} .$$

Each sink r checks the two conditions in order. If both pass, r sends an ACK signal to its parent; otherwise, it waits for more data to arrive. Observe that as time progresses, $F_r(z)$ grows in size, until decodability is achieved. This scheme for verifying the invertibility of $F_r(z)$ is based on the theory of decodable convolutional codes [12], which transfers the determinant calculation of a polynomial matrix into the rank computation of extended numerical matrices. We do not elaborate on the details and refer interested readers to the original work.

C. Algorithm Statement for Cyclic Networks

In a cyclic network, a sufficient condition for a convolutional code to be successful is that the constant coefficient matrix consisting of all local encoding kernels be nilpotent [11], [13]; this condition is satisfied if we code over an acyclic topology at time 0 [11]. In other words, at time 0, we want to remove a minimal number of edges such that the network becomes acyclic, and to choose $K_v(0)$ randomly for each v

over the resulting acyclic network. This process is essentially the problem of finding the minimal feedback edge set, which is NP-hard [7]. Approximation algorithms with polynomial complexity and other possible heuristic algorithms exist, but we do not give detailed descriptions here owning to the lack of space. The goal is to guarantee that each cycle contains at least a single delay. After initialization, the algorithm proceeds exactly the same as in the acyclic case.

III. Analysis of ARCNC

A. Success probability

Discussions in [2], [3], [13] state that in a network with delays, ANC gives rise to random processes which can be written algebraically in terms of a delay variable z. In other words, a convolutional code can naturally evolve from the message propagation and the linear encoding process. ANC in the delay-free case is therefore equivalent to CNC with constraint length 1. Similarly, using a CNC with constraint length $l > 1$ on a delay-free network is equivalent to performing ANC on the same network, but with $l-1$ self-loops attached to each encoding node. Each self-loop carries $z, z^2, \ldots, z^{l-1}$ units of delay respectively. The ARCNC algorithm we have proposed therefore falls into the framework given by Ho et al. [3], in the sense that the convolution process either arises naturally from cycles with delays, or can be considered as computed over self-loops appended to acyclic networks. From [3], we have the following theorem,

Theorem 3.1: For multicast over a general network with d sinks, the ARCNC algorithm over $\mathbb{F}_q$ can achieve a success probability of at least $(1-d/q^{t+1})^\eta$ at time t, if $q^{t+1} > d$, and η is the number of links with random coefficients.

Proof: At node v, the local encoding kernel $k_{e',e}(z)$ at time t is a polynomial with maximal degree t, i.e., $k_{e',e}(z) = k_{e',e,0} + k_{e',e,1}z + \cdots + k_{e',e,t}z^t$, where $k_{e',e,i}$ is randomly chosen over $\mathbb{F}_q$. If we group the encoding coefficients, the ensuing vector, $k_{e',e} = \{k_{e',e,0}, k_{e',e,1}, \cdots, k_{e',e,t}\}$, is of length $t+1$, and corresponds to a random element over the extension field $\mathbb{F}_{q^{t+1}}$. Using the result in [3], we conclude that the success probability of ARCNC at time t is at least $(1-d/q^{t+1})^\eta$, as long as $q^{t+1} > d$. ■

We could similarly consider the analysis done by Balli et al. [14], which states that the success probability is at least $(1-d/(q-1))^{|J|+1}$, $|J|$ being the number of encoding nodes, to show that a tighter lower bound can be given on the success probability of ARCNC, when $q^{t+1} > d$.

B. Stopping time

We define the stopping time T_i for sink i, $1 \le i \le d$, as the time it takes i to achieve decodability. Also denote by T_N the time it takes for all sinks in the network to successfully decode, i.e., $T_N = \max\{T_1, \ldots, T_d\}$. Then we have:

Corollary 3.2: For any given $0 < \varepsilon < 1$, there exists a $T_0 > 0$ such that for any $t \ge T_0$, ARCNC solves the multicast problem with probability at least $1-\varepsilon$, i.e., $P(T_N > t) < \varepsilon$.

Proof: Let $T_0 = \lceil \lg_q d - \lg_q(1 - \sqrt[\eta]{1-\epsilon}) \rceil - 1$, then $T_0 + 1 \ge \lceil \log_q d \rceil$ since $0 < \varepsilon < 1$, and $(1 - d/q^{T_0+1})^\eta > 1 - \varepsilon$. Applying Theorem 3.1 gives $P(T_N > t) \le P(T_N > T_0) < 1 - (1-d/q^{t+1})^\eta < \varepsilon$ for any $t \ge T_0$. ■

Since $Pr\{\cup_{i=t}^\infty [T_N \le t]\} = 1 - Pr\{\cap_{i=t}^\infty [T_N > t]\} \ge 1 - Pr[T_N > t]$, Corrollary 3.2 shows that as t goes to infinity, ARCNC converges and stops in a finite amount of time with probability one for a multicast connection.

Another relevant measure of the performance of ARCNC is the average stopping time $E[T] = \frac{1}{d}\sum_{i=1}^d T_i$. Observe that $E[T] \le E[T_N]$, where

$$\begin{aligned}
E[T_N] &= \sum_{i=0}^{\infty} tP(T_N = t) \\
&= \sum_{t=1}^{\lceil lg_q d\rceil - 1} P(T_N \ge t) + \sum_{t=\lceil lg_q d\rceil}^{\infty} P(T_N \ge t) \\
&\le \lceil lg_q d\rceil - 1 + \sum_{t=\lceil lg_q d\rceil}^{\infty} [1 - (1 - \frac{d}{q^t})^\eta] \\
&= \lceil lg_q d\rceil - 1 + \sum_{k=1}^{\eta} (-1)^{k-1} \begin{pmatrix} \eta \\ k \end{pmatrix} \frac{d^k}{q^{\lceil lg_q d\rceil k} - 1} .
\end{aligned}$$

When q is large, the summation term becomes $1 - (1-d/q)^\eta$ by the binomial expansion. Hence as q increases, the second term above diminishes to 0, while the first term $\lceil lg_q d\rceil - 1$ is 0. $E[T]$ is therefore upper-bounded by a term converging to 0; it is also lower bounded by 0 because at least one round of random coding is required. Therefore, $E[T]$ converges to 0 as q increases. In other words, if the field size is large enough, ARCNC reduces in effect to RLNC.

Intuitively, the average stopping time of ARCNC depends on the network topology. In RLNC, field size is determined by the worst case node. This process corresponds to having all nodes stop at T_N in ARCNC. ARCNC enables each node to decide locally what is a good constraint length to use, depending on side information from downstream nodes. The corresponding effective field size is therefore expected to be smaller than in RLNC. Two possible consequences of a smaller effective field size are reduced decoding delay, and reduced memory requirements.

C. Complexity

To study the computation complexity of ARCNC, first observe that once the adaptation process terminates, the amount of computation needed for the ensuing code is no more than a regular CNC. In fact, the expected computation complexity is proportional to the average code length of ARCNC. We therefore omit the details of the complexity analysis of regular CNC here and refer interested readers to [7].

For the adaptation process, the encoding operations are described by $f_{e,t} = \sum_{e' \in In(v)} (\sum_{i=0}^t k_{e',e,i} f_{e',t-i})$. If the algorithm stops at time T_N, then the number of operations in the encoding steps is $O(D_{in}|\mathcal{E}|T_N^2 m)$, where D_{in} represents the maximum input degree over all nodes.

To determine decodability at a sink r, we check if the rank of the global encoding matrix $F_r(z)$ is m. A straightforward approach is to check whether the determinant of $F_r(z)$

is a non-zero polynomial. Alternatively, Gaussian elimination can be applied. At t, because $F_r(z)$ is an $m \times |In(r)|$ matrix and each entry is a polynomial with degree t, the complexity of checking if $F_r(z)$ is full rank is $O(D_{in}^2 2^m mt^2)$. Instead of computing the determinant or using Gaussian elimination directly, we propose to check the conditions given in Section II-B. For each sink r, at time t, determining $rank\begin{pmatrix} F_0 & F_1 & \cdots F_t \end{pmatrix}$ requires a computation complexity of $O(D_{in}^2 mt^2)$. If the first test passes, we then need to calculate $rank(M_t)$ and $rank(M_{t-1})$. Observe that $rank(M_{t-1})$ was computed during the last iteration. M_t is a $(t+1)|In(r)| \times (t+1)|In(r)|$ matrix over field $\mathbb{F}_q$. The complexity of calculating $rank(M_t)$ by Gaussian elimination is $O(D_{in}^2 mt^3)$. The process of checking decodability is performed during the adaptation process only, hence the computation complexity here can be amortized over time. In addition, as decoding occurs symbol-by-symbol, the adaptation process itself does not impose any additional delays.

IV. Examples

ARCNC adapts to the topology of general networks by locally increasing the convolutional code length, and generating coefficients randomly. Such adaptation allows nodes to code with different lengths, thus possibly reducing decoding delay and memory overheads associated with overestimating according to the worst case. As examples, next we consider a small combination network to illustrate how ARCNC operates, and how delay and memory overheads can be measured. We also consider a general combination network to show that ARCNC can obtain significant gains in decoding delay here.

A $\binom{n}{m}$ combination network contains a single source s that multicasts m independent messages over $\mathbb{F}_q$ through n intermediate nodes to d sinks [15]; each sink is connected to a distinct set of m intermediate nodes, $d = \binom{n}{m}$. Assuming unit capacity links, the min-cut to each sink is m. In combination networks, routing is insufficient and network coding is needed to achieve the multicast capacity m. Decoding delay at a sink r is defined as the time between the start of the coding process, and when the first symbol is decoded at r.

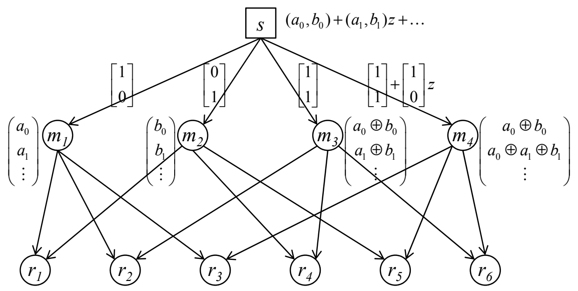
A. A $\binom{4}{2}$ combination network

Fig. 1 illustrates a simple $\binom{4}{2}$ combination network. To see how ARCNC operates, let the messages generated by source s be $\sum_{t=0}^{\infty}(a_t, b_t)z^t$. Assume field size is $q = 2$. Observe that only s is required to code; intermediate nodes relay on received messages directly. At time 0, s chooses randomly the local encoding kernel matrix. Suppose the realization is

$$K_s(z)|_{t=0} = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}.$$

The first 5 sinks can therefore decode directly at time 0, but sink r_6 cannot. Therefore, at time 1, s increases the convolutional code length for the right two intermediate nodes. Suppose the updated local encoding kernel is

$$K_s(z)|_{t=1} = \begin{pmatrix} 1 & 0 & 1 & 1+z \\ 0 & 1 & 1 & 1 \end{pmatrix}.$$



s
$(a_0, b_0) + (a_1, b_1)z + \ldots$
$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$
$\begin{bmatrix} 0 \\ 1 \end{bmatrix}$
$\begin{bmatrix} 1 \\ 1 \end{bmatrix}$
$\begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} z$
$\begin{pmatrix} a_0 \\ a_1 \\ \vdots \end{pmatrix}$
m_1
$\begin{pmatrix} b_0 \\ b_1 \\ \vdots \end{pmatrix}$
m_2
m_3
$\begin{pmatrix} a_0 \oplus b_0 \\ a_1 \oplus b_1 \\ \vdots \end{pmatrix}$
m_4
$\begin{pmatrix} a_0 \oplus b_0 \\ a_0 \oplus a_1 \oplus b_1 \\ \vdots \end{pmatrix}$
r_1
r_2
r_3
r_4
r_5
r_6

Fig. 1. A $\binom{4}{2}$ combination network.

Sink r_6 is now capable of decoding. It therefore acknowledges its parents, which in turn causes s to stop incrementing the corresponding code length. By decoding sequentially, r_6 can recover messages (a_0, b_0) at time 1, (a_1, b_1) at time 2, and (a_{t-1}, b_{t-1}) at time t. Observe that for sinks r_2 to r_5, which are also connected to the two right intermediate nodes, the global encoding kernels increases in length until time 1 as well. In other words, these sinks decode with minimal delay of 0, but require twice the memory when compared to r_1.

In this example, at the single encoding node s, the code lengths used are $(1, 1, 2, 2)$, with an average of $3/2$. At the sinks, the decoding delays are $(0, 0, 0, 0, 0, 1)$, with an average of $1/6$. For the same $\binom{n}{2}$ combination network, the deterministic BNC algorithm given by Xiao et al. [8], designed specifically for combination networks, requires an average decoding delay of 1, since the entire block of data needs to be received before decoding, and the block length is 2. In the next subsection we will show that such gains in decoding delay can be achieved in general combination networks as well. In terms of memory, the BNC algorithm requires 4 bits per node to store data, while ARCNC requires 2 bits at r_1, and 4 bits at all other nodes, with an overall average of $\frac{42}{11}$. Of course, it may possibly need more coding rounds, and the average rounds needed is 2.67. As an example, it implies ARCNC learns the topology of this network automatically, and achieves much lower decoding delays without compromising the amount of memory needed, when compared to the deterministic BNC algorithm designed specifically for combination networks.

Furthermore, if RLNC is used, even for a single sink r_1 to achieve decodability at time 0, the field size needs to be a minimum of 2^3 for the decoding probability to be $\frac{49}{64}$. More coding rounds will be needed to achieve a higher success probability. In other words, with RLNC over $\mathbb{F}_{2^3}$, the average decoding delay will be higher than 1, and the amount of memory needed is at the minimum 6 bits per node. In other words, ARCNC adapts to the network topology at significantly lower costs than RLNC, achieving both lower decoding delays and lower memory overheads, while operating with lower complexities in a smaller field.

B. Decoding delays in a $\binom{n}{m}$ combination network

We now consider a general $\binom{n}{m}$ combination network, and show that the average decoding delay can be significantly improved by ARCNC when compared to the deterministic BNC algorithm. Recall from definitions in Section III-B that the average decoding delay is the average stopping time $E[T] = E\left[\frac{1}{d}\sum_{i=1}^{d} T_i\right] = E[T_i]$. At time $t-1$, for sink node i, if it has not stopped increasing the constraint length of the convolution code, the global encoding kernel is a $m \times m$ matrix of degree $t-1$ polynomials in z over $\mathbb{F}_q$. This matrix has full rank with probability $Q = (q^{tm}-1)(q^{tm}-q^t)\cdots(q^{tm}-q^{t(m-1)})/q^{tm^2}$, so the probability that sink i decodes after time $t-1$ is $P(T_i \geq t) = 1 - Q$. The average stopping time over all sink nodes is then upper bounded by

$$E[T] = E[T_i] = \sum_{t=1}^{\infty} P(T_i \geq t) < \sum_{t=1}^{\infty}(1-(1-\frac{1}{q^t})^m)$$
$$= \sum_{k=1}^{m}(-1)^{k-1}\left(\begin{array}{c} m \\ k \end{array}\right)\frac{1}{q^k-1} \triangleq ET_{UB} \qquad (1)$$

First observe that $E[T]$ a function of m and q, independent of the value of n. In other words, if m is fixed, but n increases, the expected decoding delay does not change. Next observe that if q is large, ET_{UB} becomes 0, consistent with the general analysis in Section III-B.

A similar upper bound can be found for the variance of T as well. It can be shown that

$$var(T) = \frac{E[T_i^2]}{d} + \frac{1}{d^2}\left[\sum_{i=1}^{d}\sum_{j\neq i} E[T_iT_j]\right] - E^2[T_i]$$
$$< \frac{ET_{UB}^2}{d} + \frac{m}{n}\rho_{UB} - (1+\frac{m}{n})(ET_{LB})^2\,, \qquad (2)$$

where ET_{UB}^2 is an upperbound for $E[T_i^2]$, ρ_{UB} is an upperbound for $E[T_iT_j]_{i\neq j}$, and ET_{LB} is a lowerbound for $E[T_i]$. All three quantities are functions of m and q, independent of n. If m and q are fixed, as n increases, d also increases, and $var(T)$ diminishes to 0. Combining this result with a bounded expectation, what we can conclude is that even if more intermediate nodes are added, a large proportion of the sink nodes can still be decoded within a small number of coding rounds. On the other hand, if m and n are comparable in scale, for example, if $m = n/2$, then the bound above depends on the exact value of ET_{UB}^2, ρ_{UB} and ET_{UB}. We leave the detailed analysis of this case for journal version.

Comparing with the deterministic BNC algorithm proposed by Xiao et al. [8], we can see that for a large combination network, with fixed q and m, ARCNC achieves much lower decoding delay. In the BNC scheme, the block length is required to be $p \geq n-m$ at the minimum. Thus the decoding delay increases at least linearly with n. Similar comparisons can be made with RLNC, and it is not hard to see that we can obtain gains in both decoding delay and memory.

So far we have used $\binom{n}{m}$ combination networks explicitly as an example to illustrate the operations and the decoding delay gains of ARCNC. It is important to note, however, that this is a very special network, in which only the source node is required to code, and each sink shares at least 1 parent with other $\binom{n-1}{m-1}$ sinks. If sink r cannot decode, all other $\binom{n-1}{m-1}$ sinks sharing parents with r are required to increase their memory capacity. Therefore, in combination networks, we do not see much gains in terms of memory overheads when compared with BNC algorithms. In more general networks, however, when sinks do not share ancestors with as many other sinks, ARCNC can achieve gains in terms memory overheads as well, in addition to decoding delay. Due to space limitations, we do not give any detailed analysis, but it can be shown, for example, that in an umbrella-shaped network, memory overheads can be significantly reduced with ARCNC when compared to other network codes.

V. Conclusion

We propose an adaptive random convolutional network code (ARCNC), which operates in a small field, and locally and automatically adapts to the network topology by incrementally growing the constraint length of the convolution process. We show through analysis that ARCNC performs no worse than random algebraic linear network codes, and illustrate through a combination network example that it can reduce the average decoding delay significantly. ARCNC can also reduce memory overheads in networks where sinks do not share the majority of their ancestors with other sinks. One possible future direction of analysis is to characterize the behavior of this algorithm over random networks, the results of which will enable us to decide on the applicability of ARCNC to practical systems.

References

[1] R. Ahlswede, N. Cai, S. Li, and R. Yeung, "Network information flow," *IEEE Trans. on Inform. Theory*, vol. 46, no. 4, pp. 1204–1216, 2002.

[2] S. Li, R. Yeung, and N. Cai, "Linear network coding," *IEEE Trans. on Inform. Theory*, vol. 49, no. 2, pp. 371–381, 2003.

[3] T. Ho, M. Médard, R. Koetter, D. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Trans. on Inform. Theory*, vol. 52, no. 10, pp. 4413–4430, 2006.

[4] M. Médard, M. Effros, D. Karger, and T. Ho, "On coding for non-multicast networks," in *Proc. of the 41st Allerton Conference*, vol. 41, no. 1, 2003, pp. 21–29.

[5] 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. on Inform. Theory*, vol. 51, no. 6, pp. 1973–1982, 2005.

[6] S. Li and R. Yeung, "On convolutional network coding," in *Proc. of IEEE Int. Sym. on Info. Theory*, 2006, pp. 1743–1747.

[7] E. Erez and M. Feder, "Convolutional network codes," in *Proc. of IEEE Int. Sym. on Info. Theory (ISIT)*, 2005, p. 146.

[8] M. Xiao, M. Médard, and T. Aulin, "A binary coding approach for combination networks and general erasure networks," in *Proc. of IEEE Int. Sym. on Info. Theory (ISIT)*, 2008, pp. 786–790.

[9] M. Kim, C. Ahn, M. Médard, and M. Effros, "On minimizing network coding resources: An evolutionary approach," in *Proc. NetCod*, 2006.

[10] S. Jaggi, M. Effros, T. Ho, and M. Medard, "On linear network coding," in *Proc. of the 42nd Allerton Conference*, 2004.

[11] N. Cai and W. Guo, "The conditions to determine convolutional network coding on matrix representation," in *Proc. NetCod*, 2009, pp. 24–29.

[12] J. MasseyY and M. Sain, "Inverses of linear sequential circuits," *IEEE Trans. on Comp.*, vol. 100, no. 4, pp. 330–337, 1968.

[13] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Trans. on Networking*, vol. 11, no. 5, pp. 782–795, 2003.

[14] H. Balli, X. Yan, and Z. Zhang, "On randomized linear network codes and their error correction capabilities," *IEEE Trans. on Inform. Theory*, vol. 55, no. 7, pp. 3148–3160, 2009.

[15] R. Yeung, S. Li, N. Cai, and Z. Zhang, "Network Coding Theory: Single Sources," *Foundations and Trends® in Communications and Information Theory*, vol. 2, no. 4, pp. 241–329, 2005.