A Connection between Network Coding and Convolutional Codes

Christina Fragouli, Emina Soljanin

christina.fragouli@epfl.ch, emina@lucent.com

Abstract

The min-cut, max-flow theorem states that a source node can send a commodity through a network to a sink node at the rate determined by the flow of the min-cut separating the source and the sink. Recently it has been shown that by liner re-encoding at nodes in communications networks, the min-cut rate can be also achieved in multicasting to several sinks. In this paper we discuss connections between such coding schemes and convolutional codes. We propose a method to simplify the convolutional encoder design that is based on a subtree decomposition of the network line graph, describe the structure of the associated matrices, investigate methods to reduce decoding complexity and discuss possible binary implementation.

I. Introduction

Communication networks are, like their transportation counterparts, mathematically represented as directed graphs $G = (V, E)$. We are concerned with multicast communications networks in which h unit rate information sources $S_1, \ldots, S_h$ simultaneously transmit information to N receivers $R_1, \ldots, R_N$ located at N distinct nodes. The number of edges of the min-cut between a source and each receiver node is h. The famous max-fbw, min-cut theorem states that a single source S_1 can send a commodity through a network to the receiver R_1 at the rate determined by the fbw of the min-cut separating S_1 and R_1. Recently it has been realized that nodes in communication networks can not only re-route but also re-encode the information they

receive, and shown that by liner re-encoding, the min-cut rate can be also achieved in multicasting to several sinks [1], [7].

Multicast at rate h can be achieved by linear coding, as was shown constructively by Li, Yueng, and Cai in [7]. A fast implementation of their approach was developed by Sanders, Egner, and Tolhuizen in [9], resulting in polynomial time algorithms for constructing linear codes for multicast. An algebraic framework for network coding was developed by Koetter and Medard in [5], who translated the network code design to an algebraic problem by deriving transfer matrices of size determined by the underlying graph. By applying tools from algebraic geometry, they were able to answer questions related to achievable rates for linear codes over communication graphs.

In this paper we present an alternative simpler derivation of the transfer function result in [5] by observing the connection with convolutional codes over finite fields. We then proceed to expand this observation in different directions.

Most of the previous work [7], [5], [9] assumes zero delay meaning that all nodes simultaneously receive all their inputs and produce their outputs. The convolutional code framework naturally takes delay into account, but at the cost of increased complexity for both encoding and decoding. We investigate different methods to reduce the complexity requirements.

We first describe the subtree decomposition which we proposed to a different end in [4]. The subtree decomposition allows to partition all the possible multicast configurations into equivalence classes, where two topologies are called equivalent if they employ the same network code. Grouping together different configuration allowed us to derive decentralized network coding algorithms, compute alphabet size bounds, increase the scalability of the employed code and facilitate design choices [4]. Employing the subtree decomposition in the context of convolutional codes, as we propose in this paper, can also serve multiple purposes. It significantly reduces the dimensionality of the associated matrices, thus making faster all algorithms that employ matrices. It allows to group together different configurations, and thus design a convolutional code that can be applied to all networks that share the same subtree graph. It facilitates the addition of new users, as the

structure of the convolutional code does not need to change as long as the structure of the subtree graph is preserved.

We further propose to reduce the decoding complexity individually for every receiver, by calculating the minimal equivalent encoder to the encoder that each receiver experiences. Additionally, we discuss implementation using binary encoders, and propose a simplified version of the method in [7] to deal with cycles in the network. Throughout the paper different examples illustrate the discussion. Independently, the authors in [3] propose to add node-memory to increase the alphabet size a network can support, which has a similar flavor to our proposed binary encoders implementation.

The paper is organized as follows. Section II introduces the notation. Section III establishes the connection with convolutional codes and discusses the subtree decomposition. Section IV describes the structure of the associated matrices. Section V discusses decoding complexity. Section VI investigates binary implemention, and finally Section VII concludes the paper.

II. Background and Notation

We consider a communications network represented by a directed graph $G = (V, E)$ with unit capacity lossless edges. There are h unit rate information sources $S_1, \ldots, S_h$ and N receivers $R_1, \ldots, R_N$. The number of edges of the min-cut between a source and each receiver node is h. The h sources multicast information simultaneously to all N receivers at rate h. We denote by (S_i, R_j) a path from source i to receiver j . The min-cut condition implies that for a given receiver R_j, there exist h paths (S_i, R_j), $1 \leq i \leq h$, that are edge disjoint.

For linear network coding, each node of G receives a set of inputs from some finite field $\mathbb{F}$ to be linearly combined over the same field before forwarding through output edges. Consequently, through each edge of G flows a linear combination of source symbols. Let $\sigma_i \in \mathbb{F}$ be a symbol corresponding to the source S_i for

$1 \leq i \leq h$. Then the symbol fbwing through some edge e of G is given by

$$\alpha_1(e)\sigma_1 + \cdots + \alpha_h(e)\sigma_h = \left[\alpha_1(e) \ldots \alpha_h(e)\right] \underbrace{\begin{bmatrix} \sigma_1 \\ \vdots \\ \sigma_h \end{bmatrix}}_{\mathbf{S}},$$

where vector $[\alpha_1(e) \ldots \alpha_h(e)]$ belongs to an h dimensional vector space over $\mathbb{F}$. We shall refer to this vector as the *coding vector* of edge e.

If we assume zero delay, meaning that all nodes simultaneously receive all their inputs and produce their output, then the coding vectors associated with input edges of a receiver node define a system of linear equations, that the receiver needs to solve to determine the source symbols. More specifically, consider receiver R_j. Let ρ_i^j be the symbol on the last edge of the path (S_i, R_j), and $\mathbf{A}_j$ the matrix whose i-th row is the coding vector of the last edge on the path (S_i, R_j). Then the receiver j has to solve the system of linear equations

$$[\rho_1^j \ldots \rho_h^j]^T = \mathbf{A}_j \mathbf{S}, \tag{1}$$

to retrieve the information symbols $\mathbf{S} = [\sigma_1 \ \sigma_2 \ \ldots \sigma_h]^T$ transmitted from the h sources.

III. Connection with Convolutional Codes

Since for a network code we eventually have to describe which operations each node in G has to perform on its inputs for each of its outgoing edges, we find it more transparent to work with the line graph of G, namely the graph with vertex set $E(G)$ in which two vertices are joined if and only if they are adjacent as edges in G [2]. In the line graph a coding vector is associated with every node, and each node contains a linear combination of the coding vectors of its parent nodes. Each receiver observes the contents of h such nodes.

To relax the zero delay assumption we can associate a unit delay $\mathcal{D}$ with each node of the line graph.

Associating a unit delay with each edge of the network was also proposed in [5]. Our contribution is the observation that then the line graph can be thought of as a convolutional code over a finite field, with number of memory elements m equal to the number of edges in G.

A general description of a convolutional encoder over a finite field with h inputs, N outputs, and m memory elements is given by the well known state-space equations:

$$\mathbf{s}_{j+1} = \mathbf{A}\mathbf{s}_j + \mathbf{B}\mathbf{u}_j$$

$$\mathbf{y}_j = \mathbf{C}\mathbf{s}_j + \mathbf{D}\mathbf{u}_j$$

where $\mathbf{s}_j$ is the $m \times 1$ state vector, $\mathbf{y}_j$ is the $Nh \times 1$ output vector, $\mathbf{u}_j$ is the $h \times 1$ input vector, and $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$ are matrices with appropriate dimensions. The corresponding generator matrix $\mathbf{G}(\mathcal{D})$ is given by

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

where $\mathcal{D}$ is the indeterminate delay operator. The expression in Eq. (2) coincides with the transfer matrix $\mathbf{M}$ derived in [5], giving a different and simpler derivation of the same result.

Matrix $\mathbf{A}$ reflects the way the memory elements are connected. An element in matrix $\mathbf{A}$ can be nonzero, only if a corresponding edge exists at the given network configuration. Network code design amounts to selecting the nonzero-element values for matrix $\mathbf{A}$. Matrices $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$ are completely determined by the network configuration.

Before examining the structure of matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$ in more detail (Section IV), we observe that their size depends upon the number of memory elements of the convolutional code, which in turns is equal to the number of edges in the original graph G. This number can get quite large, resulting in large size of matrices to handle. Thus following (Sec III-A) we introduce the subtree decomposition which allows to significantly decrease the number of memory elements, and thus the involved dimensionality.

A. Decomposition into Subtrees

The basic idea is that we can group together the parts of the line graph through which the same information flows [4]. Throughout this discussion we use the example in Fig. 1, which is an example of a network topology with two sources multicasting to the same set of three receivers. The associated line graph is

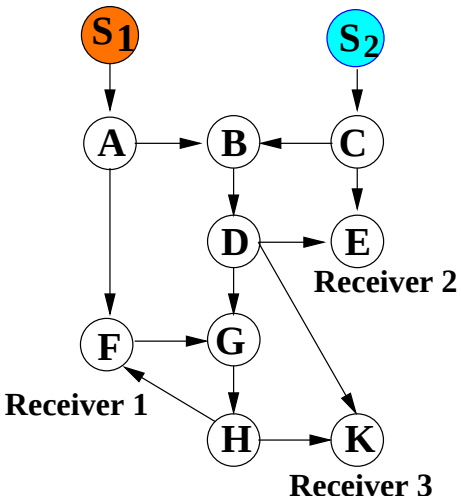

Fig. 1. Topology with two sources $\{S_1, S_2\}$ and three receivers $\{F, E, K\}$.

depicted in Fig. 2.

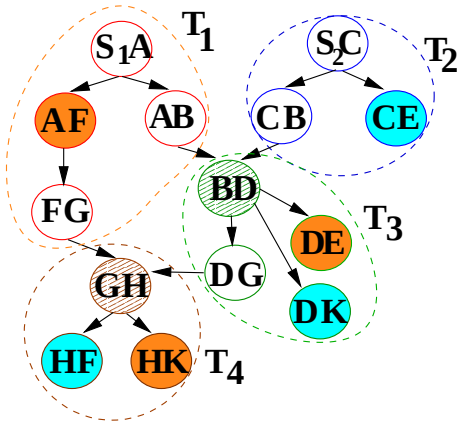

Fig. 2. Line graph that illustrates the coding points BD and GH and the subtree decomposition.

Without loss of generality, by possibly introducing auxilliary nodes, we may assume that the line graph contains a node corresponding to each of the h sources. We refer to these nodes as *source nodes*. Each of the remaining nodes performs at most one coding operation (linear combining) on its inputs, and forwards the result to all of its outputs. We shall refer to the nodes which perform coding operations as *coding points*. A coding point computes a linear combination of its inputs, and forwards the resulting symbol to all of

its children. The children forward the symbol to their children up to the next coding point, which again computes a linear combination of its inputs.

We partition the line graph into a disjoint union of subsets T_i so that the following properties hold:

1) Each subset T_i contains exactly one source node or coding point.

2) Every other node belongs to the subset T_i containing its first ancestral coding point or source node.

Note that two nodes of the line graph belong to the same subset T_i if and only if the same linear combination of source symbols flows through them. Also, each T_i is a tree. We shall call subset T_i a *source subtree* if it starts with a source node or a *coding subtree* if it starts with a coding point. Fig. 2 shows the four subtrees $\{T_1, T_2, T_3, T_4\}$ of the network in Fig. 1; T_1 and T_2 are source subtrees.

For the network code design problem, the structure of the network inside a subtree does not play any role. The only information we need to retain from the internal structure of a subtree is which receiver nodes are contained in it. Thus we can contract each subtree to a node and retain only the edges that connect the subtrees, to get the *subtree graph*.

The subtree graph for our example network of Fig. 1 is shown in Fig. 3. The nodes corresponding to

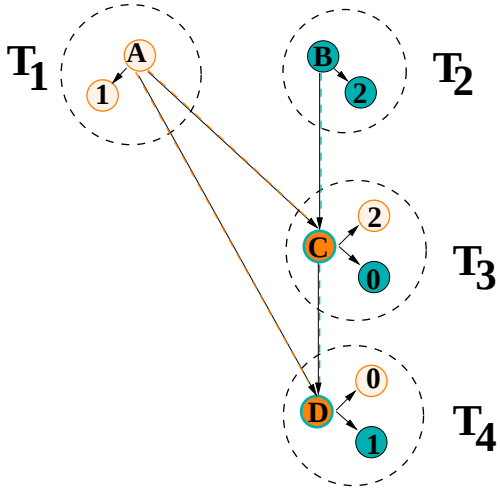

Fig. 3. Subtree Graph.

source-receiver pairs inside each subtree are represented pictorially in Fig. 3.

B. Subtree Graph Convolutional Code

Since each edge within a subtree is in the same state, we can associate a unit delay $\mathcal{D}$ with each subtree and neglect the delay within the subtree. That is, we can consider the subtree configuration as a convolutional code where each subtree is a single memory element. An example subtree configuration is shown in Fig. 6 and its corresponding convolutional code in Fig. 7.

Employing the subtree graph serves multiple purposes. It significantly reduces the dimensionality of the associated matrices, thus making faster all algorithms that employ matrices. It allows to group together different configurations, and thus design a convolutional code that can be applied to all networks that share the same subtree graph. It actually allows to enumerate all the possible configurations for a small number of sources/receivers [4] and thus possibly store precompute codes for frequently observed cases. It facilitates the addition of new users, as the structure of the convolutional code does not need to change as long as the structure of the subtree graph is preserved.

Unless otherwise stated, we will following be considering the convolutional code associated with the subtree graph.

IV. Structural Properties

We next examine the structure of matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$. We distinguish two cases, depending on whether a partial order constraint, which we will following describe, is satisfied.

Partial order constraint

We refer to the line graph of the union $\bigcup_{j=1}^{N}(S_i, R_j)$ of paths from source S_i to all receivers as the transmission structure corresponding to source i. Each transmission structure induces a partial order on the set of the line graph nodes: if edge a is a child of edge b then we say that $a < b$. The source node is the maximal element.

Each different transmission structure imposes a different partial order on the same set of edges. We distinguish the graphs depending on whether the partial order imposed by the different transmission structures

is consistent. Consistency implies that for all pairs of edges a and b, if $a < b$ in some transmission structure, there does not exist a transmission structure where $b < a$. A sufficient, but not necessary, condition is that the underlying graph G is acyclic. For example, consider Fig. 4, that depicts a subgraph of a network graph. Sources S_1 and S_2 use the cycle $ABCD$ to transmit information to receivers R_1 and R_2, respectively. In the case A, the transmission structures for sources S_1 and S_2 impose consistent partial orders on the edges of the cycle. In the case B, for the transmission structure of source S_1, we have $AB > CD$, whereas for the transmission structure of source S_2, we have $CD > AB$.

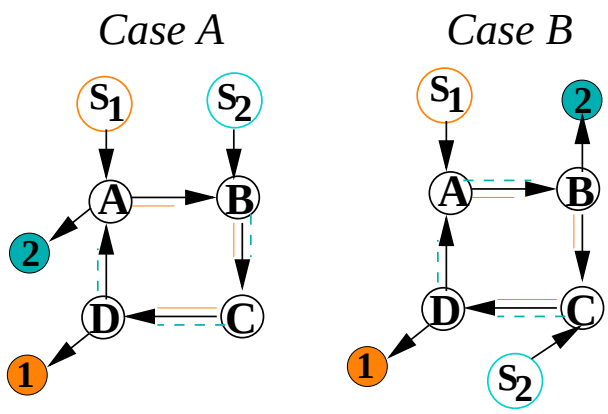

Fig. 4. Case A: partial order preserved. Case B: partial order not preserved.

A. Case 1: Consistent Partial Order

Each subtree corresponds to one element in the state vector $\mathbf{s}$. We group the subtrees into classes, according to their maximum distance from the source subtrees, where the distance is defined as the maximum number of subtrees in a path from the source subtrees. Let d_i be the number of subtrees (memory-elements) at distance i, then $\sum_i d_i = m$, where m is the total number of subtrees. We order the elements in $\mathbf{s}$ according to the distance of the corresponding states from the source subtrees. Thus the h source-subtrees correspond to the first h elements of the state vector $\mathbf{s}$. [1]

Since a state at time $j+1$ can depend only on the states at time j that are closer to the source, it is easy to

[1] We could have selected not to have memory elements associated with the source subtrees. However we believe that including memory elements for the source subtrees makes the presentation clearer, and it does not end up incurring any additional complexity in either the design or the decoding.

see that with this ordering of states within the vector $\mathbf{s}$ the $m \times m$ matrix $\mathbf{A}$ has the following form

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{A}_{1,1} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} & \mathbf{0} & \cdots & \\ & & & & \\ \mathbf{A}_{d-1,1} & \mathbf{A}_{d-1,2} & \cdots & \mathbf{A}_{d-1,d-1} & \mathbf{0} \end{bmatrix}, \tag{3}$$

where each block matrix element (i, j) has dimension $d_i \times d_j$ and $d = \max_i\{d_i\}$. Obviously $\mathbf{A}^d = 0$, that is, we always have a feedforward encoder of effective memory d. Since only the h source subtrees directly depen on the input, the $m \times h$ matrix $\mathbf{B}$ has the form

$$\mathbf{B} = \begin{bmatrix} \mathbf{I} \\ \mathbf{0} \end{bmatrix}, \tag{4}$$

where $\mathbf{I}$ is the $h \times h$ identity matrix. The $Nh \times m$ matrix $\mathbf{C}$ is a zero-one matrix of the form

$$\mathbf{C} = \begin{bmatrix} \mathbf{C}_1 \\ \vdots \\ \mathbf{C}_h \end{bmatrix}, \tag{5}$$

where the $h \times m$ matrix $\mathbf{C}_i$ corresponds to the receiver i. Each row of $\mathbf{C}_i$ corresponds to one of the subtrees whose state is observed by the receiver i. Thus matrix $\mathbf{C}_i$ has exactly one 1 in each row and at most one 1 in each column. Matrix $\mathbf{D}$ is identically zero since we associate with source subtrees. Matrices $\mathbf{C}_i$ are completely determined by the subtree configuration.

The min-cut, max-fbw requirement is equivalent to the condition that the $h \times h$ transfer matrix that

corresponds to each receiver

$$\mathbf{G}_i(\mathcal{D}) = \mathbf{C}_i(\mathcal{D}^{-1}\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}, \quad i = 1 \ldots R, \tag{6}$$

has full rank, as described in [5].

B. Case 2: Non-Consistent Partial Order

In some networks with cycles, the partial order imposed by the different transmission structures is not consistent. The corresponding subtree graph form recursive convolutional encoders, and can be analyzed by taking into account the feedback as proposed in [5].

Alternatively, we may follow a simplified version of the approach in [7]. Observe that an information source may need to be transmitted through the edges of a cycle at most once, and then can be removed from the circulation by the node that introduced it. For example consider the cycle in Fig. 4-B, and assume that each edge corresponds to one memory element. Then the flows through the edges of the cycle are

$$\begin{aligned} AB: &\quad \sigma_1(k) + \sigma_2(k-2) \\ BC: &\quad \sigma_1(k-1) + \sigma_2(k-3) \\ CD: &\quad \sigma_1(k-2) + \sigma_2(k) \\ DA: &\quad \sigma_1(k-3) + \sigma_2(k-1). \end{aligned} \tag{7}$$

where $\sigma_i(k)$ is the symbol transmitted from source S_i at time k. Equation (7) can be easily implemented by employing a block of memory elements as shown in Fig. 5. Thus, we can still have a feedforward encoder, by representing the cycle with a block of memory elements in the subtree graph, and accordingly altering the structure of matrices $\mathbf{A}$ and $\mathbf{C}$.

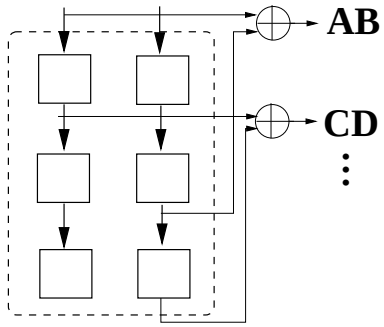

Fig. 5. Block representation of a cycle.

V. Decoding Complexity

Taking delay into account implies that each receiver no longer has a linear system of equations to solve (Eq. 1), but needs to perform trellis decoding (Eq. 8). However, the task of decoding is very much simplified by the fact that there is no noise.

Receiver i experiences a rate-1 noncatastrophic encoder with generator matrix

$$\mathbf{G}_i(\mathcal{D}) = \mathcal{D}\mathbf{C_i}(\mathbf{I} - \mathcal{D}\mathbf{A})^{-1}\mathbf{B} \tag{8}$$

Assume that at time j the receiver sees output $\mathbf{y}_j$. To decode, the receiver only needs to additionally know the previous state $\mathbf{s}_{j-1}$ at time $j-1$, and the trellis diagram of the convolutional code. In other words, the traceback depth of decoding is one. Thus the complexity of decoding is proportional to the complexity of the trellis diagram.

Two methods to reduce the complexity that a receiver experiences are the following.

- *Method 1:* Minimize the trellis diagram used for encoding.
- *Method 2:* Identify the minimal strictly equivalent encoder to $\mathbf{G}_i(\mathcal{D})$ to decode, to minimize the trellis diagram used for decoding.

1) Method 1: We are interested in identifying, among all encoders that are subject to the constraints of a given topology, and that satisfy the min-cut max-fbw conditions for each receiver, the encoder that has the smallest number of memory elements. The minimization does not need to preserve the same set of outputs,

as we are not interested in error-correcting properties, but only the min-cut property for each receiver.

A significant step towards this direction was already achieved with the subtree decomposition. Another step in this direction is to identify the minimal subtree graph with the property that the min-cut between the source and the destination nodes is equal to h. Equivalently, identify a subgraph of the subtree graph such that removing any edge would result in violating the min-cut condition for at least one of the destination nodes. This can be achieved algorithmically by removing the edges that are not necessary for the mincut condition. After this procedure, subtrees that have one parent can be incorporated with the parent, reducing the number of required memory elements.

2) Method 2: The information to be received by receiver R_i will be encoded by the encoder $\mathbf{G}_i(\mathcal{D})$. This encoder cannot be chosen arbitrarily, but subject to configuration restrictions.

However, to decode, we may use a different trellis, the one associated with the minimal strictly equivalent encoder to $\mathbf{G}_i(\mathcal{D})$. Two codes are called *strictly equivalent* if they have the same mapping of input sequences to output sequences. Among all strictly equivalent encoders, that produce the same mapping of input to output sequences, the encoder that uses the smallest number of memory elements is called *minimal*. Strict equivalence minimality coincides with the definition of minimality in control theory [8], for systems described by discrete state-space equations. The associated minimality criteria are that of observability and controllability.

So although we can not select the minimal $\mathbf{G}_i(\mathcal{D})$ to encode because we were restricted by the configuration, at the decoder we may still use the minimal strictly equivalent trellis to reduce decoding complexity.

VI. Binary Alphabet

The convolutional codes corresponding to network codes are over a finite field $\mathbb{F}$ that (but for the simplest cases) is not binary. If a network supports only binary transmission, we consider $f = \lceil \log_2 |\mathbb{F}| \rceil$ uses of the network to comprise a symbol of a higher alphabet. This implies that each node that performs network coding has to store and process f binary bits before retransmitting, thus effectively needs to use f binary memories.

An alternative approach would be to restrict the network coding alphabet to be binary, and allow each node to use up to f binary memory elements in an arbitrary fashion. In our subtree configuration, we may replace each subtree with any convolutional code with f binary memory elements. Using an alphabet of size 2^f becomes a special case, so it is guaranteed that there exists a topology that employs possibly less and at most an equal number of binary memory elements. An example is provided following.

A. Numerical Example

Consider the configuration with $h = 2$ source subtrees and $m - h = 2$ leaf subtrees depicted in Fig. 6. The corresponding convolutional encoder is depicted in Fig. 7. Matrix $\mathbf{A}$ will have the form

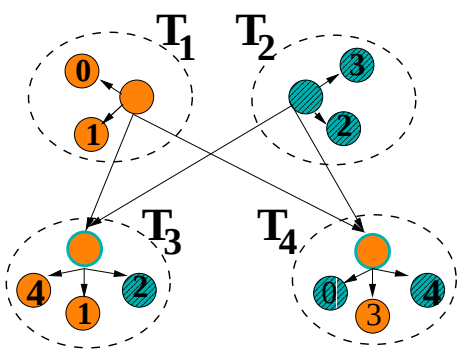

Fig. 6. Subtree configuration with two sources and five receivers.

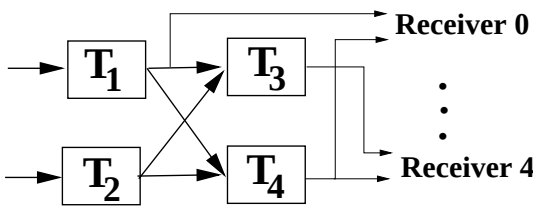

Fig. 7. Convolutional encoder corresponding to the subtree configuration with two sources and five receivers.

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{A}_{1,1} & \mathbf{0} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ * & * & 0 & 0 \\ * & * & 0 & 0 \end{bmatrix},$$

where $\mathbf{A_{1,1}}$ is of dimension 2×2, and "$*$" denotes a nonzero element. The generator matrix of dimension 2×2 corresponding to every receiver will have the form

$$\mathbf{G}_i(\mathcal{D}) = \mathcal{D}\mathbf{C_i}(\mathbf{I} + \mathcal{D}\mathbf{A})\mathbf{B} = \mathcal{D}\mathbf{C_i} \underbrace{\begin{bmatrix} \mathbf{I} \\ \\ \mathcal{D}\mathbf{A_{1,1}} \end{bmatrix}}_{\mathbf{G}}.$$

The identity matrices $\mathbf{I}$ are of different dimension, as determined by the context. Matrix $\mathbf{G}$ is common for all receivers. A possible choice for matrix A over a finite field of size greater or equal to three would be

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 2 & 0 & 0 \end{bmatrix}.$$

Alternatively, we may use a binary network code. Since we consider two uses of the network, the input/output to each subtree T_i in Fig. 7 would be two bits. Then each subtree can perform at time k the following binary operation

$$\mathbf{M}_i \cdot [\sigma_1(k) \;\; \sigma_1(k-1) \;\; \sigma_2(k) \;\; \sigma_2(k-1)]^T,$$

where

$$\mathbf{M_1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \quad \mathbf{M_2} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$
$$\mathbf{M_3} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}, \quad \mathbf{M_4} = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix}.$$

Receiver k will observe a matrix of the form

$$\mathbf{A}_k = \begin{bmatrix} \mathbf{M_i} \\ \mathbf{M_j} \end{bmatrix}, \quad i \neq j,$$

which has full rank.

VII. Conclusions

We investigated connections between network coding and convolutional codes, based on our subtree decomposition method recently proposed in [4]. We provided an alternative simpler derivation of the transfer function result in [5] using the state-space description of convolutional codes, and the subtree decomposition method to reduce the dimensions of the associated matrices. We analyzed the structure of the matrices under a partial order assumption. We also discussed reduced complexity decoding methods, and use of binary alphabet.

References

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

[2] R. Diestel, "Graph Theory" *Second Edition* Springer 2000.

[3] M. Feder, D. Ron, and A. Tavory, "Bounds on Linear Codes for Network Multicast", *Electronic Colloquium on Computational Complexity*, Report No. 33, 2003.

[4] C. Fragouli and E. Soljanin, "Network Coding Based on Subtree Decomposition", *DIMACS Technical Report, Rutgers University*, 2003.

[5] R. Koetter and M. Medard, "Beyond routing: an algebraic approach to network coding," *Proc. IEEE INFOCOM 2002*, New York, NY, 23-27 June 2002, vol 1, pp. 122–130.

[6] R. Lidl and H. Niederreiter, *Finite Fields*, New York: Cambridge Univ. Press, 1997.

[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] H.-A. Loeliger, G. D. Forney, T. Mittelholzer and M. D. Trott, "Minimality and observability of group systems", *Linear Algebra And Its Applications*, July 1994, vol. 205-206, pp. 937-963.

[9] P. Sanders, S. Egner, and L. Tolhuizen, "Polynomial Time Algorithms For Network Information Flow," *Proc. 15th ACM Symposium on Parallel Algorithms and Architectures*, 2003.