A recursive coding algorithm for two-unicast-Z networks

The MIT Faculty has made this article openly available. Please share how this access benefits you. Your story matters.
A Recursive Coding Algorithm for Two-unicast-Z Networks

Weifei Zeng  
Research Laboratory of Electronics,  
Massachusetts Institute of Technology  
weifei@mit.edu

Viveck Cadambe  
Research Laboratory of Electronics,  
Massachusetts Institute of Technology  
viveck@mit.edu

Muriel Médard  
Research Laboratory of Electronics,  
Massachusetts Institute of Technology  
medard@mit.edu

Abstract—We derive a new linear network coding algorithm for two-unicast-Z networks over directed acyclic graphs, that is, for two-unicast networks where one destination has apriori information of the interfering source message. Our algorithm discovers linear network codes for two-unicast-Z networks by combining ideas of random linear network coding and interference neutralization. We show that our algorithm outputs an optimal network code for networks where there is only one edge emanating from each of the two sources. The complexity of our algorithm is polynomial in the number of edges of the graph.

I. INTRODUCTION

Characterizing the capacity region of networks of orthogonal capacitated links, often termed the network coding capacity, is of central interest for researchers in network information theory. In the special case of multicast where there is a single message in the network, and for certain scenarios where the message sources or destinations are collocated, the capacity region of the network can be characterized in terms of the edge cut values of the graph (see [1] for example). For these cases, random linear network coding provides elegant, optimal coding solutions. However, the capacity region of even the two-unicast network, which has two distributed sources and two distributed destinations, remains unknown.

The classical techniques of routing and random linear network coding are known to be insufficient in general in finding optimal network coding solutions. In general, wireline networks require linear network codes whose coding coefficients depend on the structure of the network. While the connections to multivariate polynomial solvability [1] lead to algorithms for determining the feasibility of linear network codes, they are seldom used because of their enormous computational complexity. In fact, the task of determining solvability and solutions for specific networks is often done by trial and error. Recently, the ideas of interference alignment and interference neutralization, which originated in the study of wireless networks, have been identified to be important ingredients in developing linear network coding solutions even for wireline networks [2]–[4]. However, there exists few systematic approaches to developing linear network coding solutions that incorporate these ingredients. In the paper, we make progress towards this goal through the development of a systematic linear network coding algorithm for a special class of two-unicast networks known as the two-unicast-Z networks. Two-unicast-Z networks are two unicast networks where one destination is aware of the interfering message source apriori [3] (see Figure 1). While linear network coding has been shown to be sub-optimal for network coding capacity [5] in general, we note that the question of sufficiency of linear network coding for two-unicast-Z networks remains open.

A. Summary of Contributions

We develop a new linear network coding algorithm based on the intuition that maximizing the linear coding capacity of the two-unicast-Z network requires an approach that devises transfer matrices that maximizes the capacity of the implied end-to-end Z-interference channel [6]. For the special case where the min-cut between each source and its respective destination is bigger than or equal to 1, we recover the result of [7] which determines feasibility conditions for achievability of the rate pair (1, 1). That is, when the generalized network sharing (GNS) bound is bigger than or equal to 2, our linear network coding algorithm achieves a rate tuple that is at least as large as (1, 1). In addition, we show that the complexity of our algorithm to be polynomial in terms in of the number of edges for all graphs, and linear in the number of vertices for classes of graphs whose degree is bounded by some parameter.

II. SYSTEM MODEL

Consider a directed acyclic graph (DAG) \( G = (V, E) \), where \( V \) denotes the set of vertices and \( E \) denotes the set of edges. We allow multiple edges between vertices, hence, \( E \subset V \times V \times \mathbb{Z}_+ \), where \( \mathbb{Z}_+ \) denotes the set of positive integers. For an edge \( e = (u, v, i) \in E \), we denote \( \text{Head}(e) = v \) and \( \text{Tail}(e) = u \). For a given vertex \( v \in V \), we denote \( \text{In}(v) = \{ e \in E : \text{Head}(e) = v \} \) and \( \text{Out}(v) = \{ e \in E : \text{Tail}(e) = v \} \).

Since the graph is DAG, there exists some partial ordering of the vertices of \( G \). Let \( R \) be a total order extended from such a vertex partial order. Denote \( \text{Ord}_R(u) \) to be the order of vertex \( u \in V \) with respect to \( R \). For every edge \( (u, v, i) \in E \), we have \( \text{Ord}_R(u) < \text{Ord}_R(v) \). The subscript \( R \) is dropped from \( \text{Ord}_R \) when there is no ambiguity. We define a partial order of the edges \( E \) such that the order of an edge is equal to the order of the tail node of the edge in \( R \), i.e., \( \text{Ord}(e) = \text{Ord}(\text{Tail}(e)), e \in E \). Note that all edges sharing the same tail node have the same order.

For the graph \( G \), the set \( S = \{ s_1, s_2 \} \), where \( s_1, s_2 \in V \), is the set of two sources in the network. In this paper, we define each destination as a set of edges. Specifically, we denote \( T = \{ T_1, T_2 \} \) as the set of two destinations, where \( T_1 \) and \( T_2 \) each is a set of edges, i.e., \( T_i \subset E, i = 1, 2 \). Note that in general, we allow an edge to belong both destinations, i.e., \( T_1 \cap T_2 \neq \emptyset \). Furthermore, the head vertices for edges in the same destination may not have to be the same, i.e., \( \text{Head}(e_1) \neq \text{Head}(e_2) \) for \( e_1, e_2 \in T_i, i = 1, 2 \) is allowed.

The two-unicast-Z network model is shown in Figure 1 where the sources \( s_1 \) and \( s_2 \) generate independent messages \( W_1 \) and \( W_2 \) respectively. The message \( W_1 \) is available a priori to destination \( T_2 \). The sources transmit the coded symbols \( X_1 \) and \( X_2 \) respectively into the network. The destinations
receive collections of symbols \( Y_1 \) and \( Y_2 \) on edges \( T_1 \) and \( T_2 \) respectively. We assume that \( s_1 \) is connected to at least one edge in \( T_1 \). In this paper, we only consider scalar linear coding, in which transmitted symbols are treated as elements of some finite field \( \mathbb{F}_q \) of size \( q \). From \( Y_i \), destination \( T_i \) intends to resolve the original message \( W_i \) generated at \( s_i \). For simplicity, in this paper we denote a two-unicast-Z network coding problem \( \Omega \) using a 3-tuple, i.e. \( \Omega = (G,S,T) \), where \( S = \{s_1,s_2\}, T = (T_1,T_2) \).

A rate pair \((R_1,R_2)\) is achievable if for every \( \epsilon > 0, \delta > 0 \), there exists a coding scheme which encodes message \( W_i \) at a rate \( R_i = \delta_i \), for some \( 0 \leq \delta_i \leq \delta \), such that the average error probability of decoding is smaller than \( \epsilon \). The capacity region is the closure of the set of all achievable rate pairs.

Recall that in linear network coding, the output symbols can be represented as linear transformations of the input symbols [1]. Specifically, we have

\[
\begin{align*}
Y_1 &= X_1H_1 + X_2H_2, \\
Y_2 &= X_1G_1 + X_2G_2,
\end{align*}
\]

where, \( H_i \) is the transfer matrix from \( X_i \) to \( Y_1 \) and \( G_i \) is the transfer matrix from \( X_i \) to \( Y_2 \). We denote,

\[
M = \begin{bmatrix} H_1 & G_1 \\ H_2 & G_2 \end{bmatrix}, \quad H = \begin{bmatrix} H_1 \\ H_2 \end{bmatrix}, \quad G = \begin{bmatrix} G_1 \\ G_2 \end{bmatrix},
\]

where we use the vertical bar in \( M \) to separate the transfer matrices for \( T_1 \) and \( T_2 \) edges. Hence, \( Y_1 = XH \) and \( Y_2 = XG \). In particular, each column of \( M \) is the linear transfer vector from all the input symbols to the specific output symbol on the corresponding destination edge. For convenience, we abuse the notation to use the edges to indicate the column of the transfer matrix. For example, the columns of \( H \) correspond to the edges in the destination set \( T_1 \), while the columns of matrix \( G \) correspond to the edges in set \( T_2 \). For a subset \( U \subseteq T_i \), \( H_U \) and \( H_U^t, j = 1,2 \) denote the submatrices of \( H \) and \( H_j \) formed by the columns corresponding to destination edges in \( U \). The matrices \( G_U \) and \( G_U^t \) are defined similarly when \( U \) is a subset of \( T_2 \).

The cardinality of a set \( E \) is denoted by \(|E|\). For set \( A \) and \( B \), \( A \setminus B \) denotes the set of elements in \( A \) but not in \( B \). For a matrix \( A \), \( \text{Span}(A) \) denotes its column span and \( \text{Ker}(A) \) denotes the kernel of the matrix.

### III. Algorithms

In this section, we present a scalar linear network code construction for the two-unicast-Z network problem, which is the main contribution of the paper. That algorithm can be naturally extended to vector linear coding cases. Before introducing the algorithms formally, we consider a simple two-unicast-Z network example shown in Figure 2, where edges are labeled from \( e_1 \) to \( e_8 \). The source nodes are given by \( s_1 = 1 \) and \( s_2 = 2 \), while the destination edges are \( e_7 \) and \( e_8 \) respectively.

Suppose that the input symbols for \( s_1 \) and \( s_2 \) are \( X_1 \) and \( X_2 \) respectively. A linear coding solution to achieve the rate pair \((1,1)\) is given on the graph. In particular, note that the coding happens at node \( 6 \) neutralizes the interference from \( X_2 \) and allow the decode of \( X_1 \) on edge \( e_5 \). Such a coding solution can be easily generated by trial and error for simple networks similar to this, but systematic ways to construct structured coding solutions to achieve such rate pairs is desired. Next, we introduce an algorithm for doing this. We shall revisit the example to illustrate how the algorithm works in later part of this section.

![Fig. 1. A Two-Unicast-Z network](image1)

![Fig. 2. Example 1](image2)

The algorithm consists of two sub-routines, the destination reduction and the recursive code construction.

#### A. Destination reduction

The destination reduction algorithm takes the original problem \( \Omega = (G,S,T) \) and generates a sequence of \( N+1 \) ordered two-unicast-Z network problems, for some \( N \in \mathbb{Z}^+ \), starting with the original problem itself. We denote the sequence of problems as \( \mathcal{P} = \{ \Omega^{(0)}, \Omega^{(1)}, \Omega^{(2)}, \ldots, \Omega^{(N)} \} \), where \( \Omega^{(0)} = \Omega \) and \( \Omega^{(i)} = (G,s_i,T^{(i)}) \) with \( T^{(i)} = \{T_1^{(i)}, T_2^{(i)}\} \) being the destination sets for the problem number \( i \). In particular, all the problems have the same underlying graph \( G \) and source set \( S \), but different destination sets, i.e. \( T^{(i)} \neq T^{(j)}, i \neq j \).

**Algorithm 1 Destination reduction algorithm**

1. **procedure** REDUCTION(\( \Omega^{(0)} \))
2. \( \mathcal{P} \leftarrow \{\} \)
3. add \( \Omega^{(0)} \) to \( \mathcal{P} \)
4. \( i \leftarrow 0 \)
5. \( S \leftarrow \{e : \text{Tail}(e) \in S\} \)
6. while \( T_1^{(i)} \cup T_2^{(i)} \subseteq S \) do
7. \( E \leftarrow \{e : \text{arg max}_{e \in T_1^{(i)} \cup T_2^{(i)}} \text{Ord}(e)\} \)
8. \( E_j \leftarrow E \cap T_j^{(i)}, j = 1,2 \)
9. \( v \leftarrow \text{Tail}(E) \)
10. for \( j \leftarrow 1,2 \) do
11. if \( E_j = \emptyset \) then
12. \( T_j^{(i+1)} \leftarrow T_j^{(i)} \setminus E_j \) \( \cup \) \( \text{In}(v) \)
13. else
14. \( T_j^{(i+1)} \leftarrow T_j^{(i)} \)
15. end if
16. end for
17. \( T^{(i+1)} \leftarrow \{T_1^{(i+1)}, T_2^{(i+1)}\} \)
18. \( \Omega^{(i+1)} \leftarrow (G,s_i, T^{(i+1)}) \)
19. add \( \Omega^{(i+1)} \) into \( \mathcal{P} \)
20. \( i \leftarrow i + 1 \)
21. **end while**
22. **return** \( \mathcal{P} \)
23. **end procedure**
The details of the algorithm are given in Algorithm 1. The key procedure is to sequentially generate $\Omega^{(i+1)}$ from the previous problem $\Omega^{(i)}$. Given $\Omega^{(i)} = (G, S, \mathcal{T}^{(i)})$, where the destination set is $\mathcal{T}^{(i)} = \left\{ T_1^{(i)}, T_2^{(i)} \right\}$, there may be one or more edges that have the highest topological order in the destination sets. This set of topologically highest ordered edges shares a common tail node. Denote this tail node by $v$. For $j \in \{1, 2\}$, let $E_j$ be the set of highest ordered edges in $T_j^{(i)}$, i.e., $E_j = T_j^{(i)} \cap \text{Out}(v)$. For each destination set $j \in \{1, 2\}$, if $T_j^{(i)}$ does not contain any highest topological ordered edge (i.e., $E_j = \emptyset$), then the destination set remains unchanged in $\Omega^{(i+1)}$, i.e., $T_j^{(i+1)} = T_j^{(i)}$. Otherwise, all edges in $E_j$ are removed in $T_j^{(i)}$ and replaced by $\text{In}(v)$ to produce the new destination set $T_j^{(i+1)}$ in $\Omega^{(i+1)}$.

In brief, during the process of going from $\Omega^{(i)}$ to $\Omega^{(i+1)}$, we find the last topologically ordered edges in the union of destination sets, removed them from the destination sets that contains any of them and replace them with their immediate parent edges. As each of the iterations corresponding to one vertex, i.e., the common tail node of the last topologically ordered edges, the total number of iterations $N$ is of the order $O(|V|)$. By the construction of the algorithm, it terminates, the following property is satisfied.

**Property 1:** For the problem $\Omega^{(N)}$, the source nodes connect directly to each of the destination edges, making it a 1-hop network.

With this property, the problem $\Omega^{(N)}$ has a trivial routing solution, where each edge in the destination sets obtains an unique source symbol from the source it connects to.

### B. Recursive coding construction

We introduce an algorithm that allows us to construct a code for $\Omega^{(i+1)}$ from a code on $\Omega^{(i)}$. With the algorithm, we recursively construct codes for problems in $\mathbb{P}$, starting from the trivial routing solution of $\Omega^{(N)}$, and eventually leading to a code for the original problem $\Omega = \Omega^{(0)}$.

**A. Preliminaries**

The following definition and lemma are useful in the construction of the recursive coding algorithm.

**Definition 1 (Grank):** Given a linear code solution two-unicast-Z network represented by Equation (3), the *Grank* of the transfer matrix $M$ is defined as

$$\text{Grank}(M) = \text{rank}\left(\begin{bmatrix} H_1 & G_1 \\ H_2 & G_2 \end{bmatrix}\right) = \text{rank}\left( H_2 \right) - \text{rank}\left( H_1 \right) - \text{rank}\left( G_2 \right).$$

**Remark 1:** Note that in [6], it is shown that $\text{Grank}(M)$ is an algebraic lower bound to the minimum GNS cut value of the network. Furthermore, $\min(\text{Grank}(M), \text{rank}(H_1) + \text{rank}(G_2))$ can be achievable for the sum-rate in the two-unicast-Z network. We will provide a heuristic algorithm that is devised on the intuition of increasing the Grank.

**Lemma 1 (Adding columns does not reduce Grank):** Given matrices $M$ and $M'$, such that

$$M = \begin{bmatrix} H_1 & G_1 \\ H_2 & G_2 \end{bmatrix}, \quad M' = \begin{bmatrix} H_1 & A_1 & G_1 & B_1 \\ H_2 & A_2 & G_2 & B_2 \end{bmatrix},$$

then, $\text{Grank}(M') \geq \text{Grank}(M)$.

**Proof:** The proof is omitted for simplicity.

The recursive coding algorithm takes an existing code for problem $\Omega^{(i+1)} \in \mathbb{P}, 0 \leq i \leq N-1$, and generates a code for the problem $\Omega^{(i)}$. Consider the destination sets of problems $\Omega^{(i+1)}$ and $\Omega^{(i)}$. For $j = 1, 2$, let

$$U_j^{(i)} = T_j^{(i)} \cap T_j^{(i+1)}, \quad I_j^{(i)} = T_j^{(i+1)} \setminus U_j^{(i)}, \quad O_j^{(i)} = T_j^{(i)} \setminus U_j^{(i)}.$$  

That is, from $T_j^{(i+1)}$ to $T_j^{(i)}$, $U_j^{(i)}$ is the unchanged edges, while $I_j^{(i)}$ is the edges that are removed from destination $T_j^{(i+1)}$ and $O_j^{(i)}$ are the newly added edges to $T_j^{(i)}$.

Recall that when the destination reduction algorithm generates $\mathcal{T}^{(i+1)}$ from $\mathcal{T}^{(i)}$, it removes the last topologically ordered edges, which shares a common tail node $v$, and add in the incoming edges of $v$ into the destination sets that has edges removed. As a result, when we go from $\mathcal{T}^{(i+1)}$ back to $\mathcal{T}^{(i)}$ if $O_j^{(i)} \neq \emptyset$ then Head($I_j^{(i)}$) = $v$ = Tail($O_j^{(i)}$). Hence, $I_j^{(i)}$ contains the all incoming edges to node $v$, while $O_j^{(i)}$ consists of subsets outgoing edges from $v$ that communicate with $T_j^{(i)}$. Furthermore, if $O_1^{(i)} \neq \emptyset$ and $O_2^{(i)} \neq \emptyset$, then $I_1^{(i)} = I_2^{(i)} = \text{In}(v)$ and thus $H_1^{(i)} = G_1^{(i)}$.

To distinguish edges in $O_2^{(i)}$ more easily, we further divide $O_2^{(i)}$ into two disjoint sets, $A_2^{(i)}$ and $B_2^{(i)}$, such that $O_2^{(i)} = A_2^{(i)} \cup B_2^{(i)}$, where

$$A_2^{(i)} = O_2^{(i)} \cap O_1^{(i)}, \quad B_2^{(i)} = O_2^{(i)} \setminus O_1^{(i)}.$$  

so that $B_2^{(i)}$ contains edges that only communicate with some edges in $T_2^{(i)}$ but not $T_1^{(i)}$, while $A_2^{(i)}$ contains edges that communications with at least one edge in both $T_1^{(i)}$ and $T_2^{(i)}$.

Now consider the transfer matrix for the code on $\Omega^{(i+1)}$, let it be $M^{(i+1)}$. Following the notations introduced previously on the columns of the transfer matrix, we can write,

$$M^{(i+1)} = \begin{bmatrix} H_1^{(i)} & H_2^{(i)} & G_1^{(i)} & G_1^{(i)} \\ H_2^{(i)} & H_2^{(i)} & G_2^{(i)} & G_2^{(i)} \end{bmatrix}.$$  

Since the problems $\Omega^{(i+1)}$ and $\Omega^{(i)}$ share the same underlying graph and sources, to produce a code for $\Omega^{(i)}$ from the code on $\Omega^{(i+1)}$, we only need to design a local coding matrix at $v$, which generates $H_2^{(i)}$ from $H_1^{(i)}$, and $G_2^{(i)}$ from $G_1^{(i)}$, for $j = 1, 2$. Let $M^{(i)}$ be the transfer matrix of the code on $\Omega^{(i)}$. The columns in $M^{(i)}$ corresponding to edges $U_j^{(i)}$ remain identical to those in $M^{(i+1)}$. Therefore, $M^{(i)}$ is given by,

$$M^{(i)} = \begin{bmatrix} H_1^{(i)} & H_2^{(i)} & G_1^{(i)} & G_1^{(i)} \\ H_2^{(i)} & H_2^{(i)} & G_2^{(i)} & G_2^{(i)} \end{bmatrix}.$$  

It remains to show how the algorithm designs the local coding matrix at node $v$ to generate $H_2^{(i)}$ from $H_1^{(i)}$, and $G_2^{(i)}$ from $G_1^{(i)}$, for $j = 1, 2$. More specifically, we show how the algorithm designs local coding matrices $F_1^{(i)}$, $F_2^{(i)}$ at vertex $v$, with dimensions $|\text{In}(v)| \times |O_1^{(i)}|$, $|\text{In}(v)| \times |A_2^{(i)}|$ and $|\text{In}(v)| \times |B_2^{(i)}|$ respectively, such that,

$$H_2^{(i)} = H_1^{(i)} F_1^{(i)}, \quad G_2^{(i)} = G_1^{(i)} F_2^{(i)}, \quad B_2^{(i)} = G_2^{(i)} F_B^{(i)}.$$  

**B. Recursive Coding Algorithm**

Now we are ready to present the recursive coding algorithm. It proceeds in three steps.

1) *Step 1*: Determining of $F_B^{(i)}$, or equivalently determining $G_2^{(i)}$.

If $B^{(i)} = \emptyset$, proceed to the next step. Otherwise, select each entry of the matrix $F_B^{(i)}$ uniformly at random from the underlying finite field $F_q$. 

The recursive coding algorithm takes an existing code for problem $\Omega^{(i+1)} \in \mathbb{P}, 0 \leq i \leq N-1$, and generates a code for the problem $\Omega^{(i)}$. Consider the destination sets of problems
2) Step 2: Determining of $F_1^{(i)}$, or equivalently determining $H_{O_1}^{(i)}$. After step 1, we have found coding vectors for edges $E_2^{(i)}$, that is the edges in destination $T_2^{(i)}$ but not a part of destination $T_1^{(i)}$. That is, the following matrices are fully determined

$$\mathbf{M}^{(i)} = \begin{bmatrix} H_{I_2}^{(i)} & G_{I_2}^{(i)} & G_{E_2}^{(i)} \\ H_{E_2}^{(i)} & G_{E_2}^{(i)} & G_{E_2}^{(i)} \end{bmatrix},$$

(7)

$$\mathbf{M}^{(i+1)} = \begin{bmatrix} H_{I_2}^{(i)} & H_{I_2}^{(i)} & G_{I_2}^{(i)} & G_{I_2}^{(i)} & G_{E_2}^{(i)} & G_{E_2}^{(i)} \\ H_{E_2}^{(i)} & H_{E_2}^{(i)} & G_{E_2}^{(i)} & G_{E_2}^{(i)} & G_{E_2}^{(i)} & G_{E_2}^{(i)} \end{bmatrix}. $$

(8)

Note that by Lemma 1, $\text{Grank}(\mathbf{M}^{(i+1)}) \geq \text{Grank}(\mathbf{M}^{(i)})$. Next we intend to determine $\mathbf{M}^{(i)}$ fully. We divide this step into two cases based on the Granks of $\mathbf{M}^{(i+1)}$ and $\mathbf{M}^{(i)}$.

Case I: $\text{Grank}(\mathbf{M}^{(i+1)}) = \text{Grank}(\mathbf{M}^{(i)})$.

In this case, pick each entry of the matrices $F_1^{(i)}, j = 1, 2$ uniformly at random from the underlying finite field $\mathbb{F}_q$.

Case II: $\text{Grank}(\mathbf{M}^{(i+1)}) > \text{Grank}(\mathbf{M}^{(i)})$.

We consider two sub-cases, starting with the case where a neutralization step is required.

Case IIa (Neutralization Step): The case applies when both of the following hold,

$$\text{Span}(H_{I_2}^{(i)}) \not\subset \text{Span}(H_{I_2}^{(i)})$$

(9)

$$\text{Span}(H_{E_2}^{(i)}) \subset \text{Span}(H_{I_2}^{(i)}, H_{I_2}^{(i)}, G_{I_2}^{(i)}, G_{I_2}^{(i)}, G_{E_2}^{(i)}, G_{E_2}^{(i)})$$

(10)

The following lemma is useful for constructing the local coding matrix $F_1^{(i)}$ in the neutralization step.

Lemma 2: Given $\text{Grank}(\mathbf{M}^{(i+1)}) > \text{Grank}(\mathbf{M}^{(i)})$, if equations (9) and (10) holds, then there exists a matrix $F_1^{(i)}$ whose columns are non-zero vectors, such that

$$\text{Span}(H_{I_2}^{(i)}, F_1^{(i)}) \subset \text{Span}(H_{I_2}^{(i)}, H_{I_2}^{(i)})$$

(11)

$$\text{Span}(H_{E_2}^{(i)}, F_1^{(i)}) \not\subset \text{Span}(H_{I_2}^{(i)}, H_{I_2}^{(i)})$$

(12)

Proof: See the extended version of the paper [8].

To find such a coding matrix $F_1^{(i)}$ described in Lemma 2, consider the matrix $H_{E_2}^{(i)} = [H_{I_2}^{(i)} H_{I_2}^{(i)}]$. Note that there are $|O_1^{(i)}|$ columns in matrix $F_1^{(i)}$. We pick each column $j$, in the following way.

- Randomly choose a vector $c_j \in \ker(H_{E_2}^{(i)})$. Denote $c_j$ as

$$c_j = \begin{bmatrix} u_j \\ v_j \end{bmatrix}.$$  

(13)

Since $c_j$ belong to the kernel of $H_{E_2}^{(i)}$, we have,

$$H_{E_2}^{(i)} c_j = 0.$$  

(14)

- Set the $j$-th column of $F_1^{(i)}$ to be $c_j$.

When the underlying finite field $\mathbb{F}_q$ is sufficiently large, it can be verified that the above choice of $F_1^{(i)}$ will satisfy equation (11) and (12) with high probability.

Case IIb: Otherwise

If $\text{Grank}(\mathbf{M}^{(i+1)}) > \text{Grank}(\mathbf{M}^{(i)})$ and the condition in Case IIa is not satisfied, we pick the entries in $F_1^{(i)}$ uniformly at random from $\mathbb{F}_q$.

3) Step 3: Determination of $F_2^{(i)}$. If $A_2^{(i)} \neq \emptyset$, then $I_2^{(i)} = I_2^{(i)}$ and $H_{I_2}^{(i)} = G_{I_2}^{(i)}$. Since $A_2^{(i)} \subset O_1^{(i)}$, $G_{A_2^{(i)}}$ is a submatrix of $H_{O_1}^{(i)}$. Therefore, $F_A$ can be obtained by taken the columns of $F_1^{(i)}$ corresponding to $A_2^{(i)}$.

Lemma 3: Given $\text{Grank}(\mathbf{M}^{(i+1)}) > \text{Grank}(\mathbf{M}^{(i)})$, the recursive code construction algorithm will generate the transfer matrix $\mathbf{M}^{(i)}$, such that $\text{Grank}(\mathbf{M}^{(i)}) > \text{Grank}(\mathbf{M}^{(i)})$.

Proof: See [8].

Loosely speaking, Lemma 3 guarantees that if there is a degree of freedom contributed by the $I_2^{(i)}$ edges to the Grank of $\mathbf{M}^{(i+1)}$, it is not lost when the algorithm generates $\mathbf{M}^{(i)}$ from $\mathbf{M}^{(i+1)}$ through the neutralization step. This is consistent with our intuition of maximizing the Grank of the transfer matrix.

Note that the recursive coding algorithm traverse through all the intermediate nodes in their topological order and performs either random coding or neutralization step for outgoing edges. At each node $v$, the complexity of the coding operations is dominated by the complexity of the neutralization step, if it is performed, which is bounded by $O(d^3)$, where $d$ is the in-degree of node $v$. Therefore, we have the following property for the recursive coding algorithm.

Property 2: For the class of directed acyclic graphs with in-degree bounded by $D$, the complexity of the recursive coding algorithm is $O(|\mathcal{V}|D^3)$, which is both polynomial in the number of vertices and in the number of edges in the network.

C. Example

Let us revisit the example introduced at the beginning of the section to see how our algorithm systematically constructs a structured solution. Table I shows the destination sets generated by the destination reduction algorithm for each problem in $\mathcal{P}$, starting from initial destination set $T_1 = a$ and $T_2 = b$. There are 5 problems, $\Omega^{(0)}$ to $\Omega^{(4)}$. The algorithm terminates at the last problem $\Omega^{(4)}$ in which each destination contains only the source edges $\{f,g\}$.

Table II gives the transfer matrix from source edges to the destination edges for each problem in $\mathcal{P}$. These matrices are generated by the recursive algorithm, sequentially from $\Omega^{(4)}$ to $\Omega^{(0)}$. At the initial stage $\Omega^{(4)}$, the matrix is identity, corresponding to the trivial solution when the destination edges themselves are source edges. From $\Omega^{(4)}$ to $\Omega^{(3)}$, we are removing columns corresponding to edge $g$ and generate columns for $d$ and $h$. The neutralization criteria in Case IIa are not satisfied. Hence, random coding (routing in this case, as there is only one input column) is performed. A similar situation happens for all other stages, except the last one. In particular, from $\Omega^{(3)}$ to $\Omega^{(2)}$, column $e$ is generated from columns $f$ and $h$ from random coding, whose coefficients are chosen to be $p$ and $q$.

At the last stage, from $\Omega^{(1)}$ to $\Omega^{(0)}$, we have $H_{I_2}^{(0)} = [1 \ q]$, while $H_{I_2}^{(0)}$ is empty and $G_{E_2}^{(0)} = q$. Given that $p, q$ are randomly chosen, with high probability $p, q \neq 0$ if the underlying field is large enough. Hence, Case IIa criteria are
satisfied. Subsequently, the neutralization step will generate the a column such that the lower part is identically zero to eliminate the interference from g on a. Hence, the column \([-\frac{e_7}{q}, 0, p, q]\) is produced at a and destination 1 is free from the interference of \(s_2\).

D. The performance of the algorithm

We prove for the special case when the two-unicast-Z network aims to achieve (1, 1) rate pairs. An implication of our results is that when there is only one message per source in the two-unicast-Z network, our algorithm is optimal.

**Theorem 1:** The above algorithm cannot achieve the rate pair (1, 1) for a two-unicast-Z network problem, if and only if the graph \(G\) contains an edge whose removal disconnects \((s_1, t_1), (s_2, t_2)\) and \((s_2, t_1)\).

We very briefly outline ideas for the proof of Theorem 1 in this subsection. For the full details of the proof, we refer the reader to the extended version of the paper [8].

1) **Sufficiency:** This follows directly from the GNS outer bound [9].

2) **Necessity:** Given the problem \(\Omega = (G, S, T)\), let the graph \(G' = (V', E')\), such that \(V' = V \cup \{s_1', s_2'\}\) and \(E' = E \cup \{s_1', s_1\} \cup \{s_2', s_2\}\). Define \(S' = (s_1', s_2')\) and a new problem \(\Omega' = (G', S', T)\). We show that if the algorithm do not achieve rate pair (1, 1) for problem \(\Omega'\), then there is an edge in \(E'\) whose removal disconnects \((s_1', t_1), (s_2', t_2)\) and \((s_2', t_1)\). Equivalently, there is an edge in \(E'\), whose removal disconnects \((s_1, t_1), (s_2, t_2)\) and \((s_2, t_1)\) on \(G\). We perform the destination reduction algorithm on \(\Omega'\) to generate problems \(\Omega^{(i)}, 0 \leq i \leq N\) recursively construct codes for \(\Omega^{(N)}\) to \(\Omega^{(0)} = \Omega\). Note that on \(\Omega'\), there is one source edge for each source. Therefore, the transfer matrix \(M^{(1)}\) has exactly two rows, while each of \(H^{(1)}\) and \(G^{(1)}\) is a row vector (or scalar).

We show several claims are true when the graph \(G'\) does not contain an edge whose removal disconnects \((s_1, t_1), (s_2, t_2)\) and \((s_2, t_1)\).

**Claim 1:** For all \(0 \leq i \leq N\), \(H^{(1)i} \neq 0\).

**Claim 2:** For all \(0 \leq i \leq N\), \(G^{(1)i} \neq 0\).

**Claim 3:** For all \(0 \leq i \leq N\), \(\text{Grank}(M^{(i)}) > \text{rank}(G^{(2)i})\).

The basic technique to prove the three claims is to use induction on \(i\) and examine various cases as the coding algorithm generates codes for problem \(\Omega^{(i+1)}\) from that of problem \(\Omega^{(i)}\). We assume that underlying field \(F_q\) for the transmitted symbols is sufficiently large, such that at any stage, any two columns generated by random coding from more than one input edges (columns) are linearly independent. The following lemma is useful for considering the change of \(\text{Grank}(M^{(i)})\) when neutralization step is carried out.

**Lemma 4:** For graph \(G'\), if there is a \(k\) such that, from stage \(k + 1\) to \(k\), the algorithm performs neutralization step, then for all \(t < k\), \(H^{(2)}_t = 0\).

IV. DISCUSSION

The result of [10], which describes fundamental connections between index coding and general network coding capacity, implies that any linear network coding approach for the general index coding problem can be used to obtain a linear network code for a two-unicast-Z network. A performance comparison between index-coding based heuristics for the two-unicast-Z networks and our approach is under investigation. We however note that our approach has a linear complexity in the number of vertices (and a polynomial complexity in the number of edges), there are no obvious provable complexity guarantees for the index-coding solutions of [11], [12], since they rely on linear programming approaches. The question of whether our algorithm outperforms routing in general is under investigation.

**REFERENCES**


