Overcoming Delay, Synchronization

and Cyclic Paths

Meir Feder

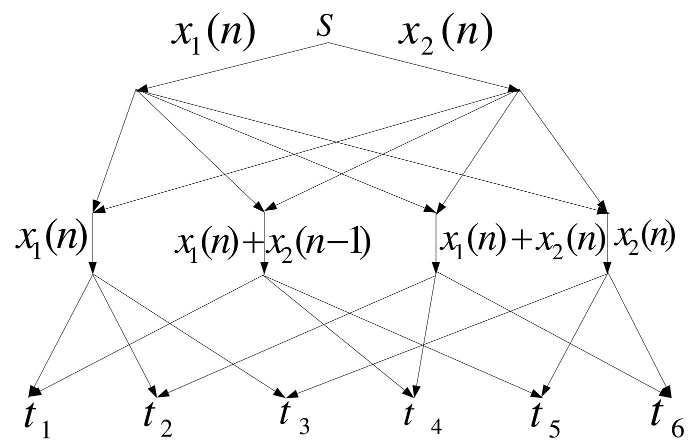
Tel Aviv University

joint work with Elona Erez

Root of the problem

- In much of network coding assume "instantaneous coding"
- Instantaneous coding cannot work with cycles
- Node delay, which may be beneficial cycles, introduces a synchronization problem in code implementation
- How to deal with node delay in case of long input sequence?
- What about decoding Delay?
- **Solution: Convolutional codes**

Motivating Example

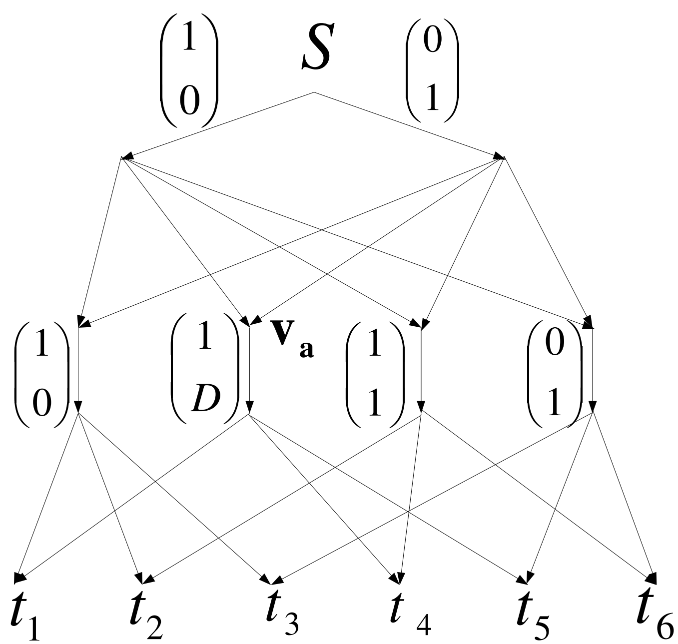


- At $n = 4$ sink t_1 receives $x_1(0)$ on both of its incoming links.
- At time instant $n = 5$, t_1 receives $x_1(1)$ and $x_1(1) + x_2(0)$.
- The effective decoding delay is 5.
- Only a single memory element is required.

Convolutional Network Codes - Definition

- Let $F(D)$ be the ring of polynomials over the binary field.

- Link e is associated with $\mathbf{b}(e)$, whose elements are in $F(D)$.



- Input stream $x_i(n)$ can be represented by a power series:

$$\boxed{X_i(D) = \sum_{n=0}^{\infty} x_i(n)D^n, i = 1, \cdots, h}$$

- In linear convolutional network codes:

$$Y_e(D) = \sum_{n=0}^{\infty} y_e(n)D^n = \sum_{e' \in \Gamma_I(v)} m_e(e')Y_{e'}(D) = \mathbf{b}(e)^T \mathbf{x}(\mathbf{D})$$

where $y_e(n)$ are the symbols transmitted on the link e.

- To achieve rate h, the global coding vectors on the incoming links to t have to span $F[D]^h$, where $F[D]$ is the field of rational function over the binary field.

Dealing with Cycles

- Previous Results
- Precoding
- Code Construction
- Algorithm Complexity
- Decoding Delay

Previous Results

- Ahlswede et al (00): the cyclic network was unrolled into an acyclic layered network.
  - The resulting scheme is time-variant, requires complex encoding/decoding and large delay
- Koetter and Médard (02): if each edge has delay, then there exists a time-invariant linear code with optimal rate.
- Li et al (03): a heuristic code construction for a linear time-invariant code.

Line Graph

- Originally the network is modeled as a directed graph $G = (V, E)$
- $L(\mathcal{V}, \mathcal{E})$ is the line graph with:
  - Vertex Set: $\mathcal{V} = E \cup s \cup T$
  - Edge Set: $\mathcal{E} = \{(e, e') \in E^2 : \text{head}(e) = \text{tail}(e')\} \cup \{(s, e) : e \text{ outgoing from } s\} \cup \{(e, t_i) : e \text{ incoming to } t_i, 1 \leq i \leq d\}$
- If there are h edge-disjoint paths between s and t in G, there are corresponding h node-disjoint paths in L.

Recall the following:

- h: the minimal min-cut between s and any of the sinks $T = \{t_1, \cdots, t_d\}$
- $F(D)$: the ring of polynomials with binary coefficients
- $F[D]$: the field of rational function over the binary field.
- $\mathbf{v}(e)$: a global coding vector (whose components maybe in $F[D]$) assigned to node $e \in L$.
- The code can be used for multicast if and only if for all $t \in T$, the global coding vectors incoming into t span $F[D]^h$.

Precoding

- We find a set of nodes $\mathcal{E}_D$ in L, such that if we eliminate them, there will be no directed cycles.
- To insure that each cycle will contain at least a single delay, the coding coefficients for this set will be restricted to be polynomials with D as a common component.
- To maintain the same number of possible coding coefficients, if we choose polynomials with maximal degree M, then for $e \in \mathcal{E}_D$ the maximal polynomial degree is $M + 1$.

- In order to minimize the delay, it is desired to minimize $|\mathcal{E}_D|$.
- Finding the minimal $\mathcal{E}_D$ is the long standing problem of finding the minimal arc feedback set, which is NP-hard.
- The best known approximation algorithm with polynomial complexity achieves performance ratio $O(\log|V|\log\log|V|)$.
- For our purposes - use any approximate solution to insert enough delays in the cycles.

Code Construction

- The code construction goes in steps over the terminals:

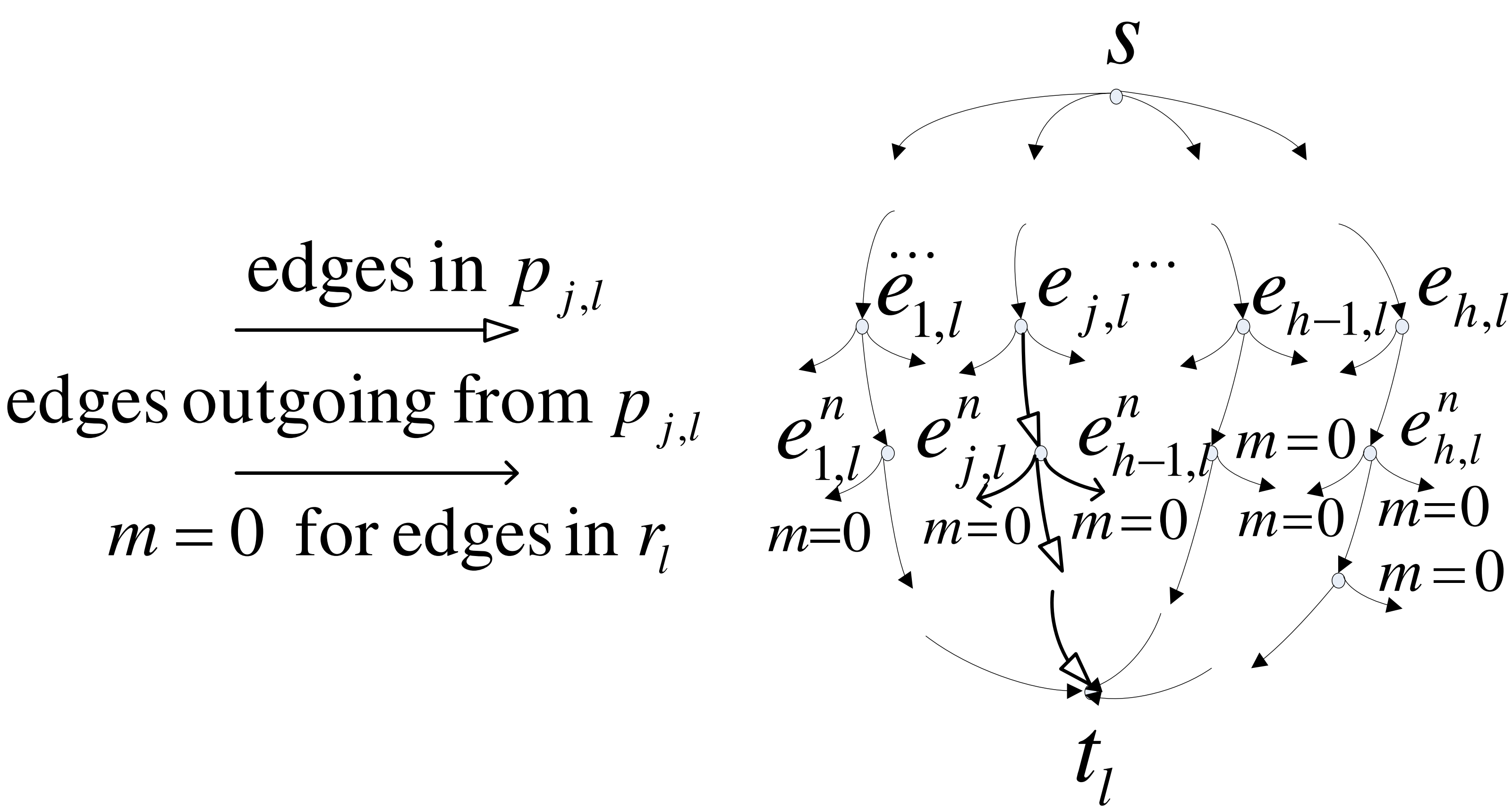
  Let L_l be the sub-graph consisting only of the nodes that participate in the flow from s to t_l. L_l is *acyclic*.

  – Go over the nodes $e \in L_l$ in a topological order.

  – Maintains a list of h nodes $\mathcal{C}_l = \{e_{1,l}, \cdots, e_{i,l}, \cdots, e_{h,l}\}$, each belongs to a different path.

- Some definitions:
  - $P_{j,l}$: the jth path of the flow from s to t_l.
  - $p_{j,l} \subset P_{j,l}$: the set of nodes following $e_{j,l} \in \mathcal{C}_l$ (not including $e_{j,l}$).
  - $c_{j,l}$: the set of coding coefficients of edges with tail in $p_{j,l}$ and head in $L \setminus p_{j,l}$.
  - r_l: the union of these sets of coefficients:

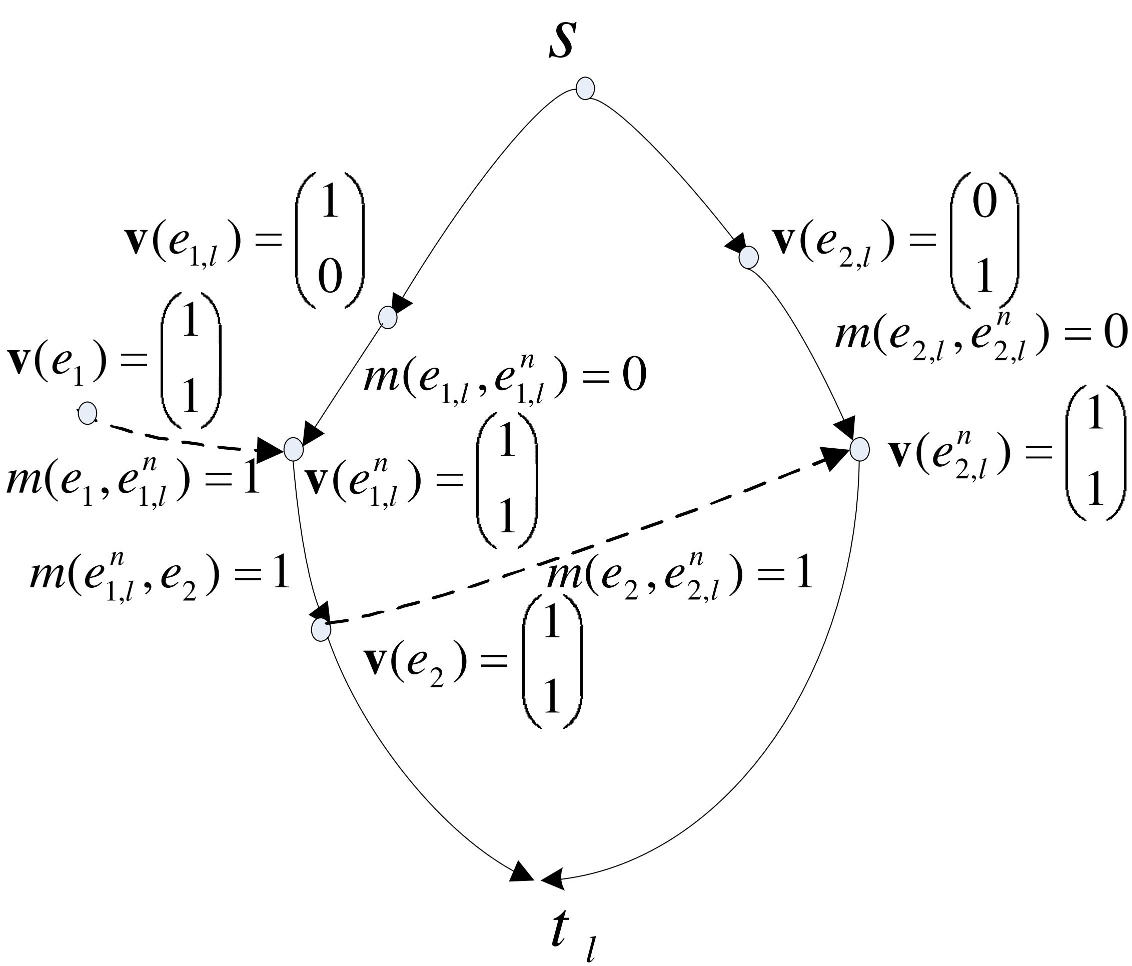
$$r_l = \cup_{1 \leq j \leq h} c_{j,l}$$


edges in $p_{j,l}$
edges outgoing from $p_{j,l}$
$m = 0$ for edges in r_l
s
$e_{1,l}$
$e_{j,l}$
$e_{h-1,l}$
$e_{h,l}$
$e^n_{1,l}$
$e^n_{j,l}$
$e^n_{h-1,l}$
$e^n_{h,l}$
$m=0$
$m=0$
$m=0$
$m=0$
$m=0$
$m=0$
$m=0$
t_l

The partial coding vector - $\mathbf{u}(e)$

- $\mathbf{u}(e)$ is defined for all $e \in \mathcal{C}_l$ as the global coding vector of e when all the coefficients in r_l are set to zero.
- $V_l = \{\mathbf{v}(e) : e \in \mathcal{C}_l\}$, $U_l = \{\mathbf{u}(e) : e \in \mathcal{C}_l\}$.
- Requiring V_l to span $F[D]^h$ is *not* sufficient.
- Requiring U_l to span $F[D]^h$ is sufficient.
- At the end of step l, $V_l = U_l$.

Requiring V_l to span $F[D]^h$ is *not* sufficient:



- The dashed edges are edges in L_k for some $k < l$.
- The current edges in $\mathcal{C}_l$ are $e_{1,l}$ and $e_{2,l}$.
- We have reached $e^n_{2,l}$ in the topological order.
- The previous value $m(e_{2,l}, e^n_{2,l}) = 0$ remains since $\mathbf{v}(e^n_{2,l})$ and $\mathbf{v}(e_{1,l})$ are already a basis .
- Next we reach $e^n_{1,l}$ and we need to determine $m(e_{1,l}, e^n_{1,l})$.
- But for any value of $m(e_{1,l}, e^n_{1,l})$, we have $\mathbf{v}(e^n_{1,l}) = \mathbf{v}(e^n_{2,l})$ and the new set of vectors cannot be a basis!

Returning to the algorithm...

- The algorithm reached node $e_{i,l}$; wishes to continue to $e^n_{i,l}$, the following node in $P_{i,l}$
- So far, the set $U_l = \{\mathbf{u}(e) : e \in \mathcal{C}_l\}$ is a basis.
- A new list is generated: $\mathcal{C}^n_l = \mathcal{C}_l \cup e^n_{i,l} \setminus e_{i,l}$.
- There is a new set of partial coding vectors:

  $U^n_l = \{\mathbf{u}^n(e_{1,l}), \cdots, \mathbf{u}^n(e^n_{i,l}), \cdots, \mathbf{u}^n(e_{h,l})\}$.

- The algorithm determines a coding coefficient $m(e_{i,l}, e^n_{i,l})$ between node $e_{i,l}$ and $e^n_{i,l}$ so that U^n_l will be a basis.

- Let $m'(e_{i,l}, e^n_{i,l})$ be the coding coefficient between $e_{i,l}$ and $e^n_{i,l}$ before this stage of the algorithm.

  – If with $m'(e_{i,l}, e^n_{i,l})$ U^n_l is a basis - done!

  – Otherwise - we have the following Theorem:

    Theorem 1 *Suppose that with $m'(e_{i,l}, e^n_{i,l})$ the set U^n_l is <u>not</u> a basis. Then with any other value $m(e_{i,l}, e^n_{i,l})$ the set U^n_l will be a basis.*

But what about the previous sinks?

- Changing $m'(e_{i,l}, e^n_{i,l})$ changes the coding vectors incoming at the previous sinks. May not be a basis anymore!
- **Theorem 2** *Let $\mathcal{C}_k$ be the set of nodes incoming into the sink $t_k, k < l$. Denote by $V'_k = \{\mathbf{v}'(e_{1,k}), \cdots, \mathbf{v}'(e_{h,k})\}$, $e_{j,k} \in \mathcal{C}_k$, the set of global coding vectors of $\mathcal{C}_k$ with $m'(e_{i,l}, e^n_{i,l})$.*

  If V'_k is a basis, then at most a single value of new coefficient $m(e_{i,l}, e^n_{i,l})$ will cause the new set $V_k = \{\mathbf{v}(e_{1,k}), \cdots, \mathbf{v}(e_{h,k})\}$ not to be a basis.

Summing it all up

- If $m'(e_{i,l}, e^n_{i,l})$ must be replaced, pick a new value $m(e_{i,l}, e^n_{i,l})$ according to some enumeration.
  - Check if the independence condition is satisfied for all sinks. Otherwise take the next value for $m(e_{i,l}, e^n_{i,l})$.
  - Since for each sink only a single choice of $m(e_{i,l}, e^n_{i,l})$ is bad, it is sufficient to enumerate over $d + 1$ coefficients.
- The l-step continues until it reaches the sink t_l,
- The algorithm terminates when it goes over all d sinks.

Computation of transfer functions

- In the construction algorithm the transfer function from a certain node to another node has to be computed in each stage.
- Define for the line graph L the $|E| \times |E|$ matrix C where $C_{i,j} = m(e_i, e_j)$ for $(e_i, e_j) \in L$ and zero otherwise.
- Koetter and Médard (02): The transfer function between e_i and e_j is $F_{i,j}$, of the matrix $F = (I - C)^{-1} = I + C + C^2 + \cdots$.
- $F_{i,j}$ can be computed from C with complexity $O(|E|^2)$.

Complexity of Code Construction

- The complexity of the precoding depends on the specific algorithm chosen.
- The algorithm begins by finding the d flows from the source s to the sinks at complexity $O(d|E|h)$.
- The algorithm has d steps and in each it may go over all nodes:
  - For each stage, when check a possible $m(\cdot,\cdot)$:
    - May compute dh transfer functions at complexity $O(dh|E|^2)$

    * Perform independence test for U_l at complexity $O(h)$, and independence test for the other sinks at complexity $O(dh^2)$
  - In the average case check a constant number of $m(\cdot,\cdot)$'s, thus stage complexity $O(dh^2 + dh|E|^2) = O(dh|E|^2)$
  - At the worst case check d values - complexity $O(d^2h|E|^2)$.

- Total complexity: $O(d^2h|E|^3)$ in the average case and $O(d^3h|E|^3)$ in the worse case.

Comparison

- Jaggi el al, 2003, presented an algorithm for *acyclic* networks with complexity $O(|E|dh^2)$ on average and $O(|E|dh^2 + |E|hd^3)$ in the worst case.
- Our algorithm can also be used for acyclic networks at complexity $O(|E|d^2h^2 + |E|^2)$ in the average case and $O(|E|d^3h^2 + |E|^2)$ in the worst case.

A single delay in a cycle

- In Koetter and Médard (02) analysis for cyclic networks it is assumed that each node in L has a single delay.
- But as we have shown it is sufficient to have only a single delay for each cycle in the network.
- Song et al (05) showed that for this "asynchronous transmission" the min-cut is an upper bound on the possible rate.
- Since this bound is achievable, this bound is tight.

Adding and Removing Sinks

- The existing construction algorithms (for acyclic networks) do not provide a simple way to add and remove sinks.
- In our algorithm - adding a new sink simply corresponds to a new step in the algorithm, as only coding coefficients in the flow between the source and the new sink might be changed.
- Removing sinks is analogous to adding sinks.
- The efficient algorithm for removing and adding sinks can be performed for block or convolutional linear network codes.

Decoding Delay of the Sequential Decoder

I. Acyclic Networks

- The delay of the sequential decoder proposed in Erez and Feder (04) is defined by the determinant of the matrix $A(D)$, whose columns are the coding vectors in V_l:
  - If the term with the smallest power of the determinant is D^N, then the delay is at most N.

- For each coding coefficient we can choose from M polynomials.
- For node e incoming into sink t_l let $P_m(e)$ be the path from s to e in the flow L_l; denote by $l_m(e)$ the length of $P_m(e)$.
- It can be shown that for a random code the average delay at t_l, for any $M > d$ is bounded by

$$\overline{\text{delay}(t_l)} = \sum_{e \in \Gamma_{in}(t_l)} l_m(e)$$

Probability Distribution of Delay

- The cost of the flow is given by

$$l_m = \sum_{e \in \Gamma_{in}(t_l)} l_m(e)$$

- For random codes, for large M, the distribution of the delay:

$$P(\text{delay} = q) = \binom{l_m + q - 1}{q} \left(\frac{1}{2}\right)^{q+l_m}$$

- The distribution is better for smaller M, as long as $M > d$.

II. Cyclic Networks

- For cyclic networks, the precoding stage adds delays even for block codes $\Rightarrow$ the sequential decoder is useful both for block and convolutional codes.
- The elements of $A(D)$ might in general be rational functions, where the denominator of each element is indivisible by D.
- Therefore the least common multiplier of the denominators, denoted by lcm is also indivisible by D.

- If we multiply $A(D)$ by lcm to yield $\tilde{A}(D)$, then the determinant is multiplied by lcm^h.
- $A(D)$ and $\tilde{A}(D)$ have the same delay with the sequential decoder.
- The delay for $\tilde{A}(D)$ is determined by the sum of delays accumulated along the h paths between the source and the sink.
- In comparison to acyclic networks, this delay might increase only by the number of nodes in $\mathcal{E}_D$.

Proof of Theorems

Lemma 1

Lemma 1 *Consider a set of nodes* $\{e_1, \cdots, e_h\}$ *and their coding vectors* $W = \{\mathbf{w}(e_1), \cdots, \mathbf{w}(e_i), \cdots, \mathbf{w}(e_h)\}$, *which may be partial or global coding vectors. Consider now the coding vectors of the same set of nodes*

$\tilde{W} = \{\tilde{\mathbf{w}}(e_1), \cdots, \tilde{\mathbf{w}}(e_i), \cdots, \tilde{\mathbf{w}}(e_h)\}$, *when* $m(e_i, e) = 0$ *for* $\forall e \in L$. *The set* W *is a basis iff the set* $\tilde{W}$ *is a basis.*

Proof Outline

- Split node e_i into 3 nodes: e_{tail}, e_{mid} and e_{head}, connected by edges (e_{tail}, e_{mid}) and (e_{mid}, e_{head}).
- G_{ee}: the transfer function from e_{head} to e_{tail} in $L \setminus e_{mid}$.
- The relation between $\mathbf{w}(e_i)$ and $\tilde{\mathbf{w}}(e_i)$ is:

$$\mathbf{w}(e_i) = \tilde{\mathbf{w}}(e_i) + G_{ee}\tilde{\mathbf{w}}(e_i) + \cdots = \frac{1}{1 - G_{ee}}\tilde{\mathbf{w}}(e_i)$$

- The other vectors are given by:

$$\mathbf{w}(e_j) = \tilde{\mathbf{w}}(e_j) + F_{ij}\frac{1}{1 - G_{ee}}\tilde{\mathbf{w}}(e_i), j \neq i$$

where F_{ij} is the transfer function from e_i to e_j.

- The relation between W and $\tilde{W}$ is linear and inverse $\Rightarrow$

a basis W will be mapped to a basis $\tilde{W}$ and vice versa.

Proof of Theorem 1

- Denote the coding vectors of $\mathcal{C}_l^n$ when all the coefficients in r_l are zero by $\tilde{U}_l^n = \{\tilde{\mathbf{u}}^n(e_{1,l}), \cdots, \tilde{\mathbf{u}}^n(e_{i,l}^n), \cdots, \tilde{\mathbf{u}}^n(e_{h,l})\}$.

- Assume $U_l = \{\mathbf{u}(e_{1,l}), \cdots, \mathbf{u}(e_{i,l}), \cdots, \mathbf{u}(e_{h,l})\}$ is a basis.

- After replacing $m'(e_{i,l}, e_{i,l}^n)$ by $m(e_{i,l}, e_{i,l}^n)$, $\tilde{\mathbf{u}}^n(e_{i,l}^n)$ equals:

$$\tilde{\mathbf{u}}^n(e_{i,l}^n) = \tilde{\mathbf{u}}'(e_{i,l}^n) + (m(e_{i,l}, e_{i,l}^n) - m'(e_{i,l}, e_{i,l}^n))\mathbf{u}(e_{i,l})$$

- Using this relation it can be shown that if U_l is a basis and if with $m'(e_{i,l}, e^n_{i,l})$ the set $\tilde{U}^n_l$ is not a basis, then for any other $m(e_{i,l}, e^n_{i,l})$ the set $\tilde{U}^n_l$ is a basis.
- From Lemma 1 it follows that the set U^n_l is also a basis.

Proof of Theorem 2

- Before the replacement of $m'(e_{i,l}, e^n_{i,l})$ the set

  $V'_k = \{\mathbf{v}'(e_{1,k}), \cdots, \mathbf{v}'(e_{h,k})\}$ is a basis.

- We want to analyze when after the replacement to

  $m(e_{i,l}, e^n_{i,l})$ the new set of global coding vectors

  $V_k = \{\mathbf{v}(e_{1,k}), \cdots, \mathbf{v}(e_{h,k})\}$, is also basis.

- Assume that the edges outgoing from $e_{i,l}$, except $(e_{i,l}, e^n_{i,l})$,

  are $\Gamma_o = \{(e_{i,l}, e_1), \cdots, (e_{i,l}, e_q)\}$.

- The system G_{ee} can be expressed as

  $G_{ee} = G_1 + m(e_{i,l}, e^n_{i,l})G_2,$

- The global coding vector $\mathbf{v}(e_{i,l})$ is given by:

$$\begin{aligned} \mathbf{v}(e_{i,l}) &= \tilde{\mathbf{v}}(e_{i,l}) + G_{ee}\tilde{\mathbf{v}}(e_{i,l}) + \cdots = \frac{1}{1-G_{ee}}\tilde{\mathbf{v}}(e_{i,l}) \\ &= \frac{1}{1-m(e_{i,l}, e^n_{i,l})Q}\mathbf{y}(e_{i,l}) \end{aligned}$$

where $Q = G_2/(1-G_1)$ and $\mathbf{y}(e_{i,l}) = \tilde{\mathbf{v}}(e_{i,l})/(1-G_1)$.

- Using the linearity of the code, it can be shown that for $1 \leq j \leq h$:

$$\mathbf{v}(e_{j,k}) - \mathbf{v}'(e_{j,k}) = f(m(e_{i,l}, e^n_{i,l}))H_j\mathbf{y}(e_{i,l})$$

where $H_j \equiv F_{1,j} + QF_{2,j}$ and $F_{1,j}$ is the transfer function from $e_{i,l}$ to $e_{j,k}$, when $m(e_{i,l}, e) = 0, e \in \Gamma_o \setminus (e_{i,l}, e^n_{i,l})$, and $F_{2,j}$ when only the coefficient $m(e_{i,l}, e^n_{i,l}) = 0$.

- Suppose the representation of $\mathbf{y}(e_{i,l})$ in basis V'_k is:

$$\mathbf{y}(e_{i,l}) = \beta_1 \mathbf{v}'(e_{1,k}) + \beta_2 \mathbf{v}'(e_{2,k}) + \cdots + \beta_h \mathbf{v}'(e_{h,k})$$

- Using this relation and the assumption that V'_k is a basis, the set V_k is not a basis only if the following set of equation has a non trivial solution:

$$-\frac{1}{f(m(e_{i,l}, e^n_{i,l}))} \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_h \end{pmatrix} = \begin{pmatrix} H_1\beta_1 & \cdots & H_h\beta_1 \\ \vdots & \ddots & \vdots \\ H_1\beta_h & \cdots & H_h\beta_h \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_h \end{pmatrix}$$

- A non trivial solution exist only if the matrix has eigenvalue:

$$\lambda = -\frac{1}{f(m(e_{i,l}, e^{n}_{i,l}))}$$

- The matrix has eigenvalue 0 with multitude $h-1$ and:

$$\lambda = \mathrm{trace}(A) = H_1\beta_1 + H_2\beta_2 + \cdots + H_h\beta_h$$

with multitude 1.

- It can be shown that V_k is not a basis only for,

$$m(e_{i,l}, e^n_{i,l}) = \frac{m'(e_{i,l}, e^n_{i,l}) - \frac{1-m'(e_{i,l},e^n_{i,l})Q}{\text{trace}(A)}}{1 - Q\frac{1-m'(e_{i,l},e^n_{i,l})Q}{\text{trace}(A)}}$$

$\Rightarrow$ for at most a single choice of $m(e_{i,l}, e^n_{i,l})$ the set V_k will not be a basis.