Network Embedding Operations Preserving Insufficiency of Linear Codes

Congduan Li, Steven Weber, John Walsh

Drexel University, ECE Dept
ASPITRG & MANL

congduan.li@gmail.com

Oct 3, 2014
Allerton confernce, UIUC, IL

Supports from NSF (CCF1016580 & 1421828) are greatly appreciated

Outline

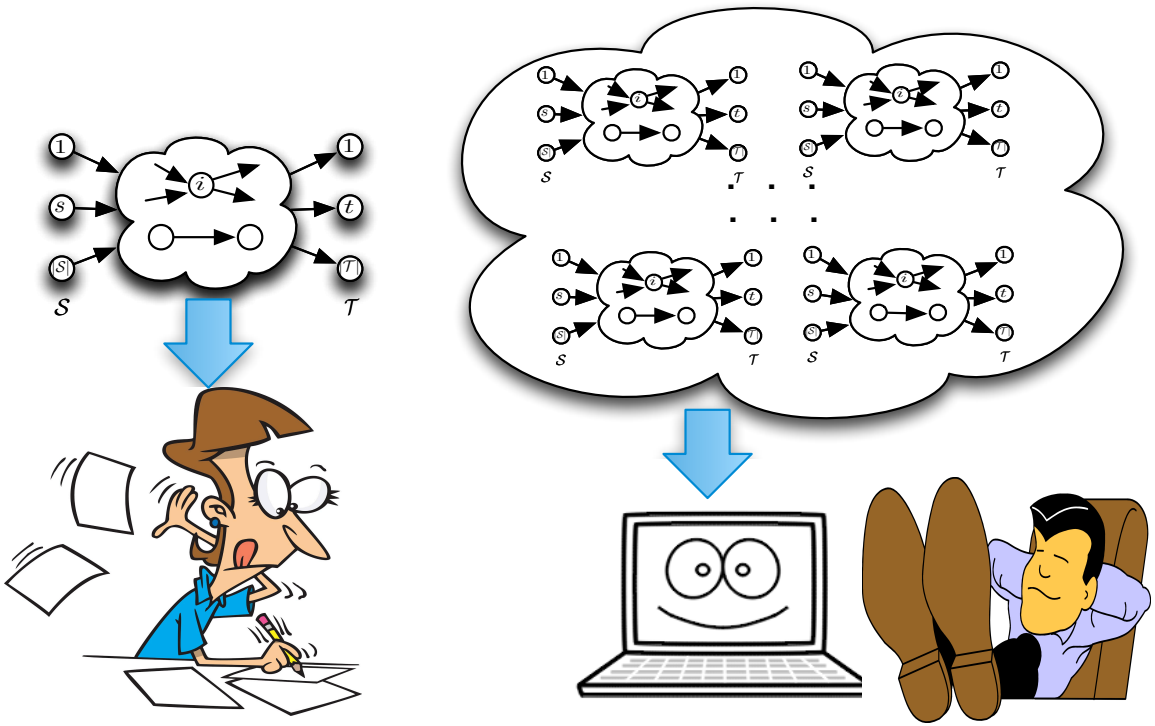
1. Motivation
2. Background on Rate regions and codes
3. Network operations
   - Source deletion
   - Edge deletion
   - Edge contraction
4. MDCS demonstration
   - MDCS model
   - Corresponding operations
   - Forbidden embedded networks for scalar binary codes

Outline

1. Motivation
2. Background on Rate regions and codes
3. Network operations
   - Source deletion
   - Edge deletion
   - Edge contraction
4. MDCS demonstration
   - MDCS model
   - Corresponding operations
   - Forbidden embedded networks for scalar binary codes

Related work

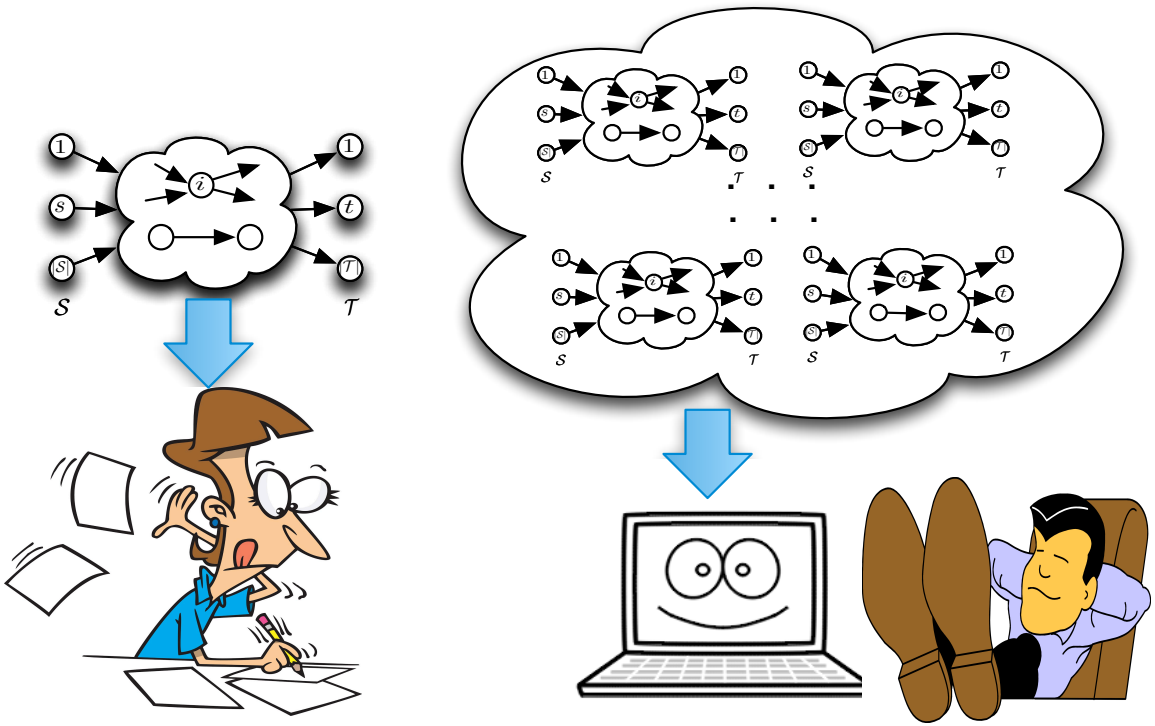
- Related work in recent TransIT submission [1] (available on arXiv)
- Software: 7360 non-isomorphic, 134617 isomorphic networks
- Conventional: 1 network, information inequalities, manually, 1 paper
- Computational: thousands networks, computer, how many papers?
- Limitation on large networks: symmetries, replications, *inheritance*



Conventional way Computational way

Related work

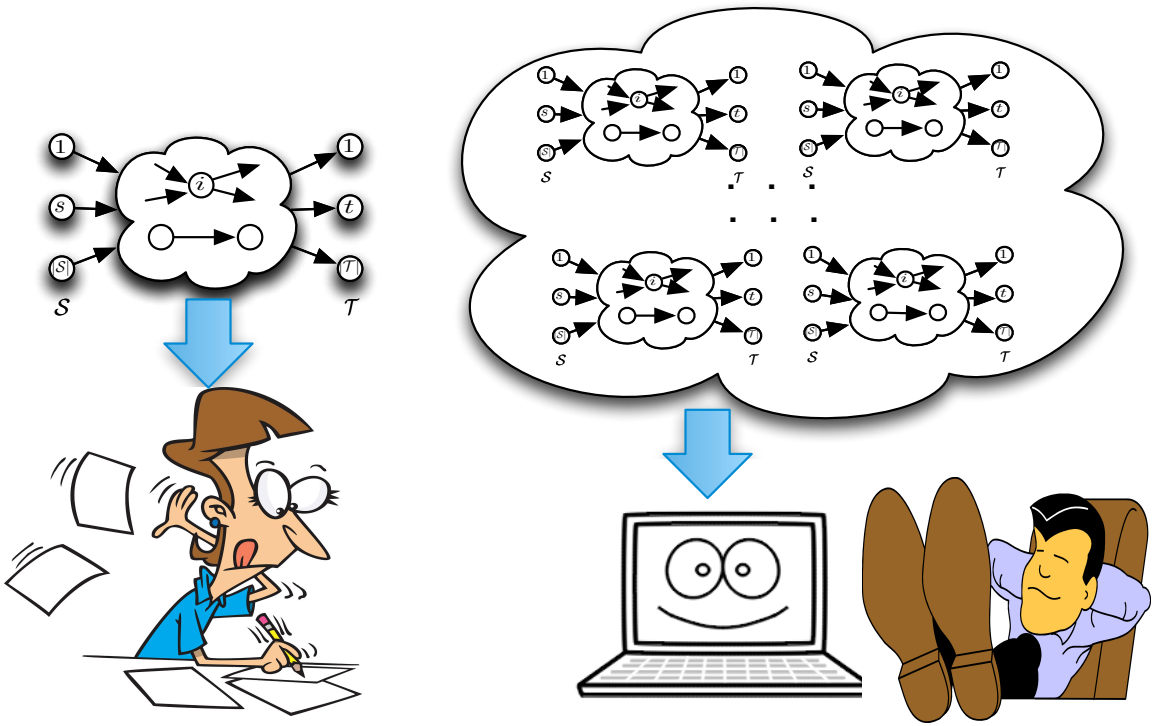
- Related work in recent TransIT submission [1] (available on arXiv)
- Software: 7360 non-isomorphic, 134617 isomorphic networks
- Conventional: 1 network, information inequalities, manually, 1 paper
- Computational: thousands networks, computer, how many papers?
- Limitation on large networks: symmetries, replications, *inheritance*



Conventional way Computational way

Related work

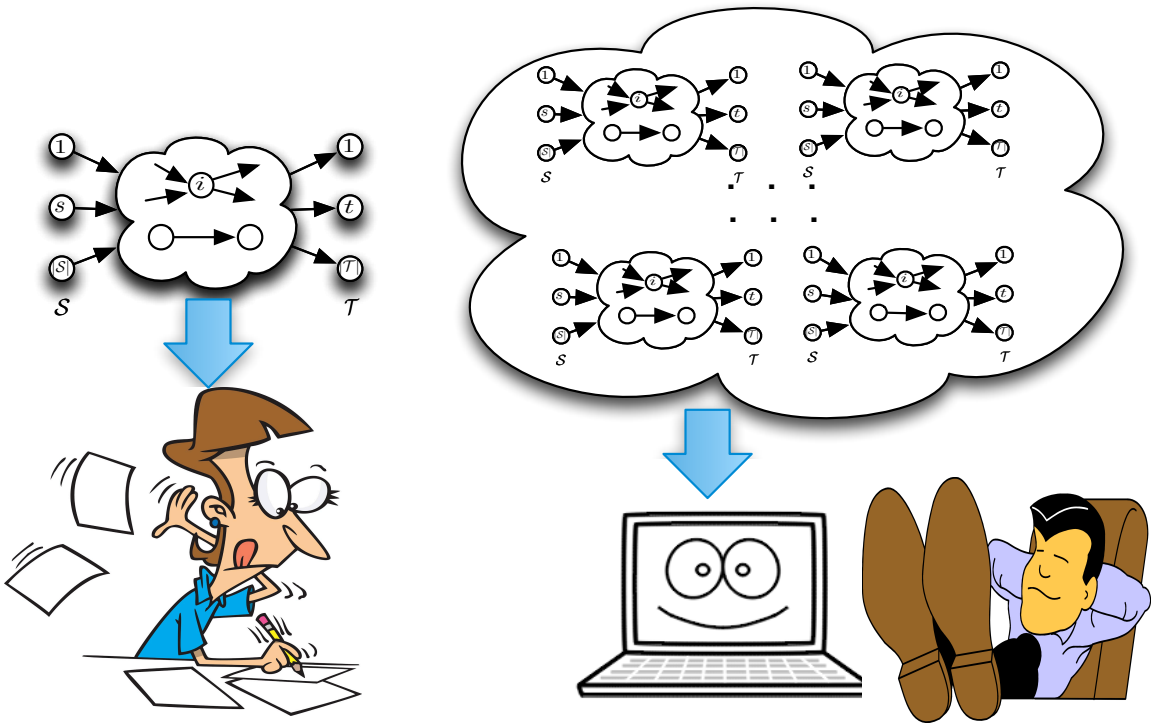
- Related work in recent TransIT submission [1] (available on arXiv)
- Software: 7360 non-isomorphic, 134617 isomorphic networks
- Conventional: 1 network, information inequalities, manually, 1 paper
- Computational: thousands networks, computer, how many papers?
- Limitation on large networks: symmetries, replications, *inheritance*



Conventional way Computational way

Related work

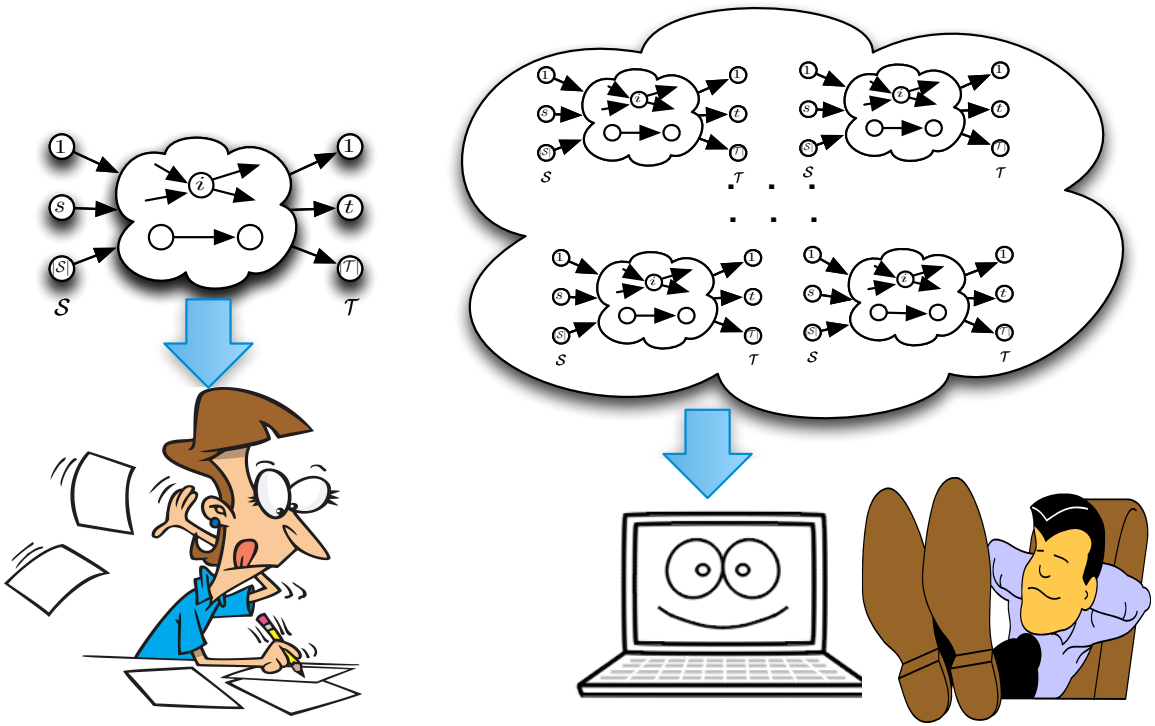
- Related work in recent TransIT submission [1] (available on arXiv)
- Software: 7360 non-isomorphic, 134617 isomorphic networks
- Conventional: 1 network, information inequalities, manually, 1 paper
- Computational: thousands networks, computer, how many papers?
- Limitation on large networks: symmetries, replications, *inheritance*



Conventional way Computational way

Related work

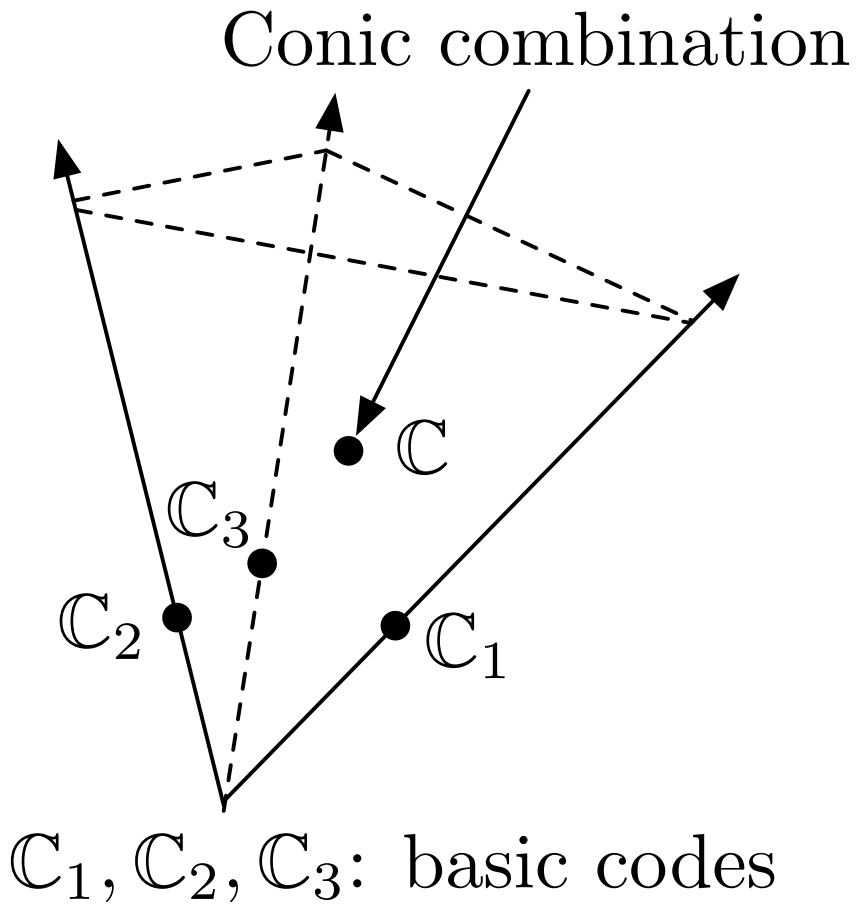
- Related work in recent TransIT submission [1] (available on arXiv)
- Software: 7360 non-isomorphic, 134617 isomorphic networks
- Conventional: 1 network, information inequalities, manually, 1 paper
- Computational: thousands networks, computer, how many papers?
- Limitation on large networks: symmetries, replications, *inheritance*



Conventional way Computational way

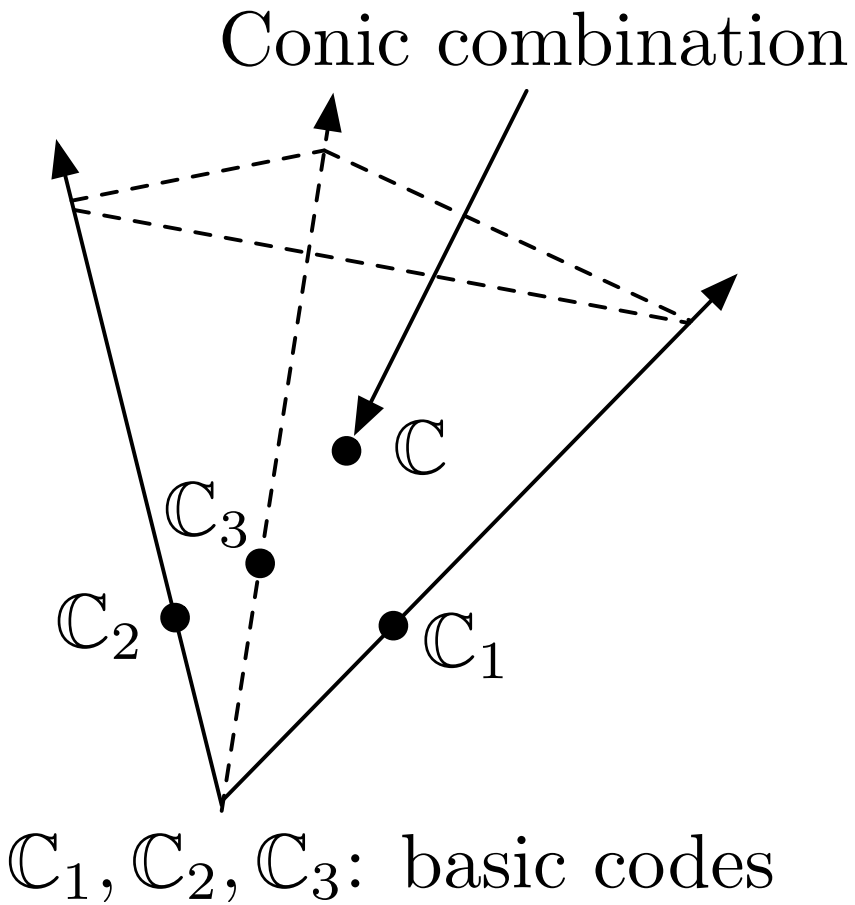
This work

- This work: inheritance property for sufficiency of $\mathbb{F}_q$ linear codes
- Sufficiency: $\mathbb{F}_q$ codes to achieve every point in rate region



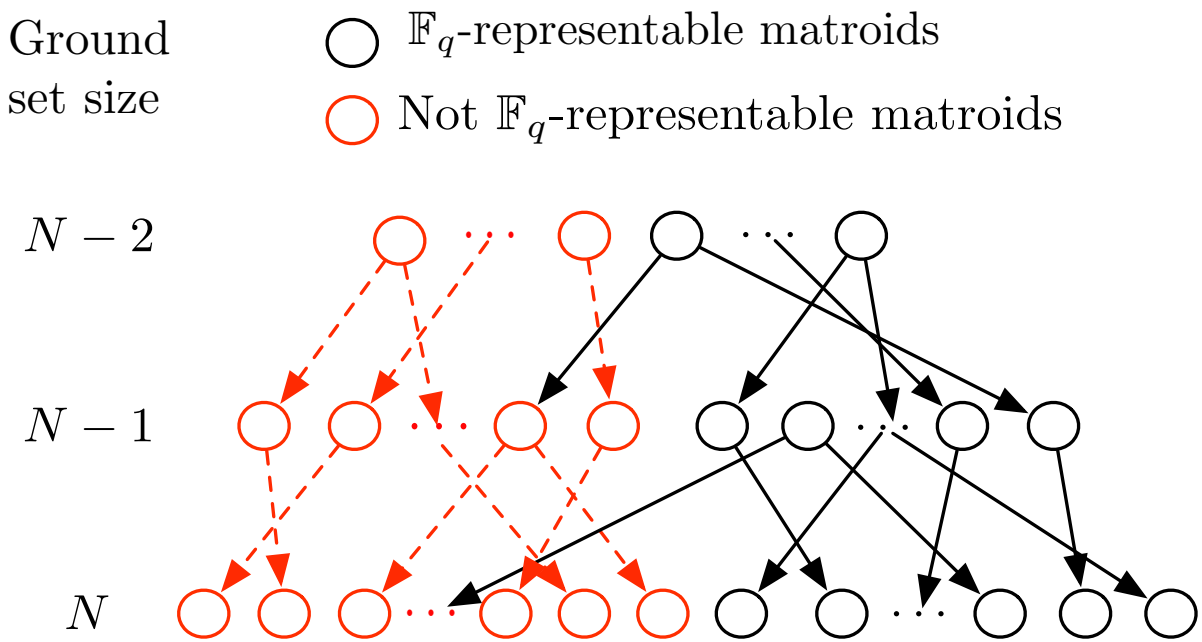
This work

- This work: inheritance property for sufficiency of $\mathbb{F}_q$ linear codes
- Sufficiency: $\mathbb{F}_q$ codes to achieve every point in rate region



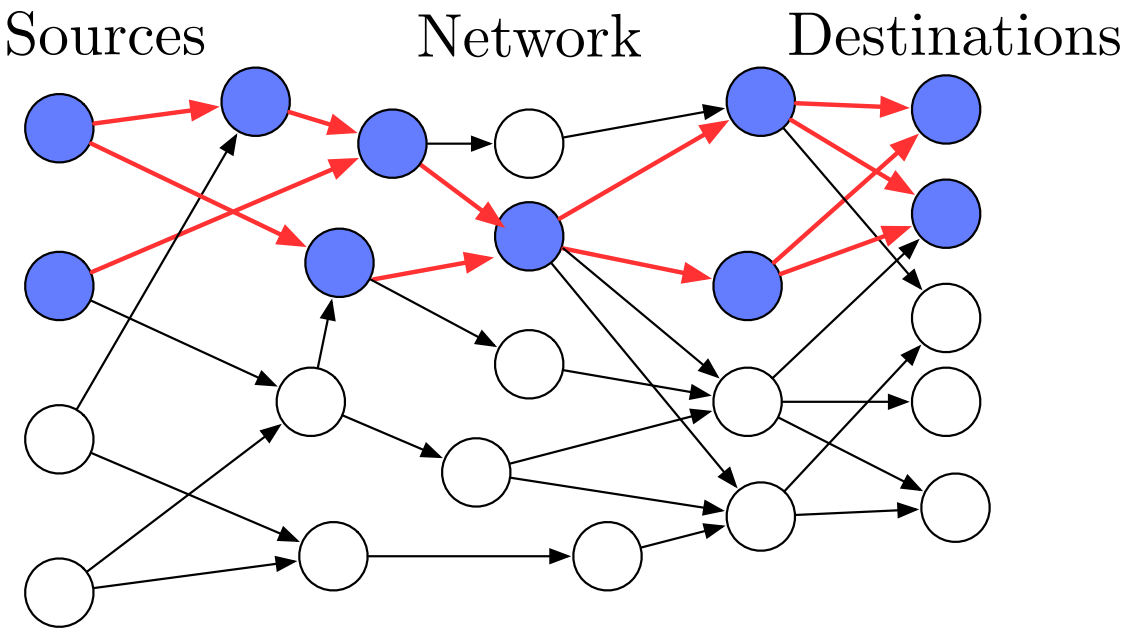
Motivated from matroid theory

- Inspired from matroid theory: forbidden minor for linear representability, e.g., $U_{2,4}$ is the forbidden minor for binary matroids



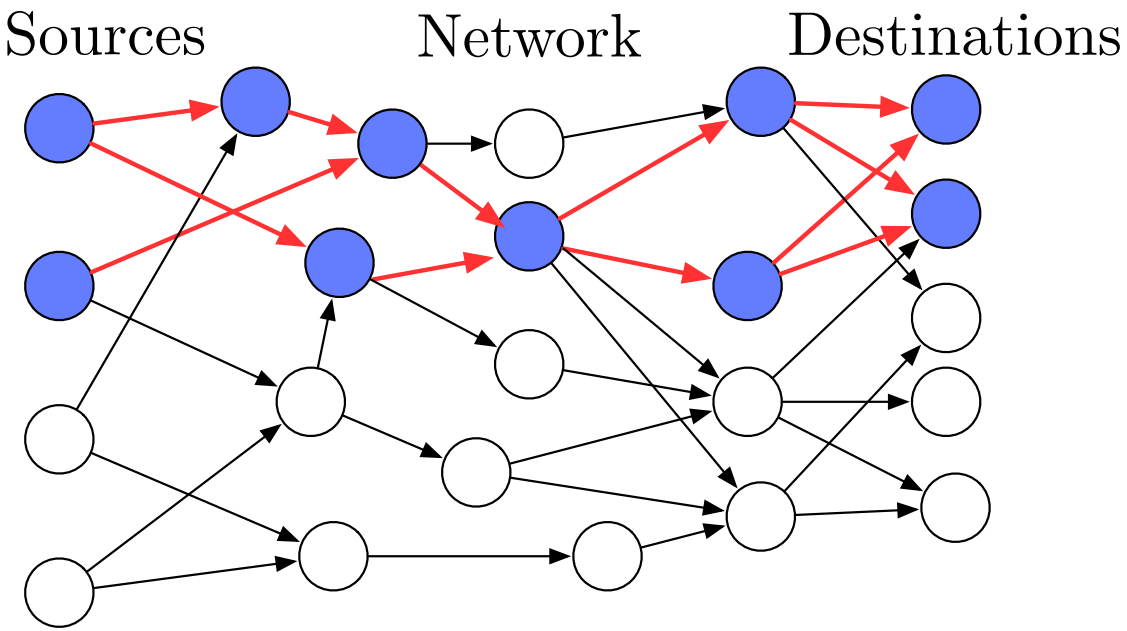
This work

- If networks have similar characterization? Possible list of forbidden embedded networks for sufficiency of linear codes over a field.
- Network operations to obtain such embedded networks preserving sufficiency.



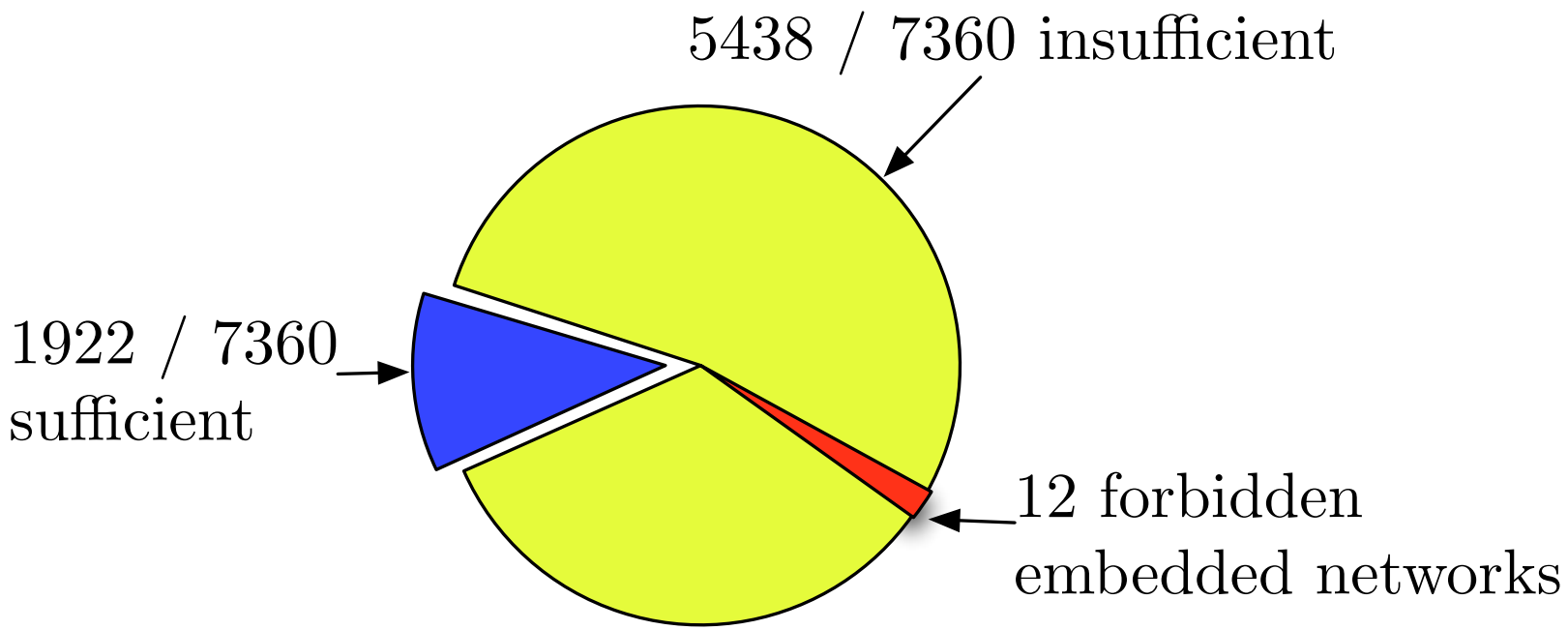
This work

- If networks have similar characterization? Possible list of forbidden embedded networks for sufficiency of linear codes over a field.
- Network operations to obtain such embedded networks preserving sufficiency.



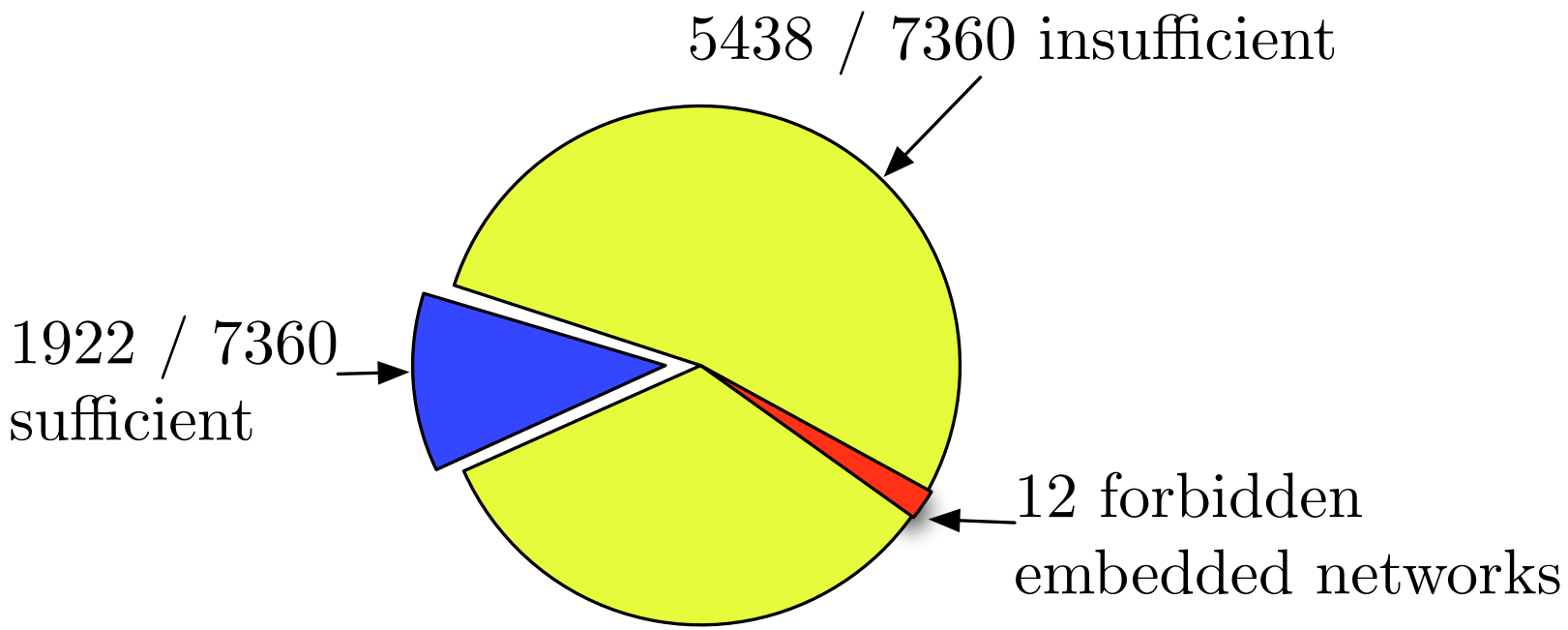
Preview of results

- Three operations
- Scalar binary codes considered
- 12 forbidden embedded networks for 7360 non-isomorphic instances.



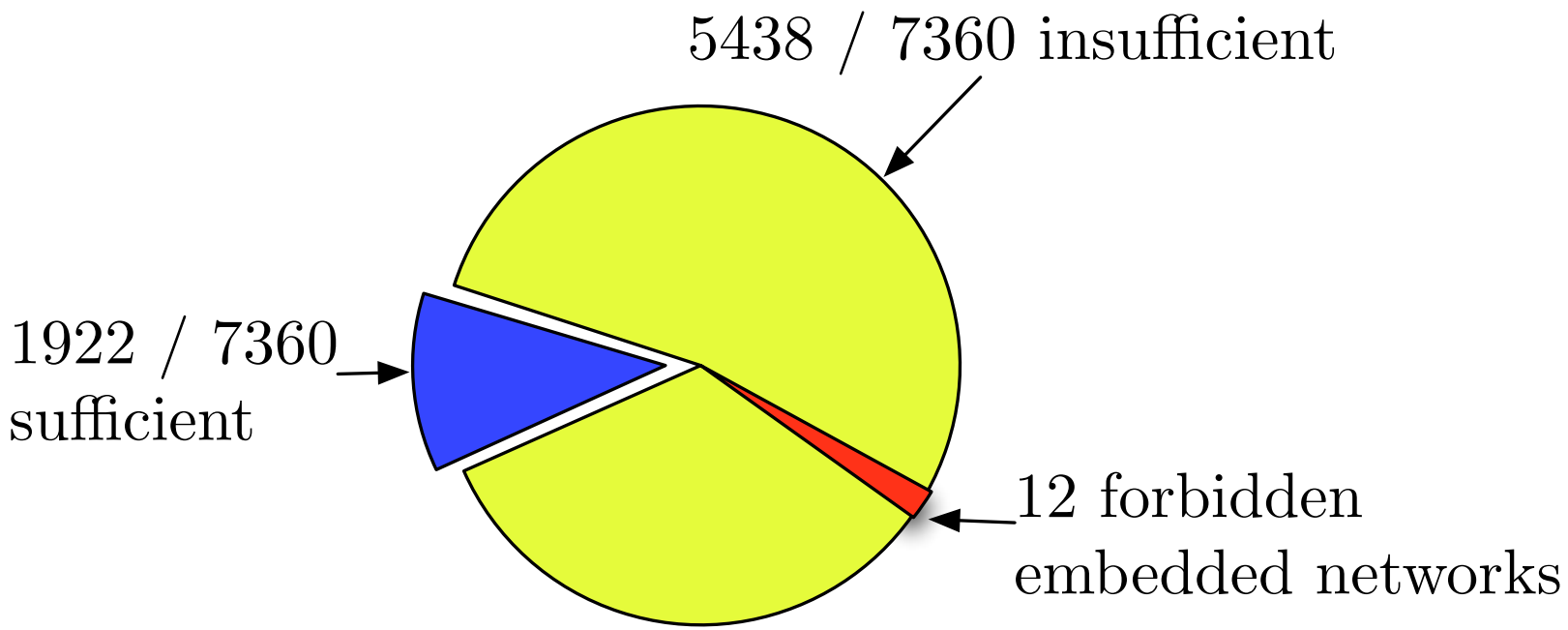
Preview of results

- Three operations
- Scalar binary codes considered
- 12 forbidden embedded networks for 7360 non-isomorphic instances.



Preview of results

- Three operations
- Scalar binary codes considered
- 12 forbidden embedded networks for 7360 non-isomorphic instances.

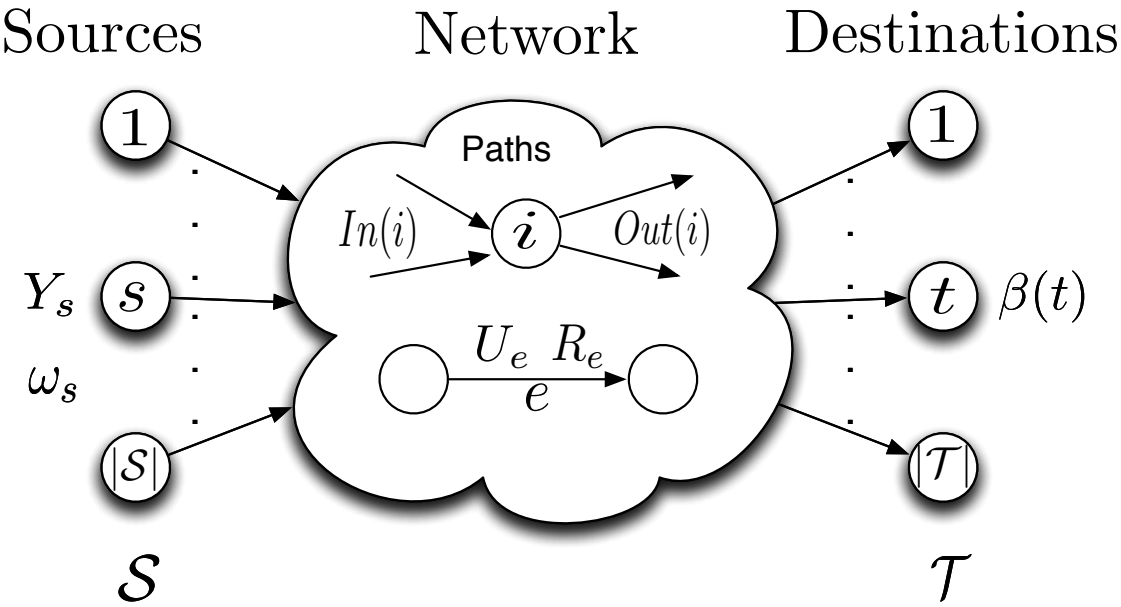


Outline

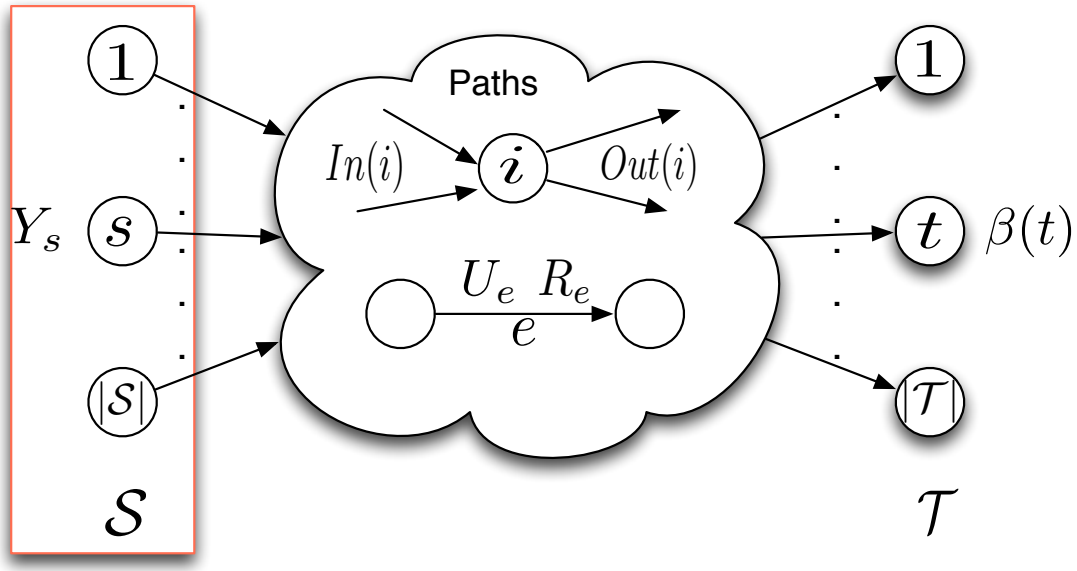
1. Motivation
2. Background on Rate regions and codes
3. Network operations
   - Source deletion
   - Edge deletion
   - Edge contraction
4. MDCS demonstration
   - MDCS model
   - Corresponding operations
   - Forbidden embedded networks for scalar binary codes

Multi-terminal network model

- Multi-terminal network model: directed acyclic graph

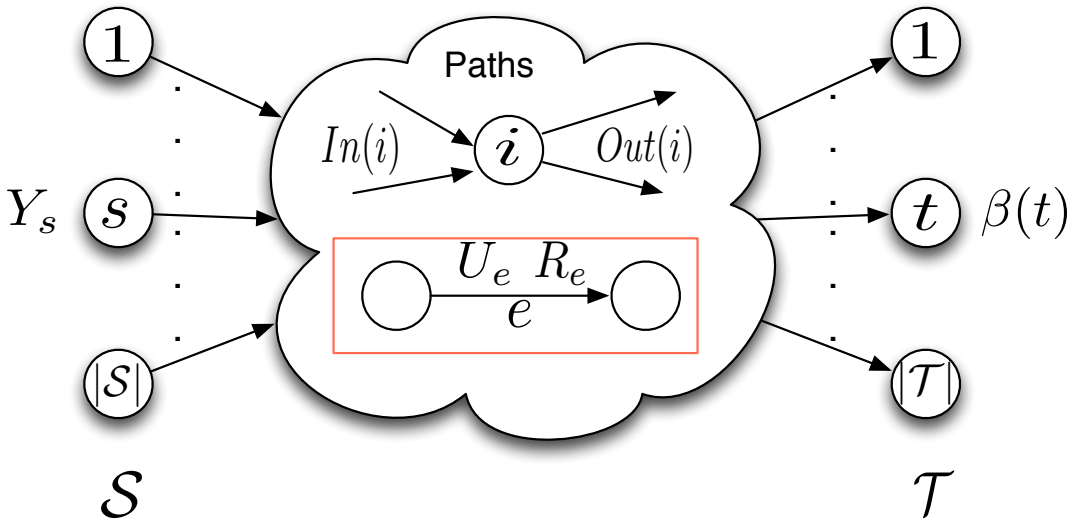


Multi-terminal network model: sources



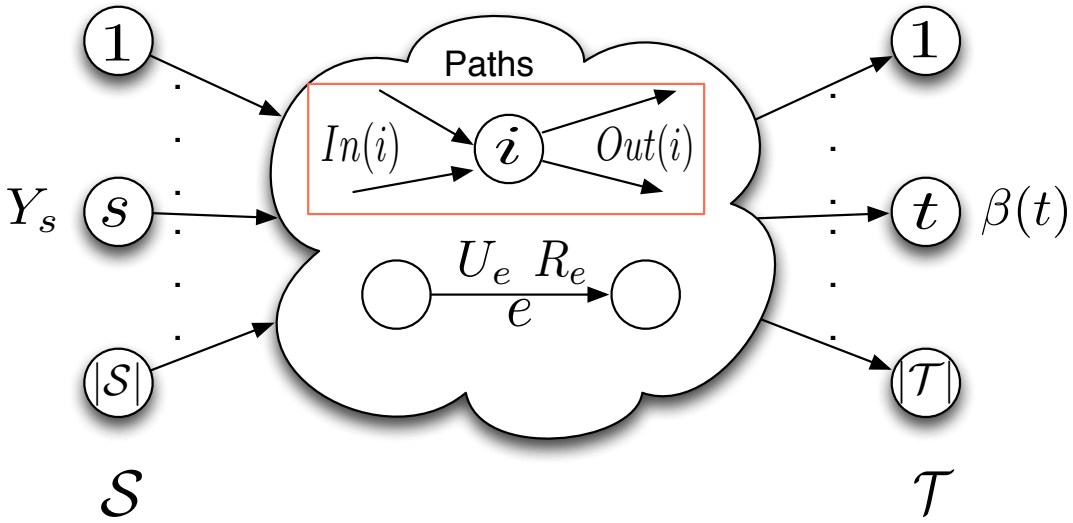
Source Rate	$H(Y_s)$	$\geq$	ω_s
Source Independence	$H(Y_{\mathcal{S}})$	$=$	$\sum_{s\in\mathcal{S}} H(Y_s)$
Source Encoding	$H(U_{\mathrm{Out}(s)}\|Y_s)$	$=$	0

Multi-terminal network model: edges



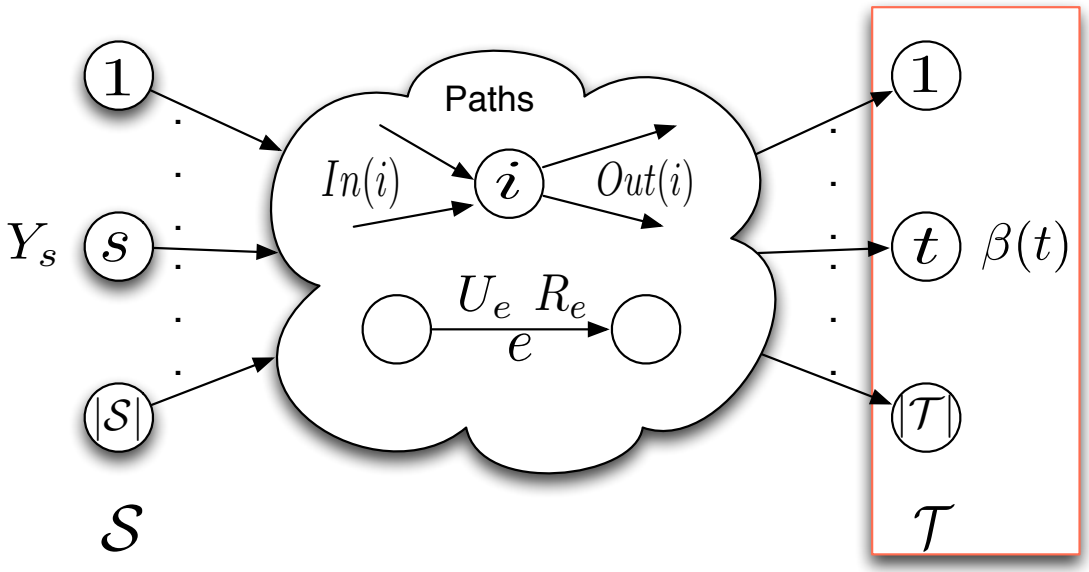
Coding Rate $R_e \geq H(U_e),\ e \in \mathcal{E}$

Multi-terminal network model: intermediate nodes



Coding Constraints $H(U_{\mathrm{Out}(i)}|U_{\mathrm{In}(i)}) = 0,\ i \in \mathcal{V}\backslash(\mathcal{S} \cup \mathcal{T})$

Multi-terminal network model: sinks



Decoding Constraints $H(Y_{\beta(t)}|U_{\mathrm{In}(t)}) = 0,\ t \in \mathcal{T}$

Rate region

- Rate region: all possible rate and source entropy vectors satisfying all network constraints.
- Collect the random variables and consider their (joint) entropies.
- Define Γ_N^*: region of valid entropy vectors.
- Constraints from network A:

$$\mathcal{L}_1 = \{\mathbf{h} \in \Gamma_N^* : h_{Y_S} = \Sigma_{s \in \mathcal{S}} h_{Y_s}\} \quad (1)$$

$$\mathcal{L}_2 = \{\mathbf{h} \in \Gamma_N^* : h_{X_{\mathrm{Out}(k)}|Y_s} = 0\} \quad (2)$$

$$\mathcal{L}_3 = \{\mathbf{h} \in \Gamma_N^* : h_{X_{\mathrm{Out}(i)}|X_{\mathrm{In}(i)}} = 0\} \quad (3)$$

$$\mathcal{L}_4 = \{(\mathbf{h}^T, \mathbf{R}^T)^T \in \mathbb{R}_+^{2^N-1+|\mathcal{E}|} : R_e \geq h_{U_e}, e \in \mathcal{E}\} \quad (4)$$

$$\mathcal{L}_5 = \{\mathbf{h} \in \Gamma_N^* : h_{Y_{\beta(t)}|U_{\mathrm{In}(t)}} = 0\}. \quad (5)$$

- Rate region in terms of rates and source entropies (derived from [2](Yan, Yeung, Zhang TranIT 2012)):

$$\mathcal{R}(\mathrm{A}) = \mathrm{proj}_{R_\mathcal{E}, H(Y_S)}(\overline{\mathrm{con}(\Gamma_N^* \cap \mathcal{L}_{123})} \cap \mathcal{L}_{45}) \quad (6)$$

Rate region

- Rate region: all possible rate and source entropy vectors satisfying all network constraints.
- Collect the random variables and consider their (joint) entropies.
- Define Γ_N^*: region of valid entropy vectors.
- Constraints from network A:

$$\mathcal{L}_1 = \{\mathbf{h} \in \Gamma_N^* : h_{Y_S} = \Sigma_{s\in\mathcal{S}} h_{Y_s}\} \quad (1)$$

$$\mathcal{L}_2 = \{\mathbf{h} \in \Gamma_N^* : h_{X_{\mathrm{Out}(k)}|Y_s} = 0\} \quad (2)$$

$$\mathcal{L}_3 = \{\mathbf{h} \in \Gamma_N^* : h_{X_{\mathrm{Out}(i)}|X_{\mathrm{In}(i)}} = 0\} \quad (3)$$

$$\mathcal{L}_4 = \{(\mathbf{h}^T, \mathbf{R}^T)^T \in \mathbb{R}_+^{2^N-1+|\mathcal{E}|} : R_e \geq h_{U_e}, e \in \mathcal{E}\} \quad (4)$$

$$\mathcal{L}_5 = \{\mathbf{h} \in \Gamma_N^* : h_{Y_{\beta(t)}|U_{\mathrm{In}(t)}} = 0\}. \quad (5)$$

- Rate region in terms of rates and source entropies (derived from [2](Yan, Yeung, Zhang TranIT 2012)):

$$\mathcal{R}(\mathrm{A}) = \mathrm{proj}_{R_\mathcal{E}, H(Y_S)}(\overline{\mathrm{con}(\Gamma_N^* \cap \mathcal{L}_{123})} \cap \mathcal{L}_{45}) \quad (6)$$

Rate region

- Rate region: all possible rate and source entropy vectors satisfying all network constraints.
- Collect the random variables and consider their (joint) entropies.
- Define Γ_N^*: region of valid entropy vectors.
- Constraints from network A:

$$\mathcal{L}_1 = \{\mathbf{h} \in \Gamma_N^* : h_{Y_S} = \Sigma_{s\in\mathcal{S}} h_{Y_s}\} \quad (1)$$

$$\mathcal{L}_2 = \{\mathbf{h} \in \Gamma_N^* : h_{X_{\mathrm{Out}(k)}|Y_s} = 0\} \quad (2)$$

$$\mathcal{L}_3 = \{\mathbf{h} \in \Gamma_N^* : h_{X_{\mathrm{Out}(i)}|X_{\mathrm{In}(i)}} = 0\} \quad (3)$$

$$\mathcal{L}_4 = \{(\mathbf{h}^T, \mathbf{R}^T)^T \in \mathbb{R}_+^{2^N-1+|\mathcal{E}|} : R_e \geq h_{U_e}, e \in \mathcal{E}\} \quad (4)$$

$$\mathcal{L}_5 = \{\mathbf{h} \in \Gamma_N^* : h_{Y_{\beta(t)}|U_{\mathrm{In}(t)}} = 0\}. \quad (5)$$

- Rate region in terms of rates and source entropies (derived from [2](Yan, Yeung, Zhang TranIT 2012)):

$$\mathcal{R}(\mathrm{A}) = \mathrm{proj}_{R_\mathcal{E}, H(Y_\mathcal{S})}(\overline{\mathrm{con}(\Gamma_N^* \cap \mathcal{L}_{123})} \cap \mathcal{L}_{45}) \quad (6)$$

Rate region

- Rate region: all possible rate and source entropy vectors satisfying all network constraints.
- Collect the random variables and consider their (joint) entropies.
- Define Γ_N^*: region of valid entropy vectors.
- Constraints from network A:

$$\mathcal{L}_1 = \{\mathbf{h} \in \Gamma_N^* : h_{Y_S} = \Sigma_{s\in\mathcal{S}} h_{Y_s}\} \quad (1)$$

$$\mathcal{L}_2 = \{\mathbf{h} \in \Gamma_N^* : h_{X_{\mathrm{Out}(k)}|Y_s} = 0\} \quad (2)$$

$$\mathcal{L}_3 = \{\mathbf{h} \in \Gamma_N^* : h_{X_{\mathrm{Out}(i)}|X_{\mathrm{In}(i)}} = 0\} \quad (3)$$

$$\mathcal{L}_4 = \{(\mathbf{h}^T, \mathbf{R}^T)^T \in \mathbb{R}_+^{2^N-1+|\mathcal{E}|} : R_e \geq h_{U_e}, e \in \mathcal{E}\} \quad (4)$$

$$\mathcal{L}_5 = \{\mathbf{h} \in \Gamma_N^* : h_{Y_{\beta(t)}|U_{\mathrm{In}(t)}} = 0\}. \quad (5)$$

- Rate region in terms of rates and source entropies (derived from [2](Yan, Yeung, Zhang TranIT 2012)):

$$\mathcal{R}(\mathrm{A}) = \mathrm{proj}_{R_\mathcal{E}, H(Y_S)}(\overline{\mathrm{con}(\Gamma_N^* \cap \mathcal{L}_{123})} \cap \mathcal{L}_{45}) \quad (6)$$

Rate region

- Rate region: all possible rate and source entropy vectors satisfying all network constraints.
- Collect the random variables and consider their (joint) entropies.
- Define Γ_N^*: region of valid entropy vectors.
- Constraints from network A:

$$\mathcal{L}_1 = \{\mathbf{h} \in \Gamma_N^* : h_{Y_\mathcal{S}} = \Sigma_{s\in\mathcal{S}} h_{Y_s}\} \quad (1)$$

$$\mathcal{L}_2 = \{\mathbf{h} \in \Gamma_N^* : h_{X_{\mathrm{Out}(k)}|Y_s} = 0\} \quad (2)$$

$$\mathcal{L}_3 = \{\mathbf{h} \in \Gamma_N^* : h_{X_{\mathrm{Out}(i)}|X_{\mathrm{In}(i)}} = 0\} \quad (3)$$

$$\mathcal{L}_4 = \{(\mathbf{h}^T, \mathbf{R}^T)^T \in \mathbb{R}_+^{2^N-1+|\mathcal{E}|} : R_e \geq h_{U_e}, e \in \mathcal{E}\} \quad (4)$$

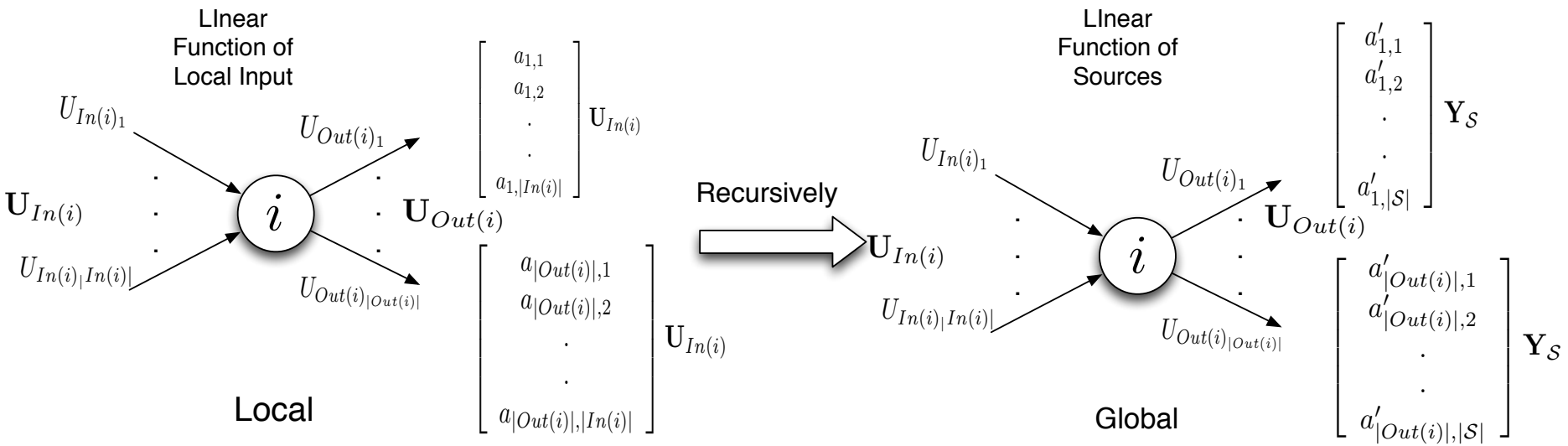
$$\mathcal{L}_5 = \{\mathbf{h} \in \Gamma_N^* : h_{Y_{\beta(t)}|U_{\mathrm{In}(t)}} = 0\}. \quad (5)$$

- Rate region in terms of rates and source entropies (derived from [2](Yan, Yeung, Zhang TranIT 2012)):

$$\mathcal{R}(\mathrm{A}) = \mathrm{proj}_{R_\mathcal{E}, H(Y_\mathcal{S})}(\overline{\mathrm{con}(\Gamma_N^* \cap \mathcal{L}_{123})} \cap \mathcal{L}_{45}) \quad (6)$$

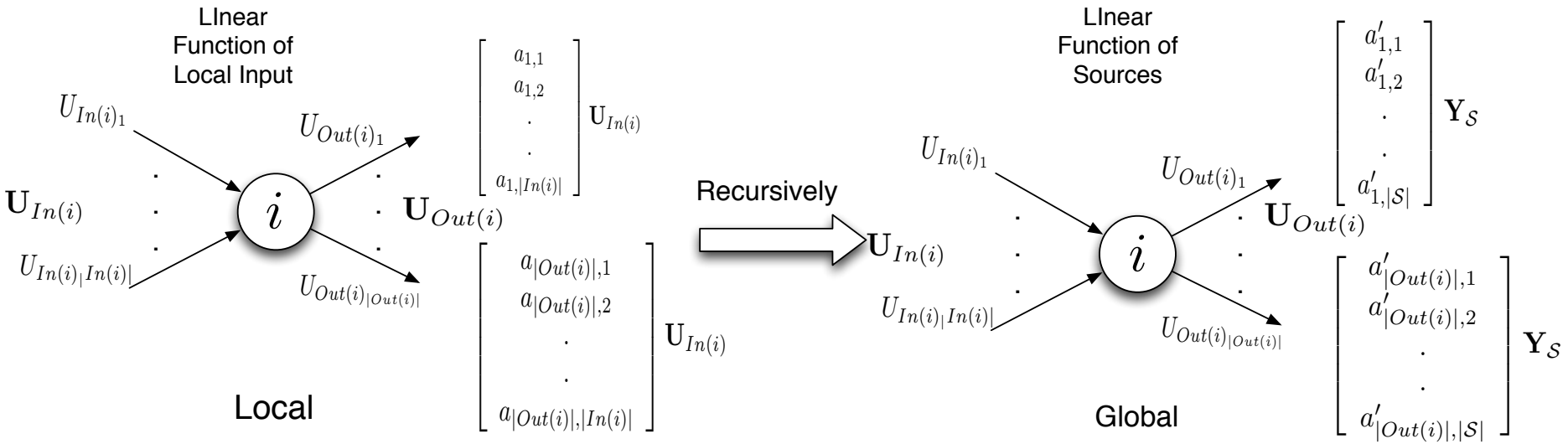
Linear Network Coding

- Local linear function: output is linear function of input
- Global linear function: output is linear function of sources
- In general, one variable, multi-columns: $\mathcal{R}_q$
- Special case, one variable, one-column: $\mathcal{R}_{s,q}$



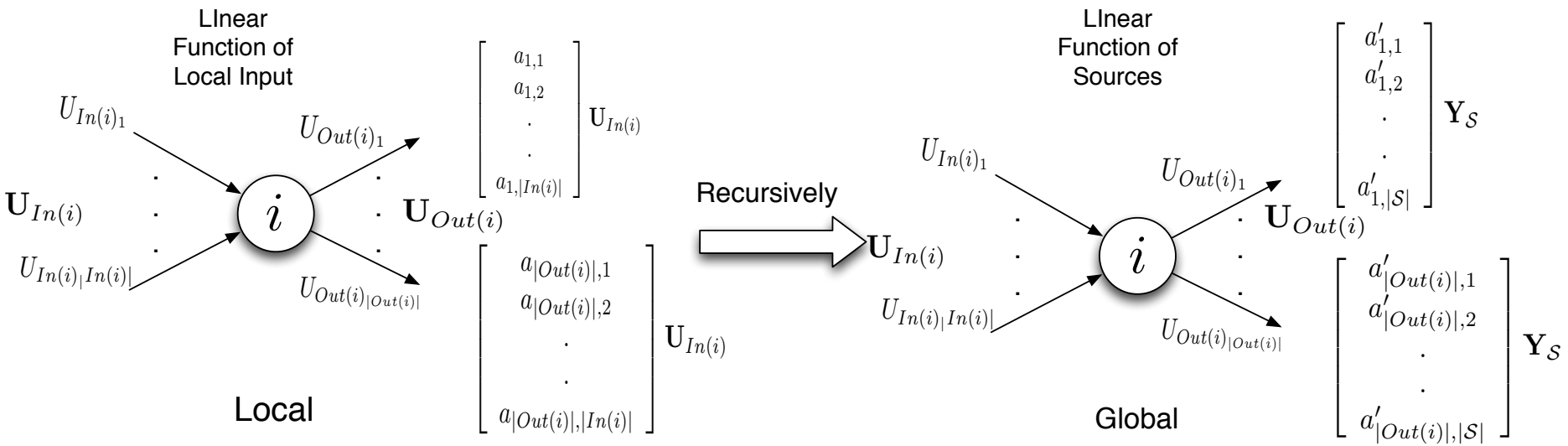
Linear Network Coding

- Local linear function: output is linear function of input
- Global linear function: output is linear function of sources
- In general, one variable, multi-columns: $\mathcal{R}_q$
- Special case, one variable, one-column: $\mathcal{R}_{s,q}$



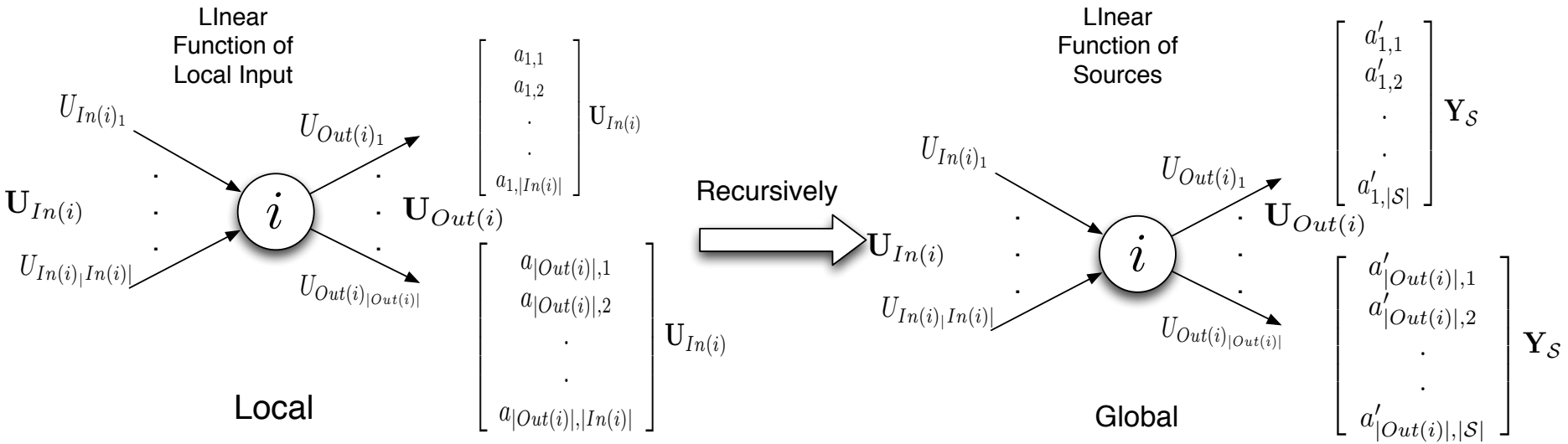
Linear Network Coding

- Local linear function: output is linear function of input
- Global linear function: output is linear function of sources
- In general, one variable, multi-columns: $\mathcal{R}_q$
- Special case, one variable, one-column: $\mathcal{R}_{s,q}$



Linear Network Coding

- Local linear function: output is linear function of input
- Global linear function: output is linear function of sources
- In general, one variable, multi-columns: $\mathcal{R}_q$
- Special case, one variable, one-column: $\mathcal{R}_{s,q}$

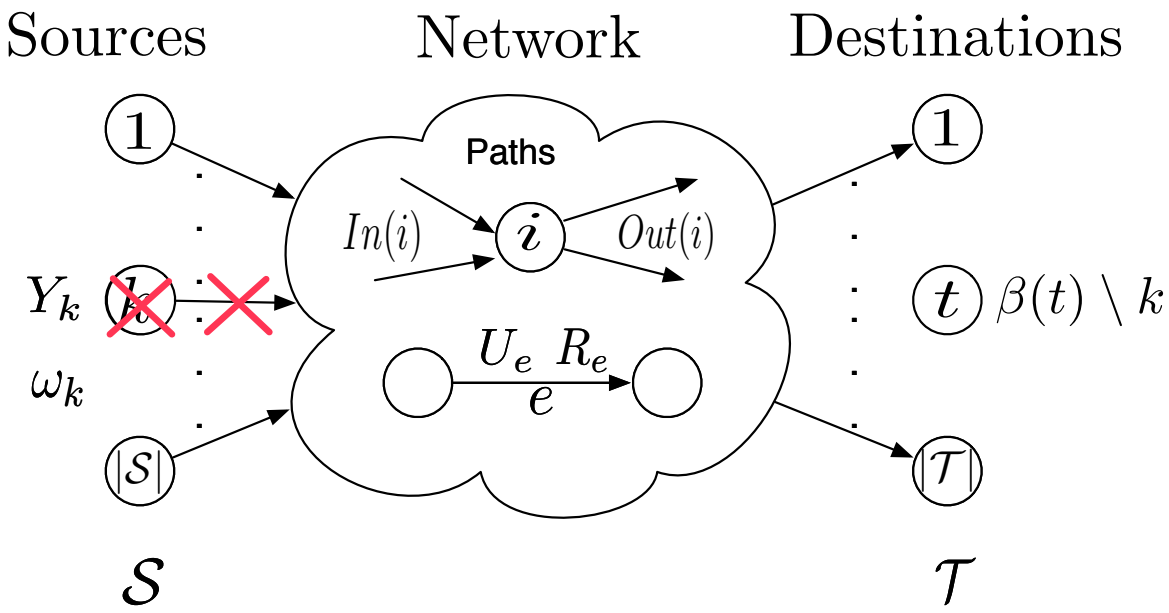


Outline

1. Motivation
2. Background on Rate regions and codes
3. Network operations
   - Source deletion
   - Edge deletion
   - Edge contraction
4. MDCS demonstration
   - MDCS model
   - Corresponding operations
   - Forbidden embedded networks for scalar binary codes

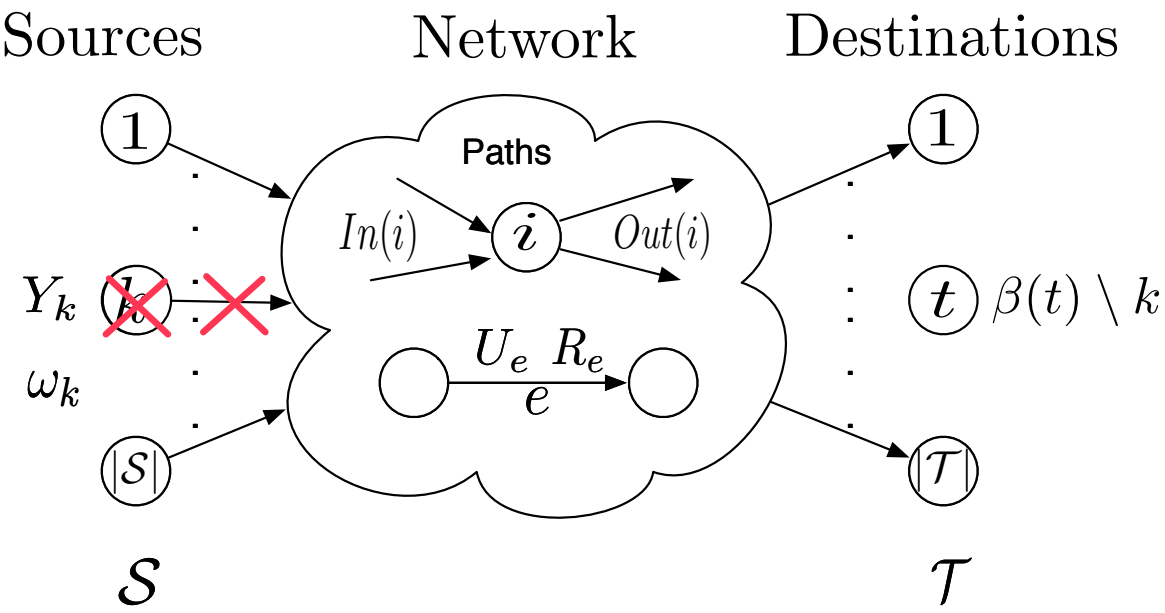
Source deletion

- $\mathsf{A}' = \mathsf{A} \setminus k$
- Source Y_k deleted, source k stops sending information to the network, $H(Y_k) = 0$
- Sinks requiring Y_k will no longer demand it.



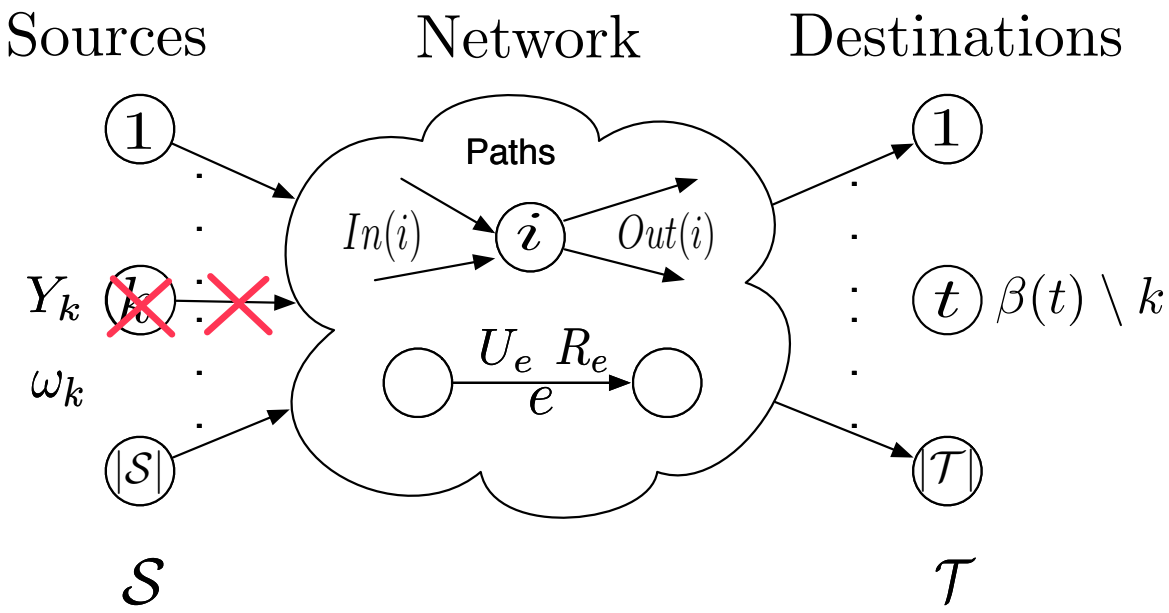
Source deletion

- $A' = A \setminus k$
- Source Y_k deleted, source k stops sending information to the network, $H(Y_k) = 0$
- Sinks requiring Y_k will no longer demand it.



Source deletion

- $\mathsf{A}' = \mathsf{A} \setminus k$
- Source Y_k deleted, source k stops sending information to the network, $H(Y_k) = 0$
- Sinks requiring Y_k will no longer demand it.



Source deletion: $\mathrm{A}' = \mathrm{A} \setminus k$

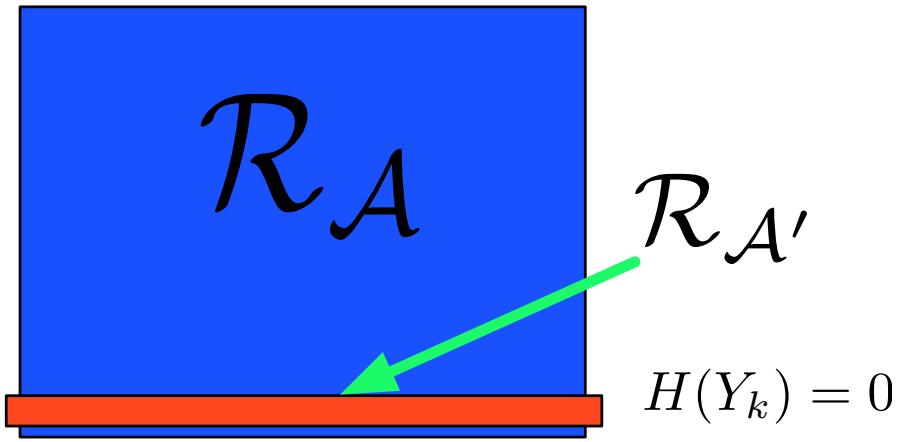
$$\mathcal{R}(\mathrm{A}') = \mathrm{Proj}_{\mathbf{Y}_{\setminus k}, \mathbf{R}_{\mathcal{E}'}} \left(\{ \mathbf{R} \in \mathcal{R}(\mathrm{A}) \,|\, H(Y_k) = 0 \} \right), \tag{7}$$

$$\mathcal{R}_q(\mathrm{A}') = \mathrm{Proj}_{\mathbf{Y}_{\setminus k}, \mathbf{R}_{\mathcal{E}'}} \left(\{ \mathbf{R} \in \mathcal{R}_q(\mathrm{A}) \,|\, H(Y_k) = 0 \} \right), \tag{8}$$

$$\mathcal{R}_{s,q}(\mathrm{A}') = \mathrm{Proj}_{\mathbf{Y}_{\setminus k}, \mathbf{R}_{\mathcal{E}'}} \left(\{ \mathbf{R} \in \mathcal{R}_{s,q}(\mathrm{A}) \,|\, H(Y_k) = 0 \} \right). \tag{9}$$

- Preservation: $\mathcal{R}_q(\mathrm{A}) = \mathcal{R}(\mathrm{A}) \Rightarrow \mathcal{R}_q(\mathrm{A}') = \mathcal{R}(\mathrm{A}')$
- Scalar case: $\mathcal{R}_{s,q}(\mathrm{A}) = \mathcal{R}(\mathrm{A}) \Rightarrow \mathcal{R}_{s,q}(\mathrm{A}') = \mathcal{R}(\mathrm{A}')$

- $\mathbf{r}' \in \mathcal{R}(\mathrm{A}') \leftrightarrow \mathbf{r} \in \mathcal{R}(\mathrm{A})$ with $H(Y_k) = 0$
- The code to achieve $\mathbf{r}'$ is the code to achieve $\mathbf{r}$ with sending nothing on source Y_k. All-zero is valid $\mathbb{F}_q$ code, can reverse.



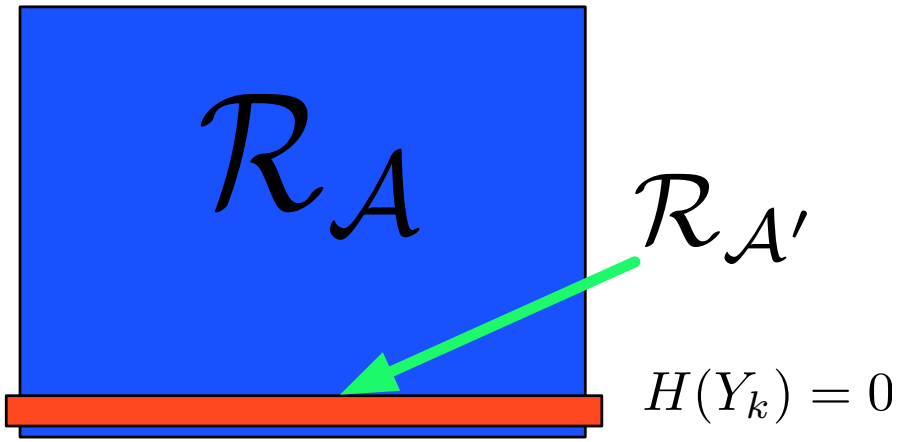
Source deletion: $\mathsf{A}' = \mathsf{A} \setminus k$

$$\mathcal{R}(\mathsf{A}') = \mathrm{Proj}_{\mathbf{Y}_{\setminus k}, \mathbf{R}_{\mathcal{E}'}} \left(\{ \mathbf{R} \in \mathcal{R}(\mathsf{A}) \,|\, H(Y_k) = 0 \} \right), \tag{7}$$

$$\mathcal{R}_q(\mathsf{A}') = \mathrm{Proj}_{\mathbf{Y}_{\setminus k}, \mathbf{R}_{\mathcal{E}'}} \left(\{ \mathbf{R} \in \mathcal{R}_q(\mathsf{A}) \,|\, H(Y_k) = 0 \} \right), \tag{8}$$

$$\mathcal{R}_{s,q}(\mathsf{A}') = \mathrm{Proj}_{\mathbf{Y}_{\setminus k}, \mathbf{R}_{\mathcal{E}'}} \left(\{ \mathbf{R} \in \mathcal{R}_{s,q}(\mathsf{A}) \,|\, H(Y_k) = 0 \} \right). \tag{9}$$

- Preservation: $\mathcal{R}_q(\mathsf{A}) = \mathcal{R}(\mathsf{A}) \Rightarrow \mathcal{R}_q(\mathsf{A}') = \mathcal{R}(\mathsf{A}')$
- Scalar case: $\mathcal{R}_{s,q}(\mathsf{A}) = \mathcal{R}(\mathsf{A}) \Rightarrow \mathcal{R}_{s,q}(\mathsf{A}') = \mathcal{R}(\mathsf{A}')$
- $\mathbf{r}' \in \mathcal{R}(\mathsf{A}') \leftrightarrow \mathbf{r} \in \mathcal{R}(\mathsf{A})$ with $H(Y_k) = 0$
- The code to achieve $\mathbf{r}'$ is the code to achieve $\mathbf{r}$ with sending nothing on source Y_k. All-zero is valid $\mathbb{F}_q$ code, can reverse.



Source deletion: $\mathsf{A}' = \mathsf{A} \setminus k$

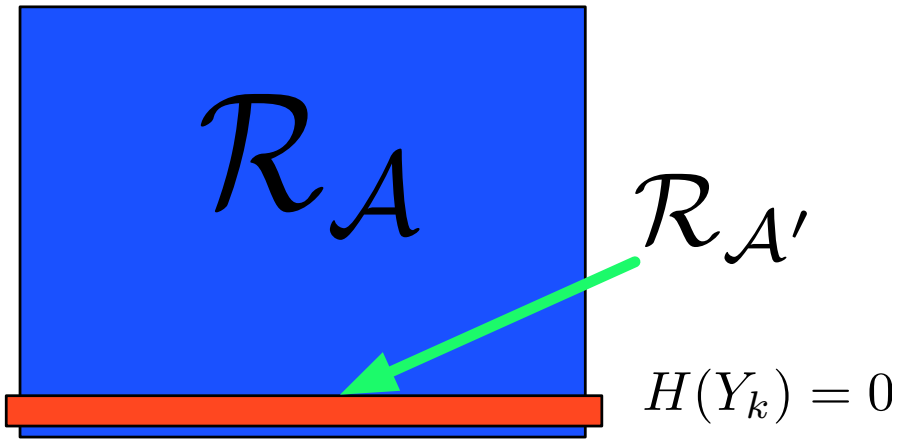
$$\mathcal{R}(\mathsf{A}') = \mathrm{Proj}_{\mathbf{Y}_{\setminus k}, \mathbf{R}_{\mathcal{E}'}} \left(\{ \mathbf{R} \in \mathcal{R}(\mathsf{A}) \,|\, H(Y_k) = 0 \} \right), \tag{7}$$

$$\mathcal{R}_q(\mathsf{A}') = \mathrm{Proj}_{\mathbf{Y}_{\setminus k}, \mathbf{R}_{\mathcal{E}'}} \left(\{ \mathbf{R} \in \mathcal{R}_q(\mathsf{A}) \,|\, H(Y_k) = 0 \} \right), \tag{8}$$

$$\mathcal{R}_{s,q}(\mathsf{A}') = \mathrm{Proj}_{\mathbf{Y}_{\setminus k}, \mathbf{R}_{\mathcal{E}'}} \left(\{ \mathbf{R} \in \mathcal{R}_{s,q}(\mathsf{A}) \,|\, H(Y_k) = 0 \} \right). \tag{9}$$

- Preservation: $\mathcal{R}_q(\mathsf{A}) = \mathcal{R}(\mathsf{A}) \Rightarrow \mathcal{R}_q(\mathsf{A}') = \mathcal{R}(\mathsf{A}')$
- Scalar case: $\mathcal{R}_{s,q}(\mathsf{A}) = \mathcal{R}(\mathsf{A}) \Rightarrow \mathcal{R}_{s,q}(\mathsf{A}') = \mathcal{R}(\mathsf{A}')$

- $\mathbf{r}' \in \mathcal{R}(\mathsf{A}') \leftrightarrow \mathbf{r} \in \mathcal{R}(\mathsf{A})$ with $H(Y_k) = 0$
- The code to achieve $\mathbf{r}'$ is the code to achieve $\mathbf{r}$ with sending nothing on source Y_k. All-zero is valid $\mathbb{F}_q$ code, can reverse.



Source deletion: $\mathrm{A}' = \mathrm{A} \setminus k$

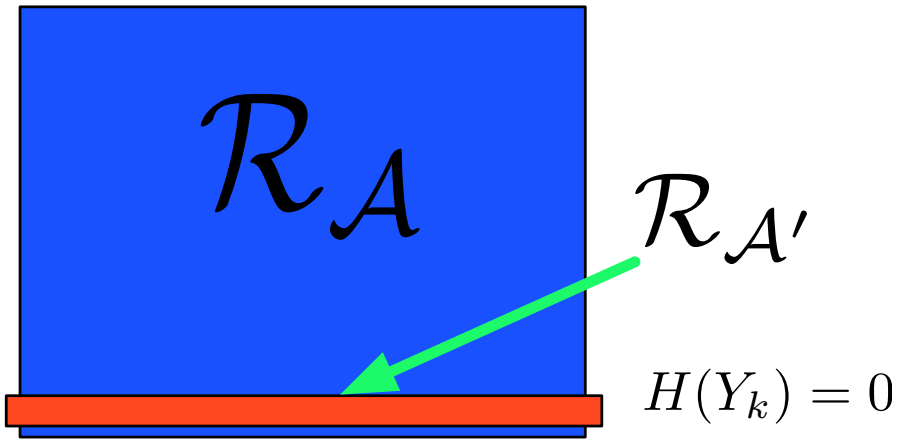
$$\mathcal{R}(\mathrm{A}') = \mathrm{Proj}_{\mathbf{Y}_{\setminus k}, \mathbf{R}_{\mathcal{E}'}} \left(\{ \mathbf{R} \in \mathcal{R}(\mathrm{A}) \,|\, H(Y_k) = 0 \} \right), \tag{7}$$

$$\mathcal{R}_q(\mathrm{A}') = \mathrm{Proj}_{\mathbf{Y}_{\setminus k}, \mathbf{R}_{\mathcal{E}'}} \left(\{ \mathbf{R} \in \mathcal{R}_q(\mathrm{A}) \,|\, H(Y_k) = 0 \} \right), \tag{8}$$

$$\mathcal{R}_{s,q}(\mathrm{A}') = \mathrm{Proj}_{\mathbf{Y}_{\setminus k}, \mathbf{R}_{\mathcal{E}'}} \left(\{ \mathbf{R} \in \mathcal{R}_{s,q}(\mathrm{A}) \,|\, H(Y_k) = 0 \} \right). \tag{9}$$

- Preservation: $\mathcal{R}_q(\mathrm{A}) = \mathcal{R}(\mathrm{A}) \Rightarrow \mathcal{R}_q(\mathrm{A}') = \mathcal{R}(\mathrm{A}')$
- Scalar case: $\mathcal{R}_{s,q}(\mathrm{A}) = \mathcal{R}(\mathrm{A}) \Rightarrow \mathcal{R}_{s,q}(\mathrm{A}') = \mathcal{R}(\mathrm{A}')$

- $\mathbf{r}' \in \mathcal{R}(\mathrm{A}') \leftrightarrow \mathbf{r} \in \mathcal{R}(\mathrm{A})$ with $H(Y_k) = 0$
- The code to achieve $\mathbf{r}'$ is the code to achieve $\mathbf{r}$ with sending nothing on source Y_k. All-zero is valid $\mathbb{F}_q$ code, can reverse.



Source deletion: $\mathrm{A}' = \mathrm{A} \setminus k$

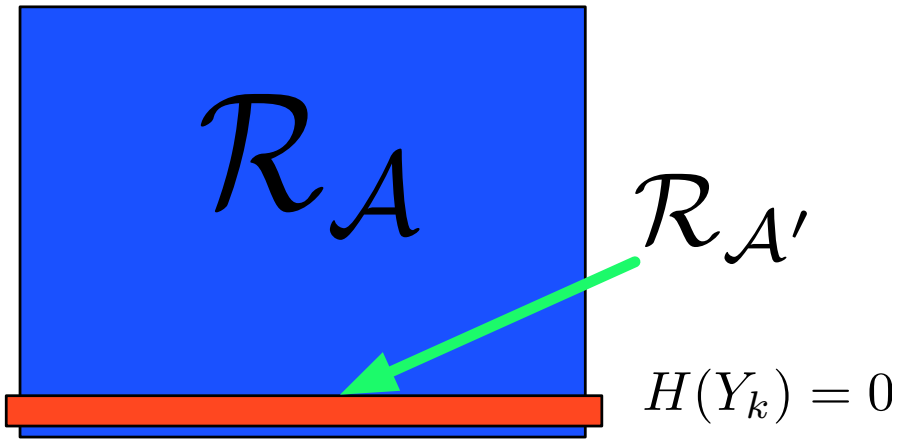
$$\mathcal{R}(\mathrm{A}') = \mathrm{Proj}_{\mathbf{Y}_{\setminus k}, \mathbf{R}_{\mathcal{E}'}} \left(\{ \mathbf{R} \in \mathcal{R}(\mathrm{A}) \,|\, H(Y_k) = 0 \} \right), \tag{7}$$

$$\mathcal{R}_q(\mathrm{A}') = \mathrm{Proj}_{\mathbf{Y}_{\setminus k}, \mathbf{R}_{\mathcal{E}'}} \left(\{ \mathbf{R} \in \mathcal{R}_q(\mathrm{A}) \,|\, H(Y_k) = 0 \} \right), \tag{8}$$

$$\mathcal{R}_{s,q}(\mathrm{A}') = \mathrm{Proj}_{\mathbf{Y}_{\setminus k}, \mathbf{R}_{\mathcal{E}'}} \left(\{ \mathbf{R} \in \mathcal{R}_{s,q}(\mathrm{A}) \,|\, H(Y_k) = 0 \} \right). \tag{9}$$

- Preservation: $\mathcal{R}_q(\mathrm{A}) = \mathcal{R}(\mathrm{A}) \Rightarrow \mathcal{R}_q(\mathrm{A}') = \mathcal{R}(\mathrm{A}')$
- Scalar case: $\mathcal{R}_{s,q}(\mathrm{A}) = \mathcal{R}(\mathrm{A}) \Rightarrow \mathcal{R}_{s,q}(\mathrm{A}') = \mathcal{R}(\mathrm{A}')$

- $\mathbf{r}' \in \mathcal{R}(\mathrm{A}') \leftrightarrow \mathbf{r} \in \mathcal{R}(\mathrm{A})$ with $H(Y_k) = 0$
- The code to achieve $\mathbf{r}'$ is the code to achieve $\mathbf{r}$ with sending nothing on source Y_k. All-zero is valid $\mathbb{F}_q$ code, can reverse.

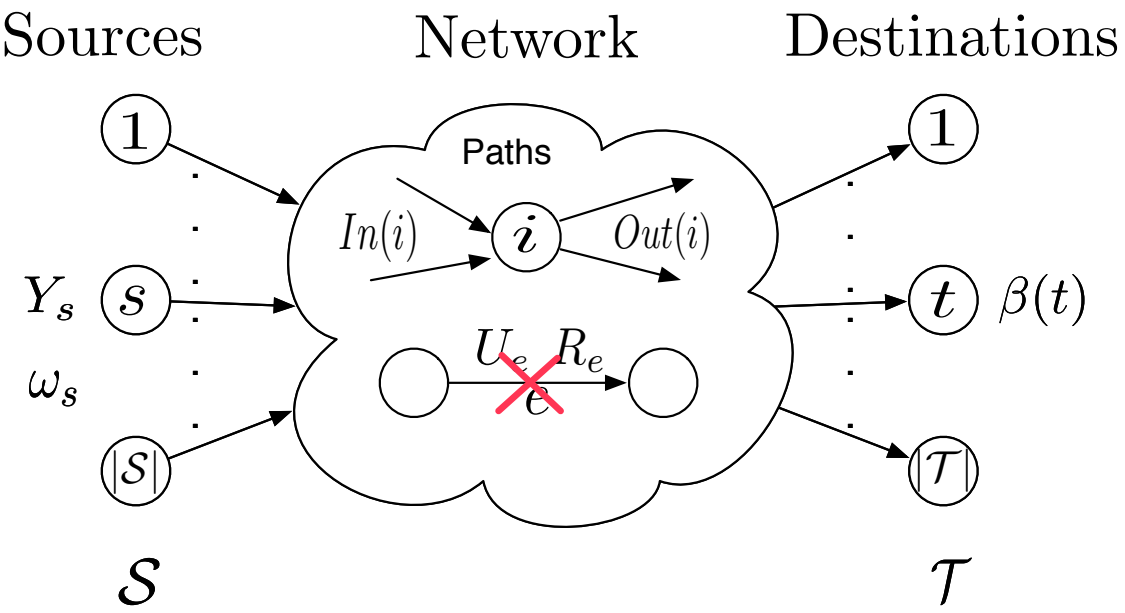


Outline

1. Motivation
2. Background on Rate regions and codes
3. Network operations
   - Source deletion
   - Edge deletion
   - Edge contraction
4. MDCS demonstration
   - MDCS model
   - Corresponding operations
   - Forbidden embedded networks for scalar binary codes

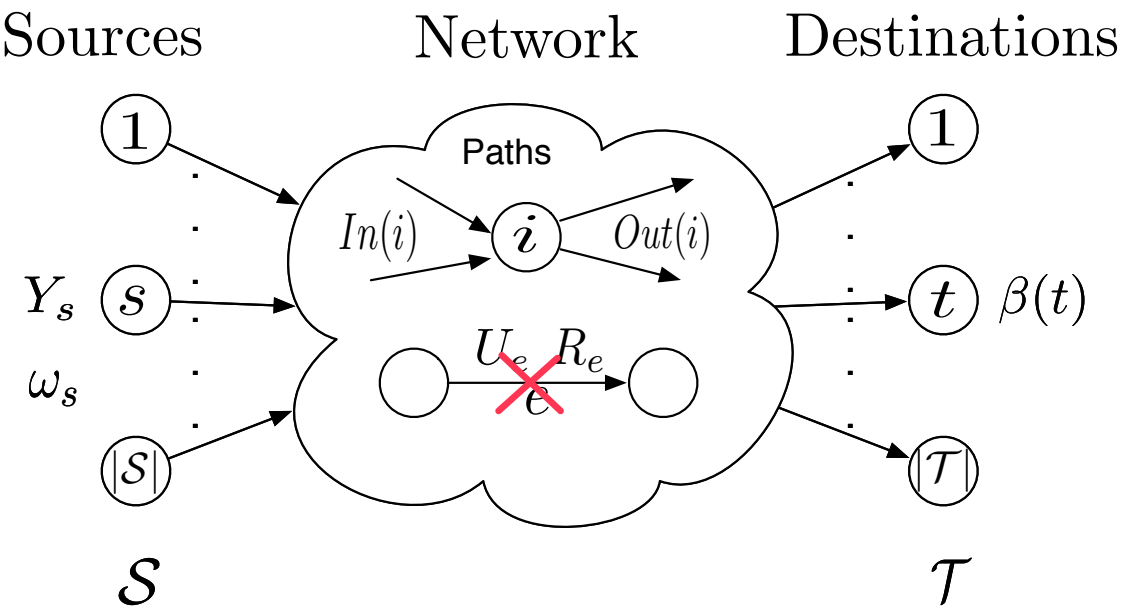
Edge deletion

- $\mathsf{A}' = \mathsf{A} \setminus e$
- Edge U_e deleted, nothing on U_e, $R_e = H(U_e) = 0$.



Edge deletion

- $A' = A \setminus e$
- Edge U_e deleted, nothing on U_e, $R_e = H(U_e) = 0$.



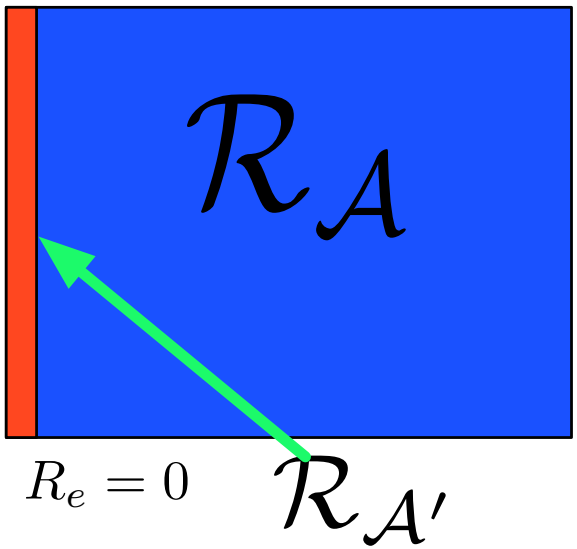
Edge deletion: $\mathsf{A}' = \mathsf{A} \setminus Y_k$

$$\mathcal{R}(\mathsf{A}') = \mathrm{Proj}_{H(Y_S),\mathbf{R}_{\mathcal{E}'}}(\{\mathbf{R} \in \mathcal{R}_q(\mathsf{A}) | R_e = 0\}), \quad (10)$$

$$\mathcal{R}_q(\mathsf{A}') = \mathrm{Proj}_{H(Y_S),\mathbf{R}_{\mathcal{E}'}}(\{\mathbf{R} \in \mathcal{R}_q(\mathsf{A}) | R_e = 0\}), \quad (11)$$

$$\mathcal{R}_{s,q}(\mathsf{A}') = \mathrm{Proj}_{H(Y_S),\mathbf{R}_{\mathcal{E}'}}(\{\mathbf{R} \in \mathcal{R}_{s,q}(\mathsf{A}) | R_e = 0\}). \quad (12)$$

- Preservation: $\mathcal{R}_q(\mathsf{A}) = \mathcal{R}(\mathsf{A}) \Rightarrow \mathcal{R}_q(\mathsf{A}') = \mathcal{R}(\mathsf{A}')$
- Scalar case: $\mathcal{R}_{s,q}(\mathsf{A}) = \mathcal{R}(\mathsf{A}) \Rightarrow \mathcal{R}_{s,q}(\mathsf{A}') = \mathcal{R}(\mathsf{A}')$
- $\mathbf{r}' \in \mathcal{R}(\mathsf{A}') \leftrightarrow \mathbf{r} \in \mathcal{R}(\mathsf{A})$ with $R_e = H(U_e) = 0$
- The code to achieve $\mathbf{r}'$ is the code to achieve $\mathbf{r}$ with deleting column(s) for U_e, i.e., sending nothing on edge e. All-zero is valid $\mathbb{F}_q$ code, can reverse.



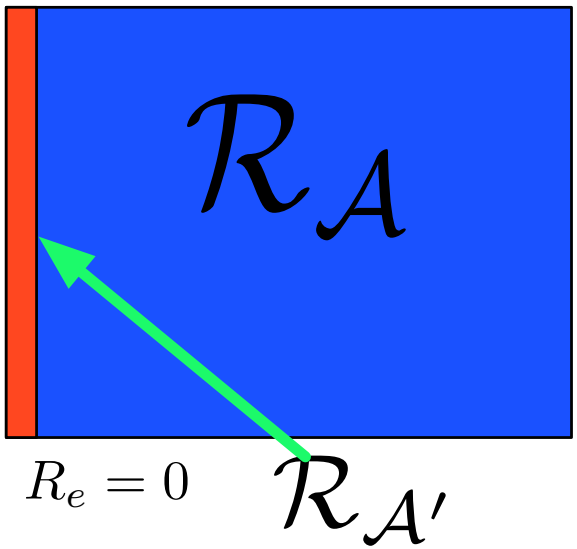
Edge deletion: $\mathsf{A}' = \mathsf{A} \setminus Y_k$

$$\mathcal{R}(\mathsf{A}') = \mathrm{Proj}_{H(Y_S),\mathbf{R}_{\mathcal{E}'}}(\{\mathbf{R} \in \mathcal{R}_q(\mathsf{A}) | R_e = 0\}), \quad (10)$$

$$\mathcal{R}_q(\mathsf{A}') = \mathrm{Proj}_{H(Y_S),\mathbf{R}_{\mathcal{E}'}}(\{\mathbf{R} \in \mathcal{R}_q(\mathsf{A}) | R_e = 0\}), \quad (11)$$

$$\mathcal{R}_{s,q}(\mathsf{A}') = \mathrm{Proj}_{H(Y_S),\mathbf{R}_{\mathcal{E}'}}(\{\mathbf{R} \in \mathcal{R}_{s,q}(\mathsf{A}) | R_e = 0\}). \quad (12)$$

- Preservation: $\mathcal{R}_q(\mathsf{A}) = \mathcal{R}(\mathsf{A}) \Rightarrow \mathcal{R}_q(\mathsf{A}') = \mathcal{R}(\mathsf{A}')$
- Scalar case: $\mathcal{R}_{s,q}(\mathsf{A}) = \mathcal{R}(\mathsf{A}) \Rightarrow \mathcal{R}_{s,q}(\mathsf{A}') = \mathcal{R}(\mathsf{A}')$
- $\mathbf{r}' \in \mathcal{R}(\mathsf{A}') \leftrightarrow \mathbf{r} \in \mathcal{R}(\mathsf{A})$ with $R_e = H(U_e) = 0$
- The code to achieve $\mathbf{r}'$ is the code to achieve $\mathbf{r}$ with deleting column(s) for U_e, i.e., sending nothing on edge e. All-zero is valid $\mathbb{F}_q$ code, can reverse.



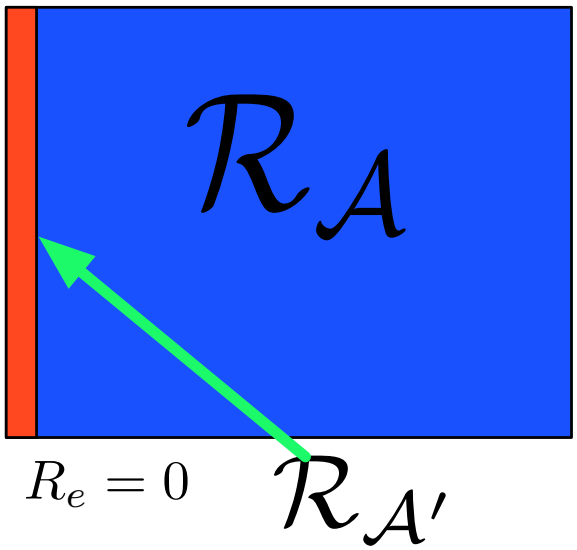
Edge deletion: $\mathsf{A}' = \mathsf{A} \setminus Y_k$

$$\mathcal{R}(\mathsf{A}') = \mathrm{Proj}_{H(Y_S),\mathbf{R}_{\mathcal{E}'}}(\{\mathbf{R} \in \mathcal{R}_q(\mathsf{A}) | R_e = 0\}), \quad (10)$$

$$\mathcal{R}_q(\mathsf{A}') = \mathrm{Proj}_{H(Y_S),\mathbf{R}_{\mathcal{E}'}}(\{\mathbf{R} \in \mathcal{R}_q(\mathsf{A}) | R_e = 0\}), \quad (11)$$

$$\mathcal{R}_{s,q}(\mathsf{A}') = \mathrm{Proj}_{H(Y_S),\mathbf{R}_{\mathcal{E}'}}(\{\mathbf{R} \in \mathcal{R}_{s,q}(\mathsf{A}) | R_e = 0\}). \quad (12)$$

- Preservation: $\mathcal{R}_q(\mathsf{A}) = \mathcal{R}(\mathsf{A}) \Rightarrow \mathcal{R}_q(\mathsf{A}') = \mathcal{R}(\mathsf{A}')$
- Scalar case: $\mathcal{R}_{s,q}(\mathsf{A}) = \mathcal{R}(\mathsf{A}) \Rightarrow \mathcal{R}_{s,q}(\mathsf{A}') = \mathcal{R}(\mathsf{A}')$
- $\mathbf{r}' \in \mathcal{R}(\mathsf{A}') \leftrightarrow \mathbf{r} \in \mathcal{R}(\mathsf{A})$ with $R_e = H(U_e) = 0$
- The code to achieve $\mathbf{r}'$ is the code to achieve $\mathbf{r}$ with deleting column(s) for U_e, i.e., sending nothing on edge e. All-zero is valid $\mathbb{F}_q$ code, can reverse.



Edge deletion: $\mathsf{A}' = \mathsf{A} \setminus Y_k$

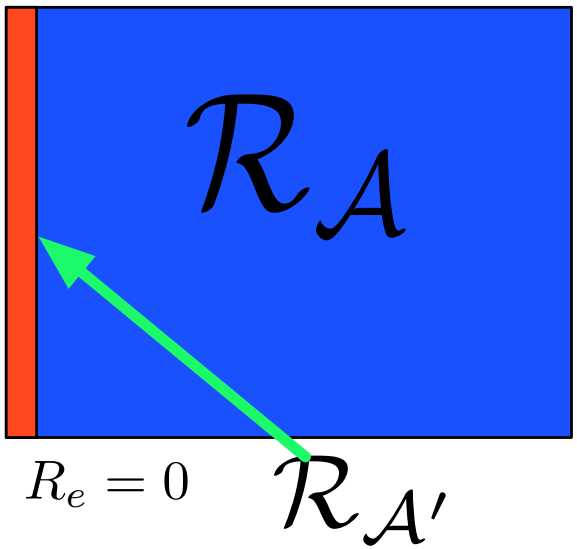
$$\mathcal{R}(\mathsf{A}') = \mathrm{Proj}_{H(Y_S),\mathbf{R}_{\mathcal{E}'}}(\{\mathbf{R} \in \mathcal{R}_q(\mathsf{A}) | R_e = 0\}), \tag{10}$$

$$\mathcal{R}_q(\mathsf{A}') = \mathrm{Proj}_{H(Y_S),\mathbf{R}_{\mathcal{E}'}}(\{\mathbf{R} \in \mathcal{R}_q(\mathsf{A}) | R_e = 0\}), \tag{11}$$

$$\mathcal{R}_{s,q}(\mathsf{A}') = \mathrm{Proj}_{H(Y_S),\mathbf{R}_{\mathcal{E}'}}(\{\mathbf{R} \in \mathcal{R}_{s,q}(\mathsf{A}) | R_e = 0\}). \tag{12}$$

- Preservation: $\mathcal{R}_q(\mathsf{A}) = \mathcal{R}(\mathsf{A}) \Rightarrow \mathcal{R}_q(\mathsf{A}') = \mathcal{R}(\mathsf{A}')$
- Scalar case: $\mathcal{R}_{s,q}(\mathsf{A}) = \mathcal{R}(\mathsf{A}) \Rightarrow \mathcal{R}_{s,q}(\mathsf{A}') = \mathcal{R}(\mathsf{A}')$

- $\mathbf{r}' \in \mathcal{R}(\mathsf{A}') \leftrightarrow \mathbf{r} \in \mathcal{R}(\mathsf{A})$ with $R_e = H(U_e) = 0$
- The code to achieve $\mathbf{r}'$ is the code to achieve $\mathbf{r}$ with deleting column(s) for U_e, i.e., sending nothing on edge e. All-zero is valid $\mathbb{F}_q$ code, can reverse.



Edge deletion: $\mathrm{A}' = \mathrm{A} \setminus Y_k$

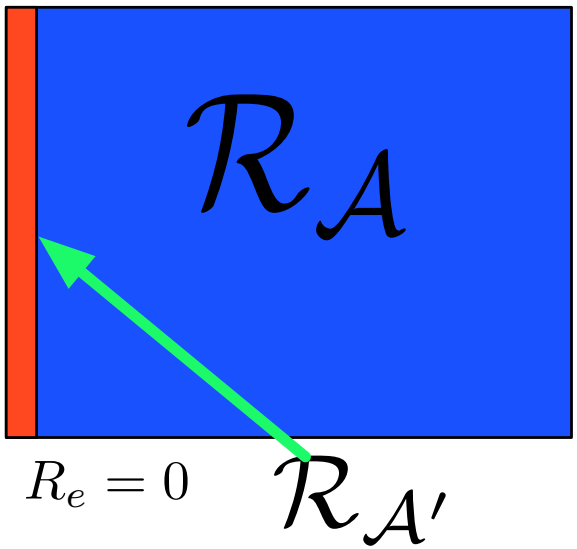
$$\mathcal{R}(\mathrm{A}') = \mathrm{Proj}_{H(Y_S),\mathbf{R}_{\mathcal{E}'}}(\{\mathbf{R} \in \mathcal{R}_q(\mathrm{A}) | R_e = 0\}), \quad (10)$$

$$\mathcal{R}_q(\mathrm{A}') = \mathrm{Proj}_{H(Y_S),\mathbf{R}_{\mathcal{E}'}}(\{\mathbf{R} \in \mathcal{R}_q(\mathrm{A}) | R_e = 0\}), \quad (11)$$

$$\mathcal{R}_{s,q}(\mathrm{A}') = \mathrm{Proj}_{H(Y_S),\mathbf{R}_{\mathcal{E}'}}(\{\mathbf{R} \in \mathcal{R}_{s,q}(\mathrm{A}) | R_e = 0\}). \quad (12)$$

- Preservation: $\mathcal{R}_q(\mathrm{A}) = \mathcal{R}(\mathrm{A}) \Rightarrow \mathcal{R}_q(\mathrm{A}') = \mathcal{R}(\mathrm{A}')$
- Scalar case: $\mathcal{R}_{s,q}(\mathrm{A}) = \mathcal{R}(\mathrm{A}) \Rightarrow \mathcal{R}_{s,q}(\mathrm{A}') = \mathcal{R}(\mathrm{A}')$

- $\mathbf{r}' \in \mathcal{R}(\mathrm{A}') \leftrightarrow \mathbf{r} \in \mathcal{R}(\mathrm{A})$ with $R_e = H(U_e) = 0$
- The code to achieve $\mathbf{r}'$ is the code to achieve $\mathbf{r}$ with deleting column(s) for U_e, i.e., sending nothing on edge e. All-zero is valid $\mathbb{F}_q$ code, can reverse.

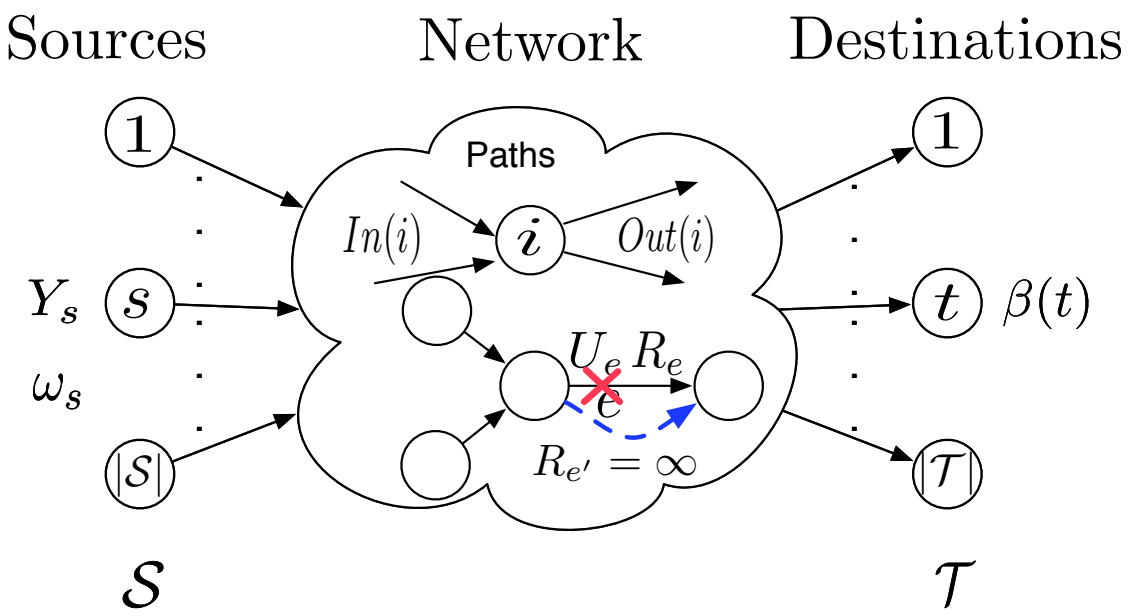


Outline

1. Motivation
2. Background on Rate regions and codes
3. Network operations
   - Source deletion
   - Edge deletion
   - Edge contraction
4. MDCS demonstration
   - MDCS model
   - Corresponding operations
   - Forbidden embedded networks for scalar binary codes

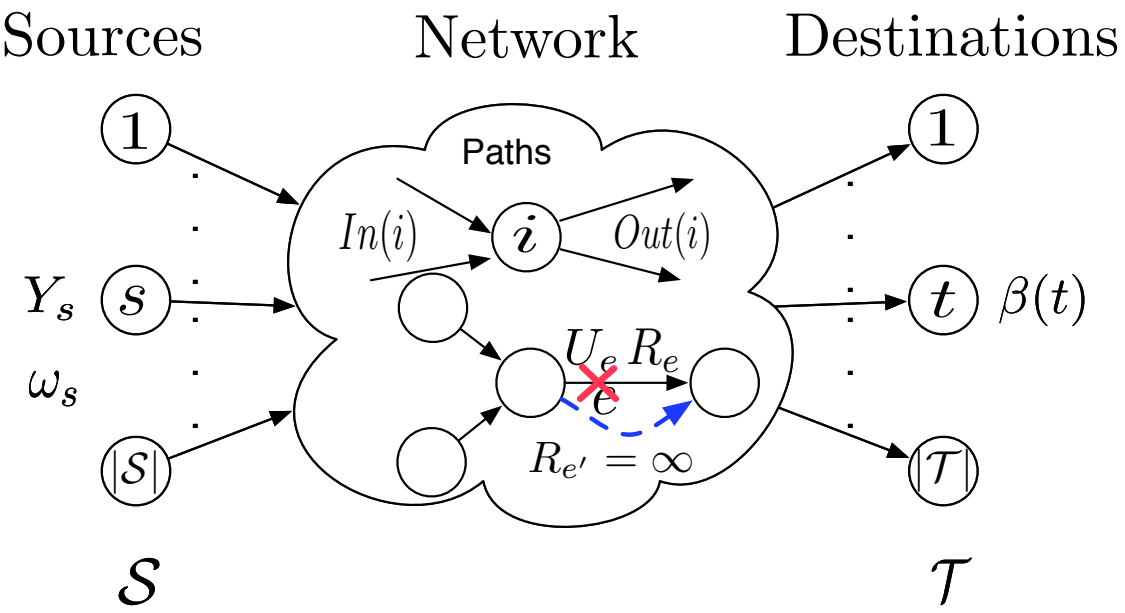
Edge contraction

- $\mathsf{A}' = \mathsf{A}/e$
- Edge e contracted, input to tail of e available for head of e, $R_e = \infty$, $H(U_e)$ free.



Edge contraction

- $\mathsf{A}' = \mathsf{A}/e$
- Edge e contracted, input to tail of e available for head of e, $R_e = \infty$, $H(U_e)$ free.



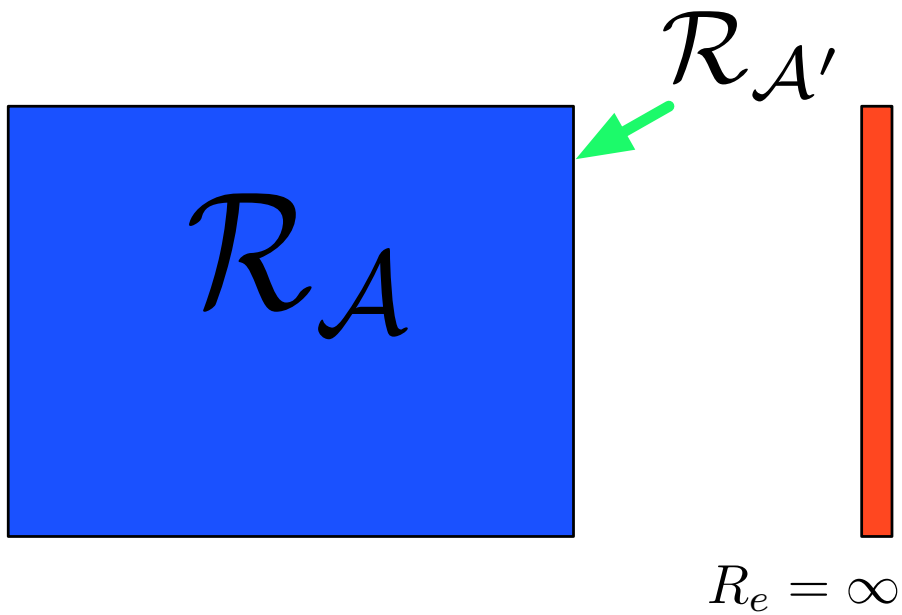
Edge contraction: $\mathrm{A}' = \mathrm{A}/e$

$$\mathcal{R}(\mathrm{A}') = \mathrm{Proj}_{H(Y_k),k\in\mathcal{S},\mathbf{R}_{\mathcal{E}'}}\mathcal{R}(\mathrm{A}), \quad (13)$$

$$\mathcal{R}_q(\mathrm{A}') = \mathrm{Proj}_{H(Y_k),k\in\mathcal{S},\mathbf{R}_{\mathcal{E}'}}\mathcal{R}_q(\mathrm{A}), \quad (14)$$

$$\mathcal{R}_{s,q}(\mathrm{A}') \supseteq \mathrm{Proj}_{H(Y_k),k\in\mathcal{S},\mathbf{R}_{\mathcal{E}'}}\mathcal{R}_{s,q}(\mathrm{A}), \quad (15)$$

- Preservation: $\mathcal{R}_q(\mathrm{A}) = \mathcal{R}(\mathrm{A}) \Rightarrow \mathcal{R}_q(\mathrm{A}') = \mathcal{R}(\mathrm{A}')$
- Scalar case: $\mathcal{R}_{s,q}(\mathrm{A}) = \mathcal{R}(\mathrm{A}) \Rightarrow \mathcal{R}_{s,q}(\mathrm{A}') = \mathcal{R}(\mathrm{A}')$
- $\mathbf{r}' \in \mathcal{R}(\mathrm{A}') \leftrightarrow \mathbf{r} \in \mathcal{R}(\mathrm{A})$ with some $H(U_e)$ (not of interest in A')
- The code to achieve $\mathbf{r}'$ is the code to achieve $\mathbf{r}$ with deleting column(s) for U_e, sending possibly everything available on e, possibly violating scalar code.



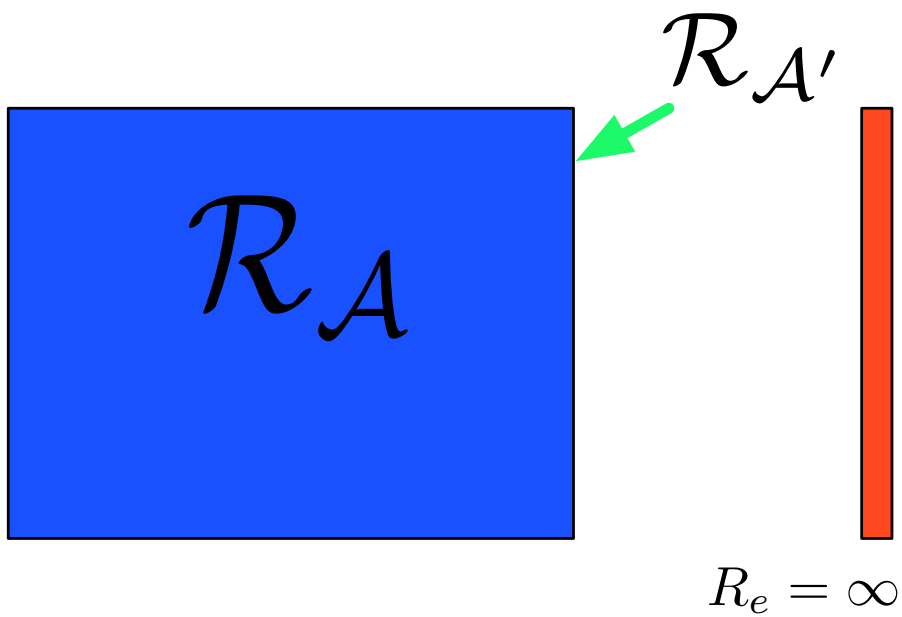
Edge contraction: $\mathsf{A}' = \mathsf{A}/e$

$$\mathcal{R}(\mathsf{A}') = \mathrm{Proj}_{H(Y_k),k\in\mathcal{S},\mathbf{R}_{\mathcal{E}'}}\mathcal{R}(\mathsf{A}), \quad (13)$$

$$\mathcal{R}_q(\mathsf{A}') = \mathrm{Proj}_{H(Y_k),k\in\mathcal{S},\mathbf{R}_{\mathcal{E}'}}\mathcal{R}_q(\mathsf{A}), \quad (14)$$

$$\mathcal{R}_{s,q}(\mathsf{A}') \supseteq \mathrm{Proj}_{H(Y_k),k\in\mathcal{S},\mathbf{R}_{\mathcal{E}'}}\mathcal{R}_{s,q}(\mathsf{A}), \quad (15)$$

- Preservation: $\mathcal{R}_q(\mathsf{A}) = \mathcal{R}(\mathsf{A}) \Rightarrow \mathcal{R}_q(\mathsf{A}') = \mathcal{R}(\mathsf{A}')$
- Scalar case: $\mathcal{R}_{s,q}(\mathsf{A}) = \mathcal{R}(\mathsf{A}) \Rightarrow \mathcal{R}_{s,q}(\mathsf{A}') = \mathcal{R}(\mathsf{A}')$
- $\mathbf{r}' \in \mathcal{R}(\mathsf{A}') \leftrightarrow \mathbf{r} \in \mathcal{R}(\mathsf{A})$ with some $H(U_e)$ (not of interest in A')
- The code to achieve $\mathbf{r}'$ is the code to achieve $\mathbf{r}$ with deleting column(s) for U_e, sending possibly everything available on e, possibly violating scalar code.



Edge contraction: $\mathsf{A}' = \mathsf{A}/e$

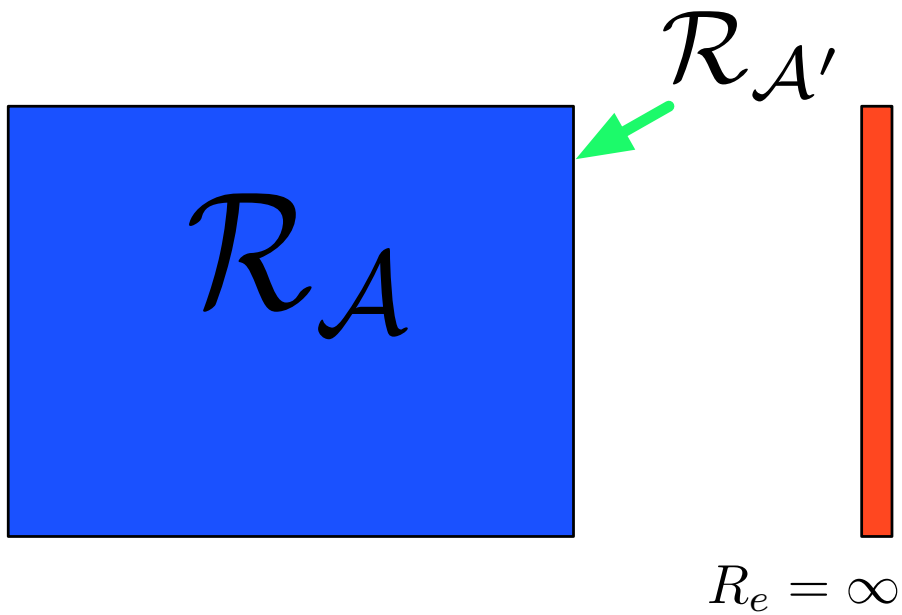
$$\mathcal{R}(\mathsf{A}') = \mathrm{Proj}_{H(Y_k),k\in\mathcal{S},\mathbf{R}_{\mathcal{E}'}}\mathcal{R}(\mathsf{A}), \quad (13)$$

$$\mathcal{R}_q(\mathsf{A}') = \mathrm{Proj}_{H(Y_k),k\in\mathcal{S},\mathbf{R}_{\mathcal{E}'}}\mathcal{R}_q(\mathsf{A}), \quad (14)$$

$$\mathcal{R}_{s,q}(\mathsf{A}') \supseteq \mathrm{Proj}_{H(Y_k),k\in\mathcal{S},\mathbf{R}_{\mathcal{E}'}}\mathcal{R}_{s,q}(\mathsf{A}), \quad (15)$$

- Preservation: $\mathcal{R}_q(\mathsf{A}) = \mathcal{R}(\mathsf{A}) \Rightarrow \mathcal{R}_q(\mathsf{A}') = \mathcal{R}(\mathsf{A}')$
- Scalar case: $\mathcal{R}_{s,q}(\mathsf{A}) = \mathcal{R}(\mathsf{A}) \Rightarrow \mathcal{R}_{s,q}(\mathsf{A}') = \mathcal{R}(\mathsf{A}')$

- $\mathbf{r}' \in \mathcal{R}(\mathsf{A}') \leftrightarrow \mathbf{r} \in \mathcal{R}(\mathsf{A})$ with some $H(U_e)$ (not of interest in A')
- The code to achieve $\mathbf{r}'$ is the code to achieve $\mathbf{r}$ with deleting column(s) for U_e, sending possibly everything available on e, possibly violating scalar code.



Edge contraction: $\mathsf{A}' = \mathsf{A}/e$

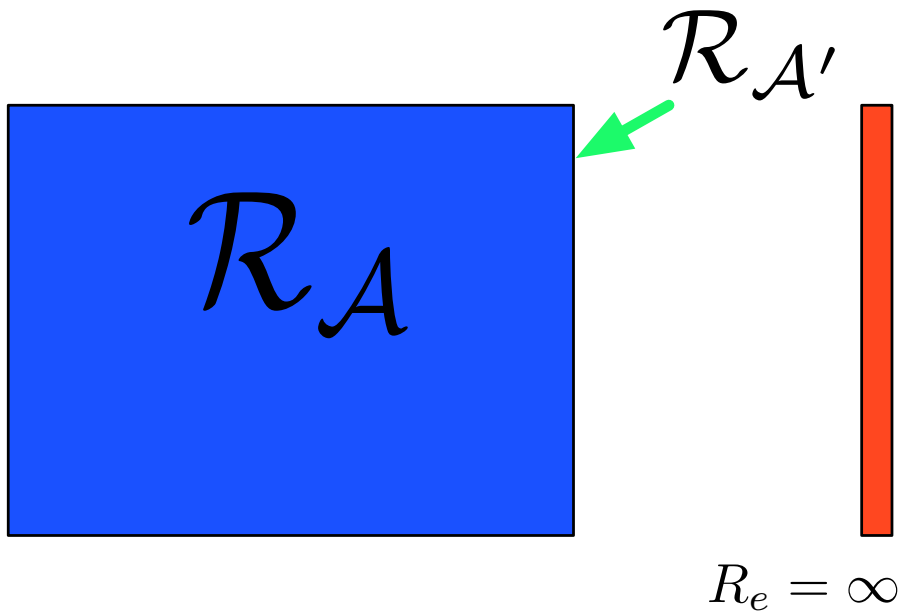
$$\mathcal{R}(\mathsf{A}') = \mathrm{Proj}_{H(Y_k),k\in\mathcal{S},\mathbf{R}_{\mathcal{E}'}}\mathcal{R}(\mathsf{A}), \quad (13)$$

$$\mathcal{R}_q(\mathsf{A}') = \mathrm{Proj}_{H(Y_k),k\in\mathcal{S},\mathbf{R}_{\mathcal{E}'}}\mathcal{R}_q(\mathsf{A}), \quad (14)$$

$$\mathcal{R}_{s,q}(\mathsf{A}') \supseteq \mathrm{Proj}_{H(Y_k),k\in\mathcal{S},\mathbf{R}_{\mathcal{E}'}}\mathcal{R}_{s,q}(\mathsf{A}), \quad (15)$$

- Preservation: $\mathcal{R}_q(\mathsf{A}) = \mathcal{R}(\mathsf{A}) \Rightarrow \mathcal{R}_q(\mathsf{A}') = \mathcal{R}(\mathsf{A}')$
- Scalar case: $\mathcal{R}_{s,q}(\mathsf{A}) = \mathcal{R}(\mathsf{A}) \Rightarrow \mathcal{R}_{s,q}(\mathsf{A}') = \mathcal{R}(\mathsf{A}')$

- $\mathbf{r}' \in \mathcal{R}(\mathsf{A}') \leftrightarrow \mathbf{r} \in \mathcal{R}(\mathsf{A})$ with some $H(U_e)$ (not of interest in A')
- The code to achieve $\mathbf{r}'$ is the code to achieve $\mathbf{r}$ with deleting column(s) for U_e, sending possibly everything available on e, possibly violating scalar code.



Edge contraction: $\mathsf{A}' = \mathsf{A}/e$

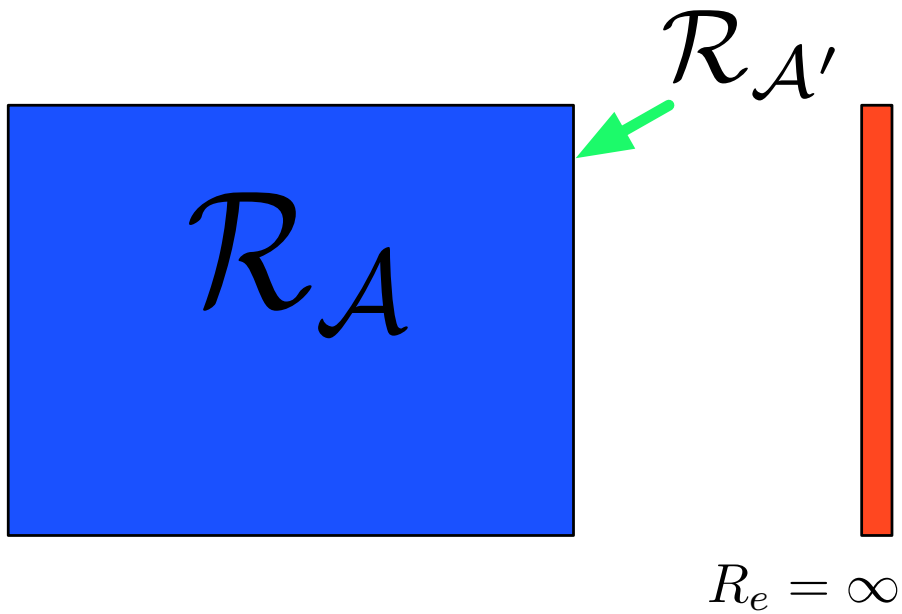
$$\mathcal{R}(\mathsf{A}') = \mathrm{Proj}_{H(Y_k),k\in\mathcal{S},\mathbf{R}_{\mathcal{E}'}}\mathcal{R}(\mathsf{A}), \quad (13)$$

$$\mathcal{R}_q(\mathsf{A}') = \mathrm{Proj}_{H(Y_k),k\in\mathcal{S},\mathbf{R}_{\mathcal{E}'}}\mathcal{R}_q(\mathsf{A}), \quad (14)$$

$$\mathcal{R}_{s,q}(\mathsf{A}') \supseteq \mathrm{Proj}_{H(Y_k),k\in\mathcal{S},\mathbf{R}_{\mathcal{E}'}}\mathcal{R}_{s,q}(\mathsf{A}), \quad (15)$$

- Preservation: $\mathcal{R}_q(\mathsf{A}) = \mathcal{R}(\mathsf{A}) \Rightarrow \mathcal{R}_q(\mathsf{A}') = \mathcal{R}(\mathsf{A}')$
- Scalar case: $\mathcal{R}_{s,q}(\mathsf{A}) = \mathcal{R}(\mathsf{A}) \Rightarrow \mathcal{R}_{s,q}(\mathsf{A}') = \mathcal{R}(\mathsf{A}')$

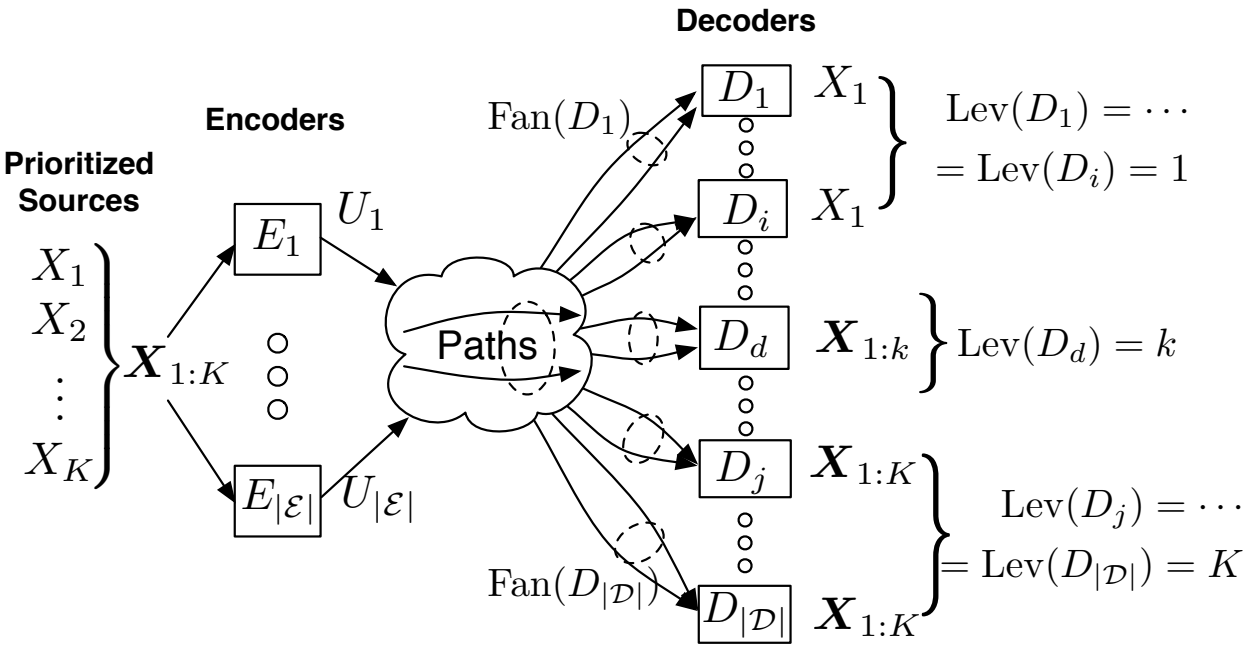
- $\mathbf{r}' \in \mathcal{R}(\mathsf{A}') \leftrightarrow \mathbf{r} \in \mathcal{R}(\mathsf{A})$ with some $H(U_e)$ (not of interest in A')
- The code to achieve $\mathbf{r}'$ is the code to achieve $\mathbf{r}$ with deleting column(s) for U_e, sending possibly everything available on e, possibly violating scalar code.



Outline

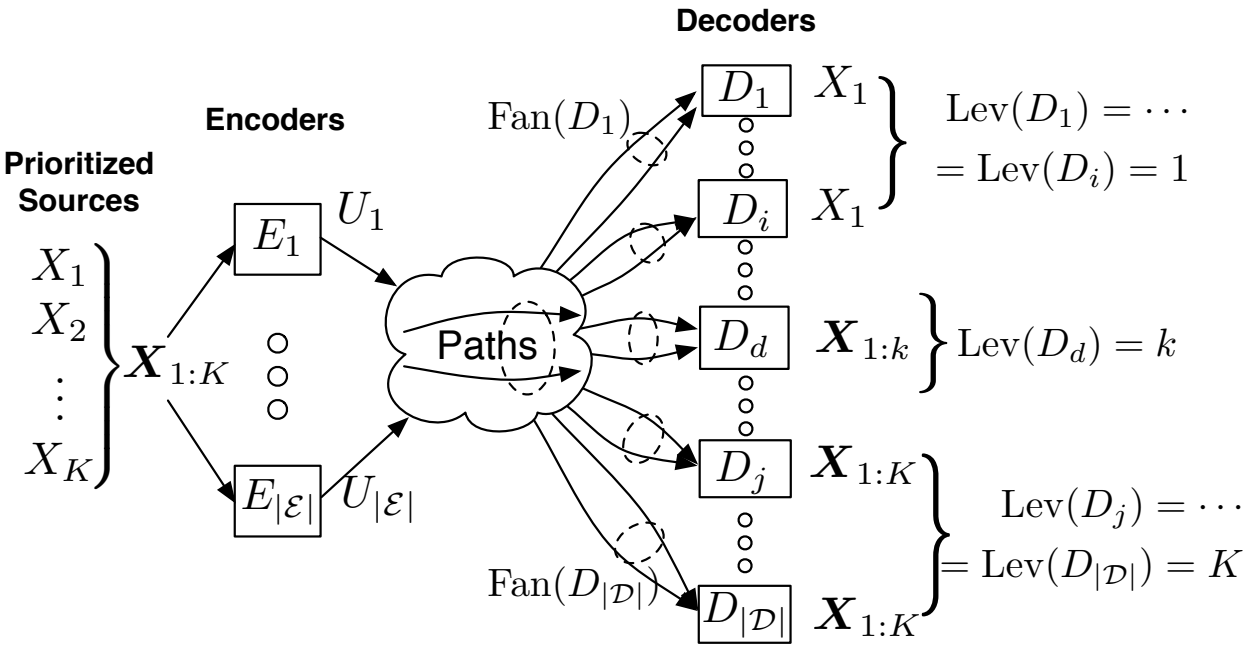
1. Motivation
2. Background on Rate regions and codes
3. Network operations
   - Source deletion
   - Edge deletion
   - Edge contraction
4. MDCS demonstration
   - MDCS model
   - Corresponding operations
   - Forbidden embedded networks for scalar binary codes

Multilevel Diversity Coding Systems (MDCS) model



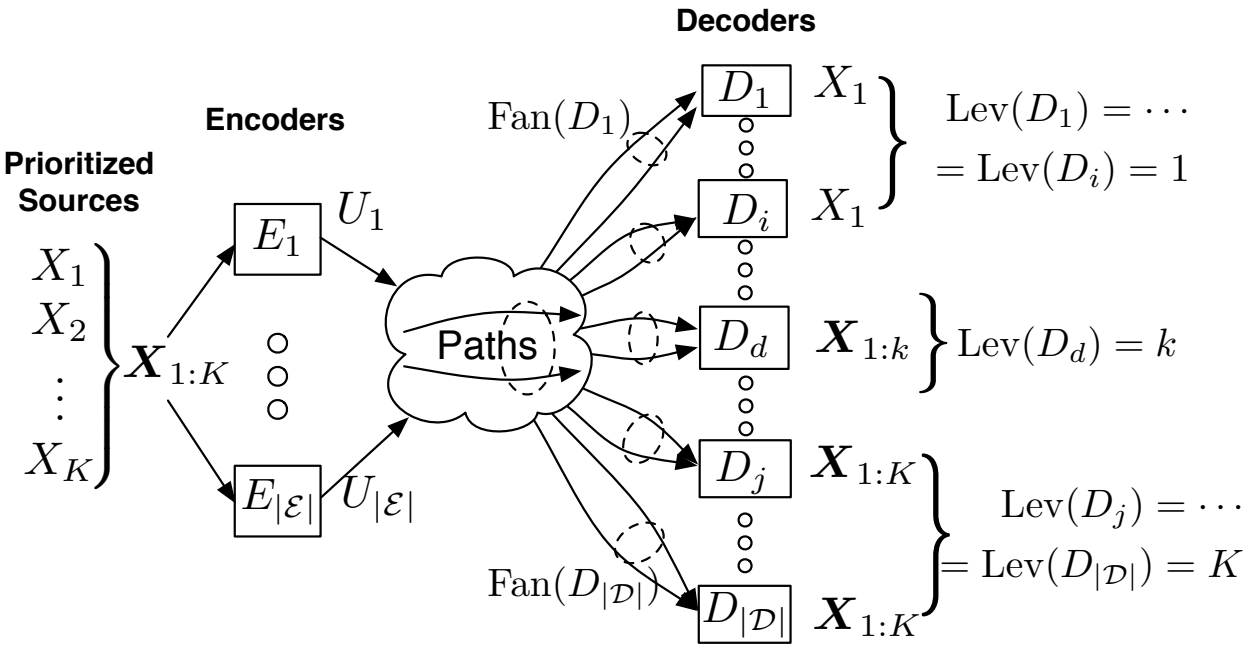
- Prioritized sources, no intermediate nodes
- Decoders are classified to levels: level k, decode $\mathbf{X}_{1:k}$
- Notation: $(k, |\mathcal{E}|)$ MDCS instances

Multilevel Diversity Coding Systems (MDCS) model



- Prioritized sources, no intermediate nodes
- Decoders are classified to levels: level k, decode $\mathbf{X}_{1:k}$
- Notation: $(k, |\mathcal{E}|)$ MDCS instances

Multilevel Diversity Coding Systems (MDCS) model

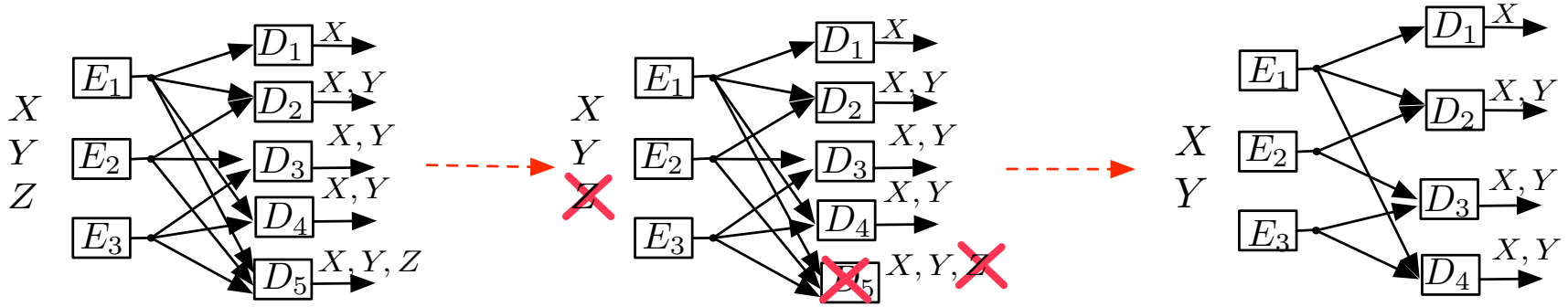


- Prioritized sources, no intermediate nodes
- Decoders are classified to levels: level k, decode $\mathbf{X}_{1:k}$
- Notation: $(k, |\mathcal{E}|)$ MDCS instances

Outline

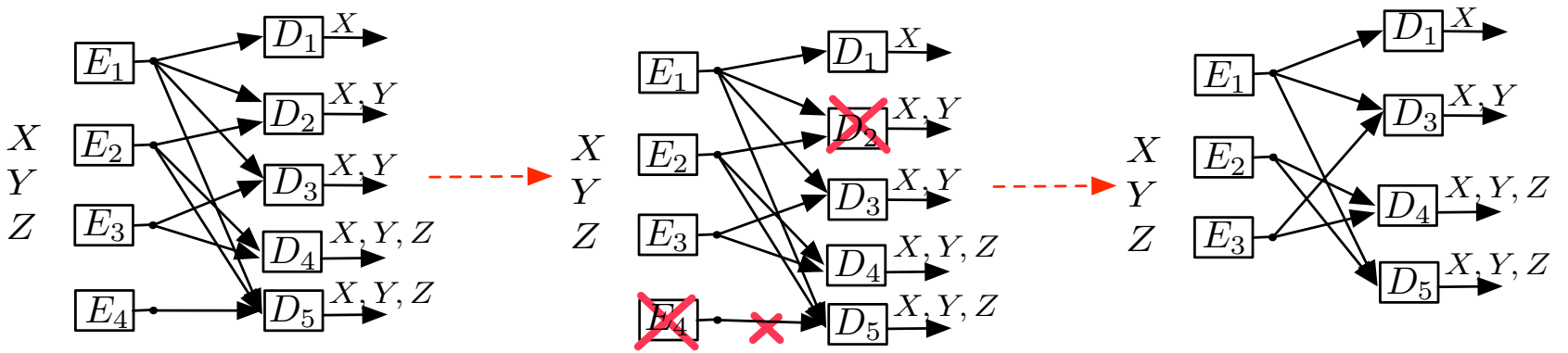
1. Motivation
2. Background on Rate regions and codes
3. Network operations
   - Source deletion
   - Edge deletion
   - Edge contraction
4. MDCS demonstration
   - MDCS model
   - Corresponding operations
   - Forbidden embedded networks for scalar binary codes

MDCS source deletion



- Decoder D_5 no longer requires Z
- Redundant decoder is deleted

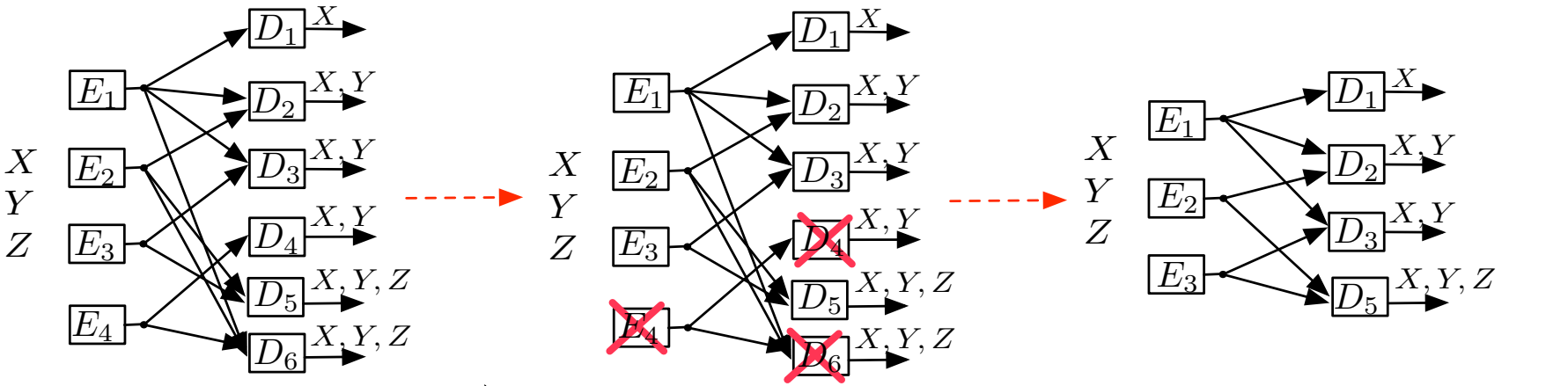
MDCS encoder deletion



i) A $(3, 4)$ MDCS *ii*) Delete encoder E_4 *iii*) After deletion of E_4

- Decoders connected to E_4 keep same decoding ability without E_4
- Redundant decoders deleted

MDCS encoder contraction



i) A $(3, 4)$ MDCS *ii*) Contract encoder E_4 *iii*) After contraction of E_4

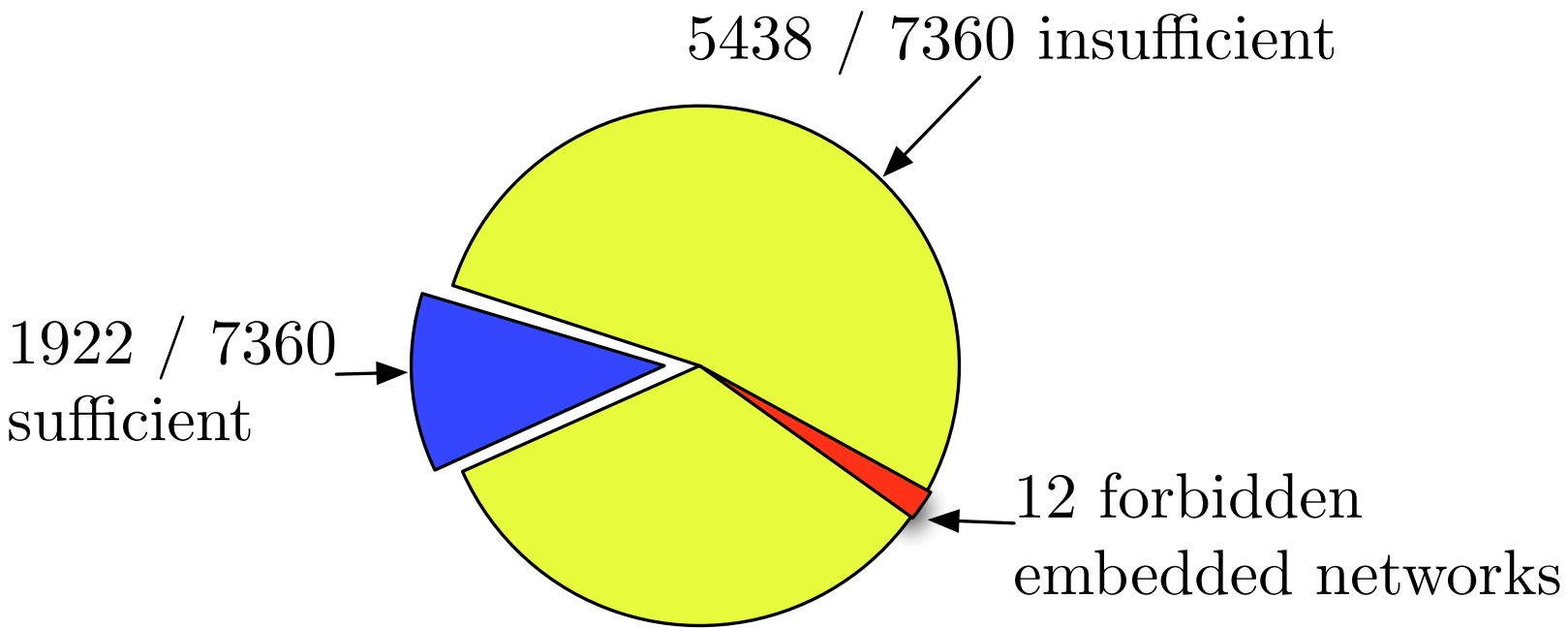
- Decoders connected to E_4 directly access to all sources
- Requirements trivially satisfied, deleted

Outline

1. Motivation
2. Background on Rate regions and codes
3. Network operations
   - Source deletion
   - Edge deletion
   - Edge contraction
4. MDCS demonstration
   - MDCS model
   - Corresponding operations
   - Forbidden embedded networks for scalar binary codes

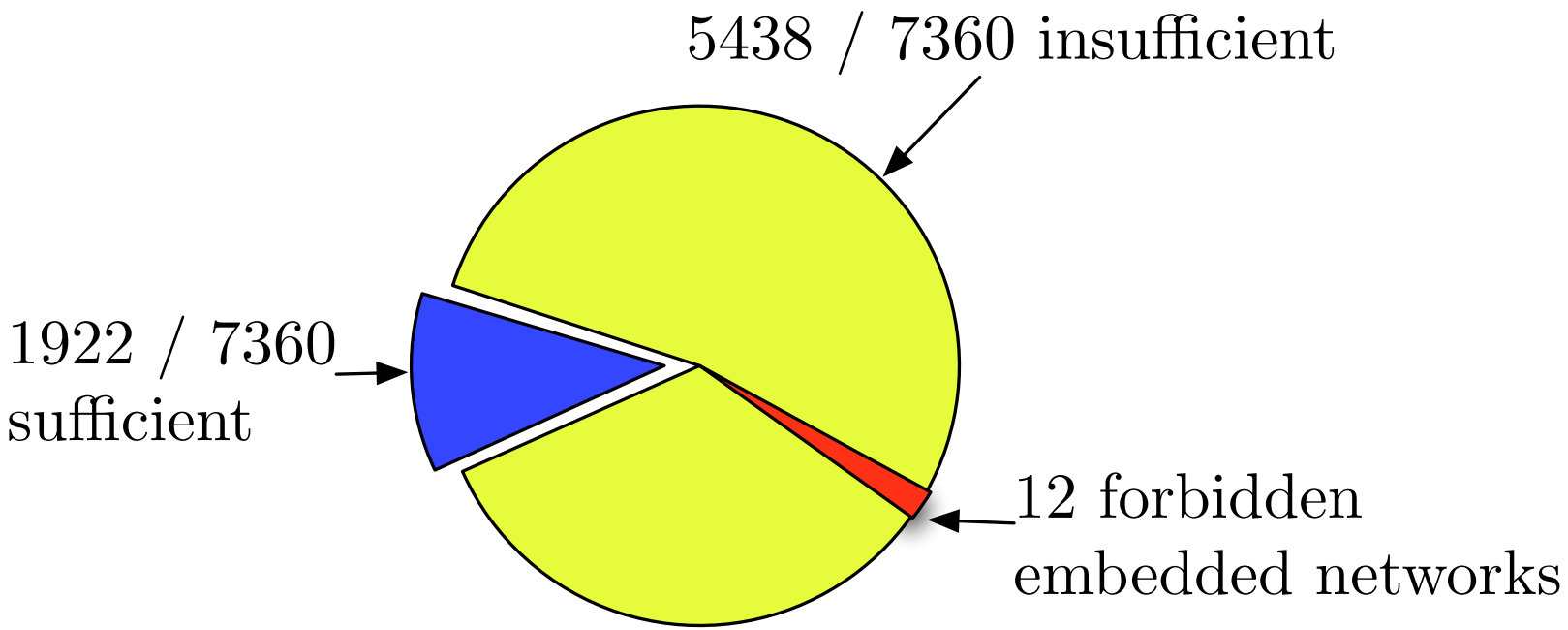
Forbidden embedded networks

- Scalar binary codes considered
- $k = 1, 2, 3$; $|\mathcal{E}| = 2, 3, 4$, 7360 non-isomorphic
- 1922 sufficient, 5438 insufficient
- 12 forbidden embedded networks



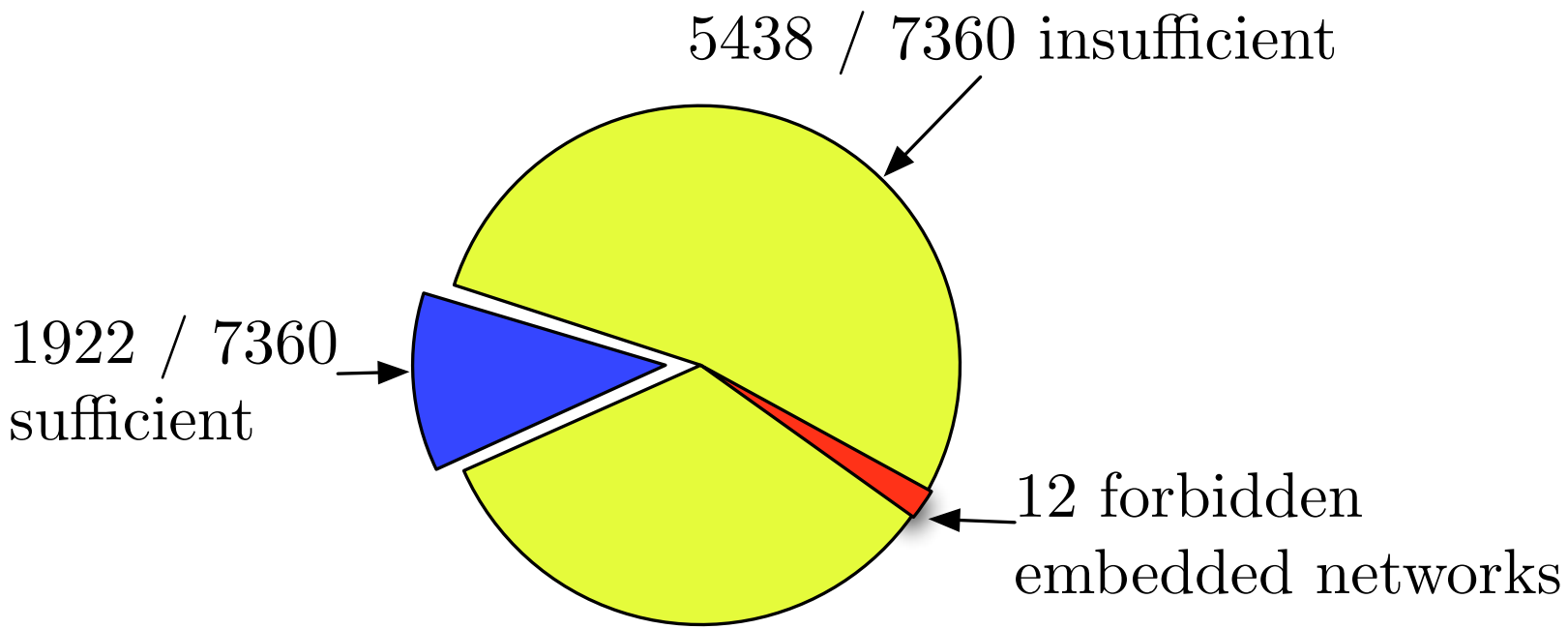
Forbidden embedded networks

- Scalar binary codes considered
- $k = 1, 2, 3$; $|\mathcal{E}| = 2, 3, 4$, 7360 non-isomorphic
- 1922 sufficient, 5438 insufficient
- 12 forbidden embedded networks



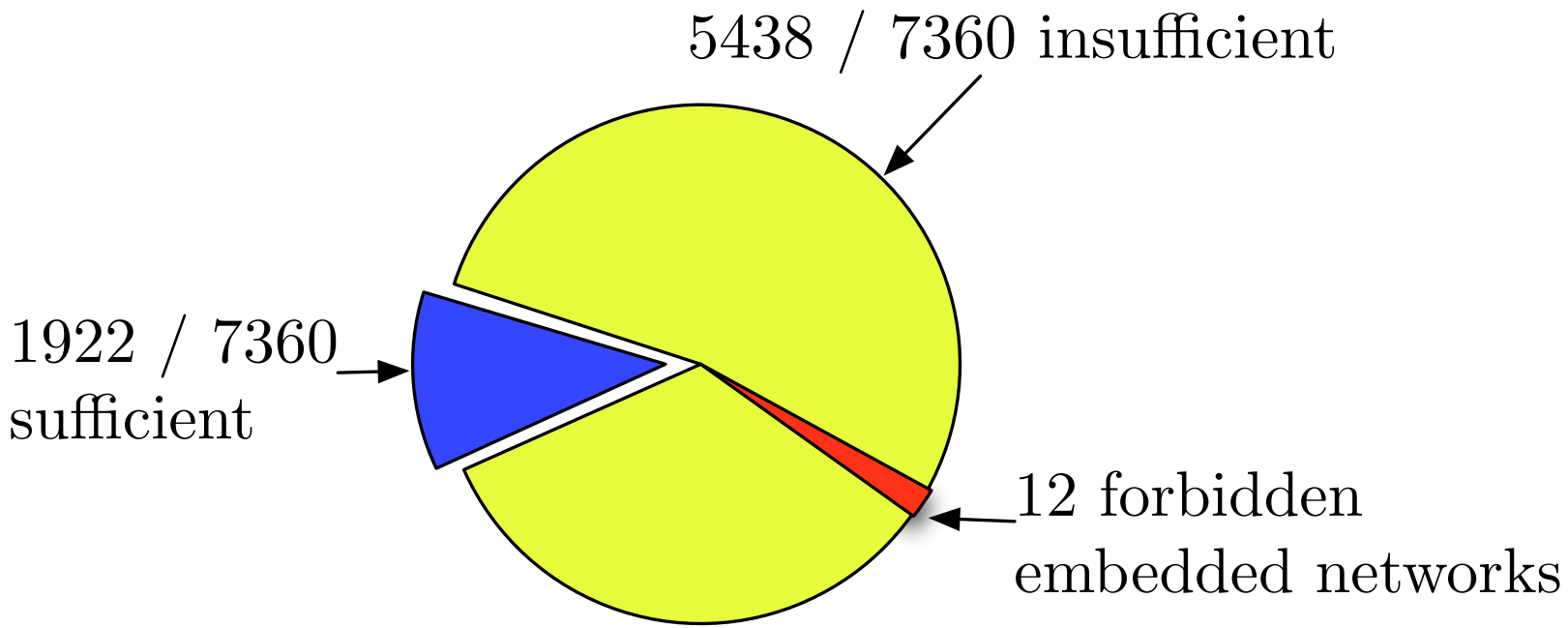
Forbidden embedded networks

- Scalar binary codes considered
- $k = 1, 2, 3$; $|\mathcal{E}| = 2, 3, 4$, 7360 non-isomorphic
- 1922 sufficient, 5438 insufficient
- 12 forbidden embedded networks



Forbidden embedded networks

- Scalar binary codes considered
- $k = 1, 2, 3$; $|\mathcal{E}| = 2, 3, 4$, 7360 non-isomorphic
- 1922 sufficient, 5438 insufficient
- 12 forbidden embedded networks



Open questions

- Complete list of forbidden embedded networks?
- Finite number of forbidden embedded networks for $\mathbb{F}_q$?
- Topology that determines the insufficiency of linear codes over $\mathbb{F}_q$?

Open questions

- Complete list of forbidden embedded networks?
- Finite number of forbidden embedded networks for $\mathbb{F}_q$?
- Topology that determines the insufficiency of linear codes over $\mathbb{F}_q$?

Open questions

- Complete list of forbidden embedded networks?
- Finite number of forbidden embedded networks for $\mathbb{F}_q$?
- Topology that determines the insufficiency of linear codes over $\mathbb{F}_q$?

References

1. Congduan Li, Steven Weber, and John M. Walsh, "Multilevel Diversity Coding Systems: Rate Regions, Codes, Computation, & Forbidden Minors", *IEEE Transactions on Information Theory*, 2014, SUBMITTED.
2. X. Yan, R.W. Yeung, Z. Zhang, "An Implicit Characterization of the Achievable Rate Region for Acyclic Multisource Multisink Network Coding," *IEEE Transactions on Information Theory*, vol.58, no.9, pp.5625,5639, Sept. 2012

Q & A

Thank you!