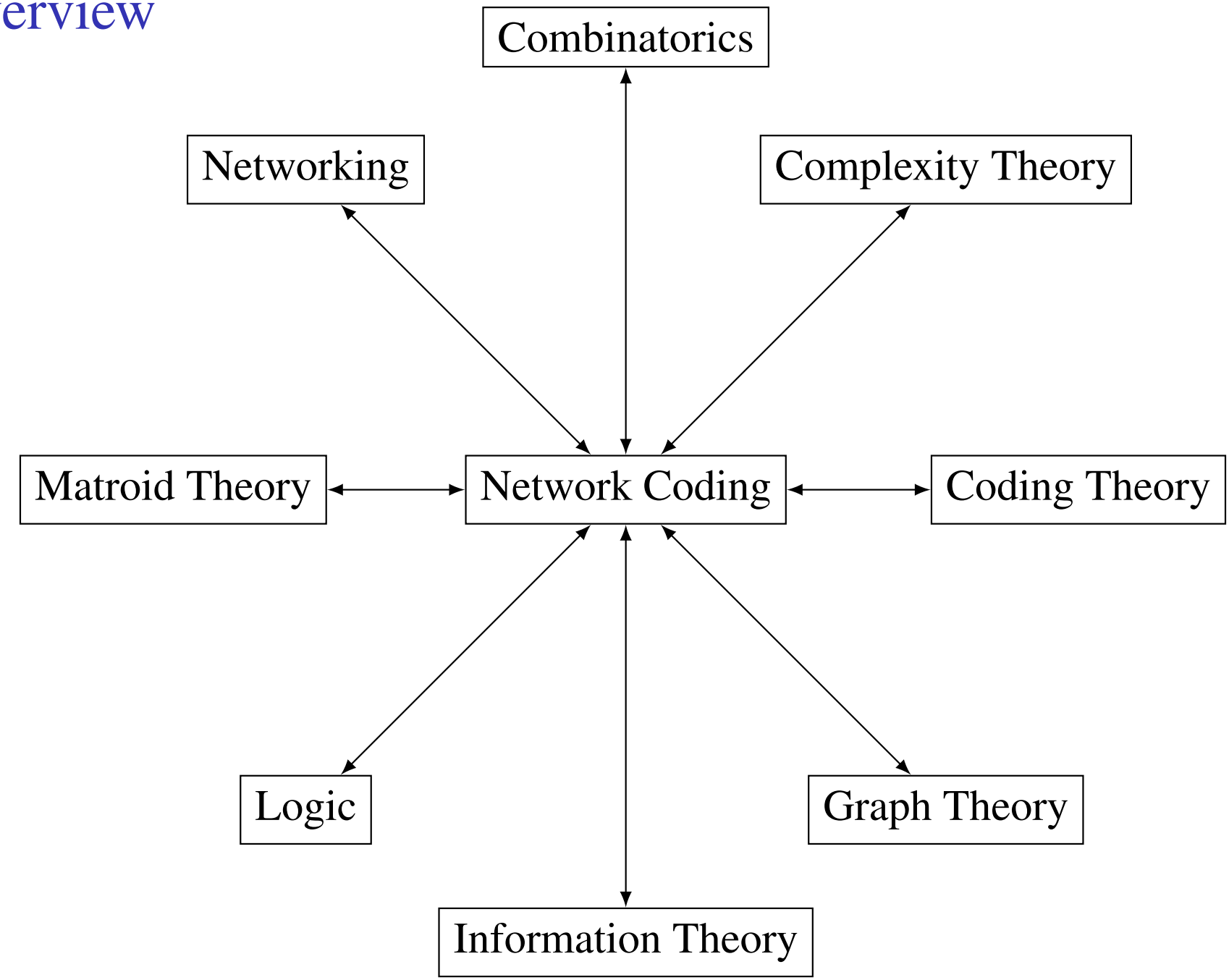
Network Coding

Maximilien Gadouleau

Durham University

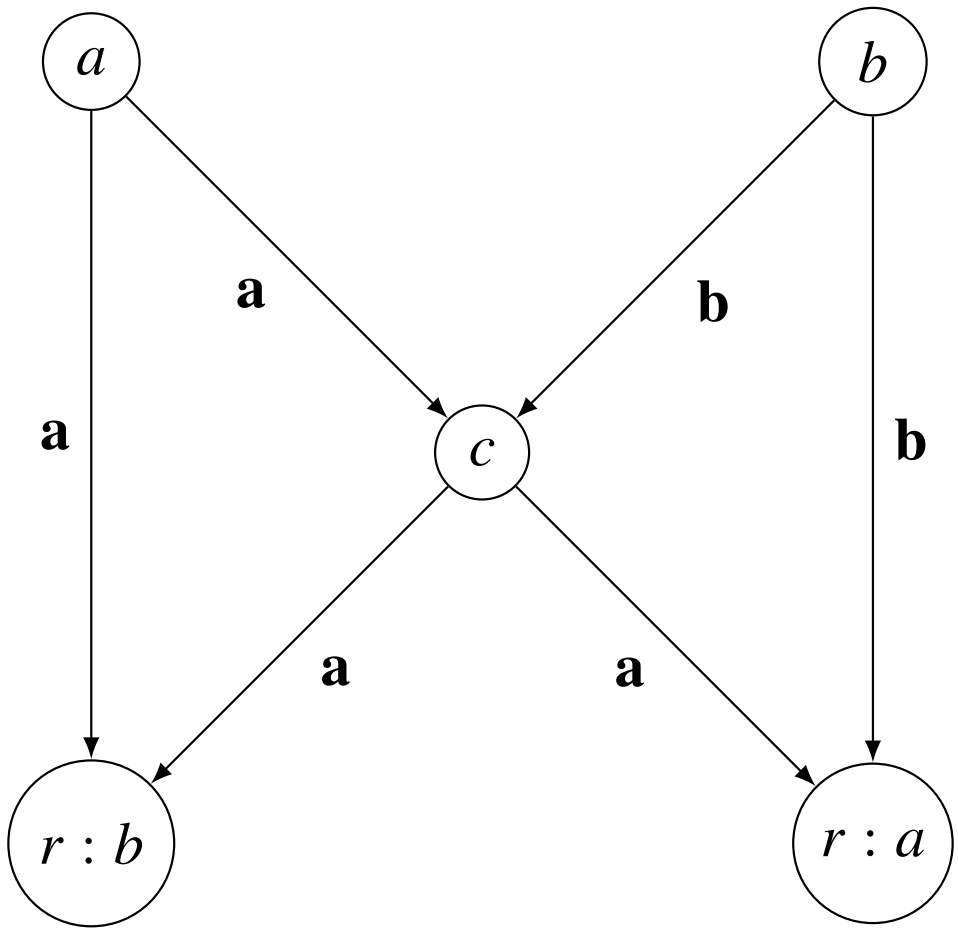
September 11, 2012

Overview


Combinatorics
Networking
Complexity Theory
Matroid Theory
Network Coding
Coding Theory
Logic
Graph Theory
Information Theory

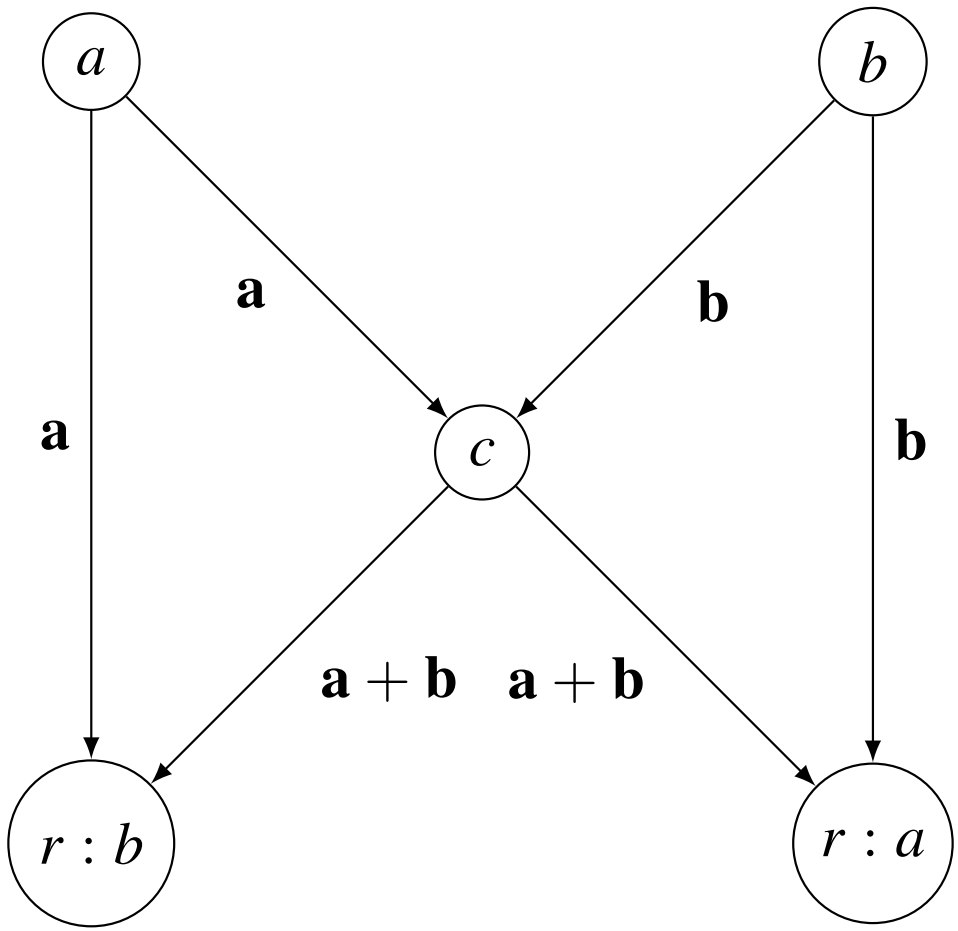
Network coding

- Network coding [Ahlswede et. al. '00]: intermediate nodes combine packets
- Increases throughput: e.g. butterfly network

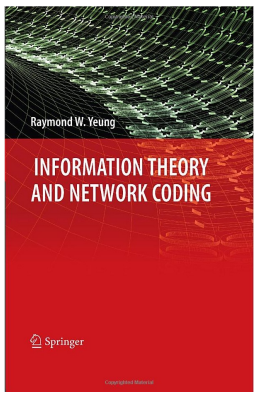


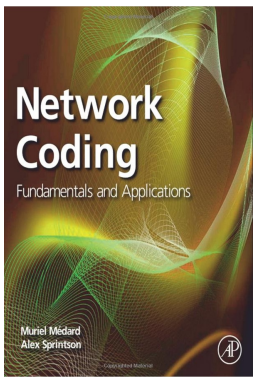
Network coding

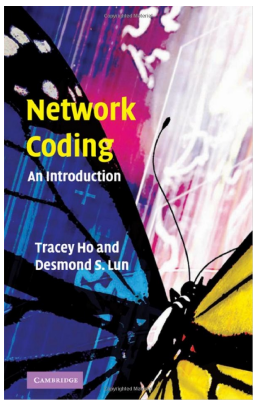
- Network coding [Ahlswede et. al. '00]: intermediate nodes combine packets
- Increases throughput: e.g. butterfly network

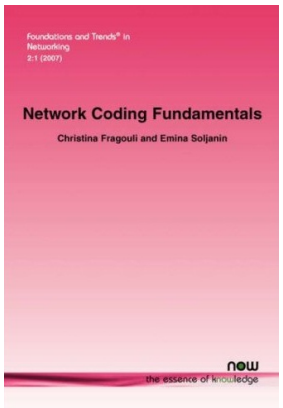


A mature subject









Yeung, *Information Theory and Network Coding*

Médard and Sprintson, *Network Coding, Fundamentals and Applications*

Ho and Lun, *Network Coding, an Introduction*

Fragouli and Soljanin, *Network Coding Fundamentals*

Linear Network Coding

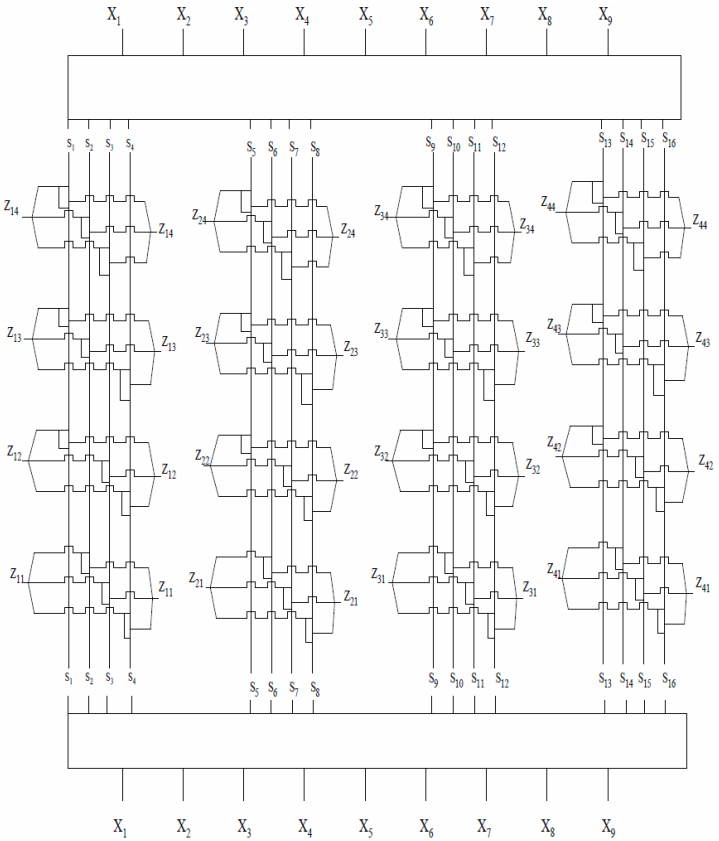
We view packets as vectors over a finite field, and we restrict combinations to Linear combinations only:

$$f(m_1, \ldots, m_k) = \sum_{i=1}^{k} a_i m_i.$$

Linear NC works well in some special cases [Li, Yeung, Cai '03, Koetter and Médard '04].
Efficient algorithms to construct the optimal choice of linear combinations. [Jaggi et. al. '05]

Linear Network Coding continued

Linear network coding is not optimal in general! [Riis '04]



Split: Finding optimal solution v. Finding practical solution

Random Linear Network Coding

- Fixed linear combinations: too complex, too rigid
  $\Rightarrow$ Solution: choose the linear combinations at random
- Advantages
  - Higher throughput for multicast
  - Robustness to packet losses
  - Greater adaptability to topology changes
- Drawback: RLNC sensitive to errors
  Error propagation: a packet in error can corrupt all packets after linear combination

Transmission of invariant: RLNC

- RLNC modifies message:

$$\begin{pmatrix} 1 & 0 & \mathbf{m} \\ 0 & 1 & \mathbf{n} \end{pmatrix} \quad \rightarrow \quad \begin{pmatrix} a_1 & a_2 & a_1\mathbf{m} + a_2\mathbf{n} \\ b_1 & b_2 & b_1\mathbf{m} + b_2\mathbf{n} \end{pmatrix}$$

- It transmits an invariant, i.e. linear space [Koetter and Kschischang '07]
- Errors and packets lost: modification of the subspace
  $\Rightarrow$ Subspace codes, $d(U,V) = \dim(U+V) - \dim(U \cap V)$
- Codes on matrices if header is fixed [Silva, Koetter and Kschischang '08]
  $d(\mathbf{M},\mathbf{N}) = r(\mathbf{M}-\mathbf{N})$: rank metric codes [Gabidulin '85]
- Everything about the network is contained in the distance between the subspaces

Transmission of invariant: Routing

- Routing also modifies message:

$$\begin{pmatrix} 0 & \mathbf{m} \\ 1 & \mathbf{n} \end{pmatrix} \quad \rightarrow \quad \begin{pmatrix} 1 & \mathbf{n} \\ 0 & \mathbf{m} \end{pmatrix}$$

- The order is lost: the invariant transmitted is a set
- A subset is a binary vector
  $\Rightarrow$ Binary codes for error correction with routing [G and Goupil '10]
- Usual solution is via header
  Not the only possible solution: in general, encode into a subset
- Generalised by [G and Goupil '11]: model based on matroids, RANC

Can NC be implemented?

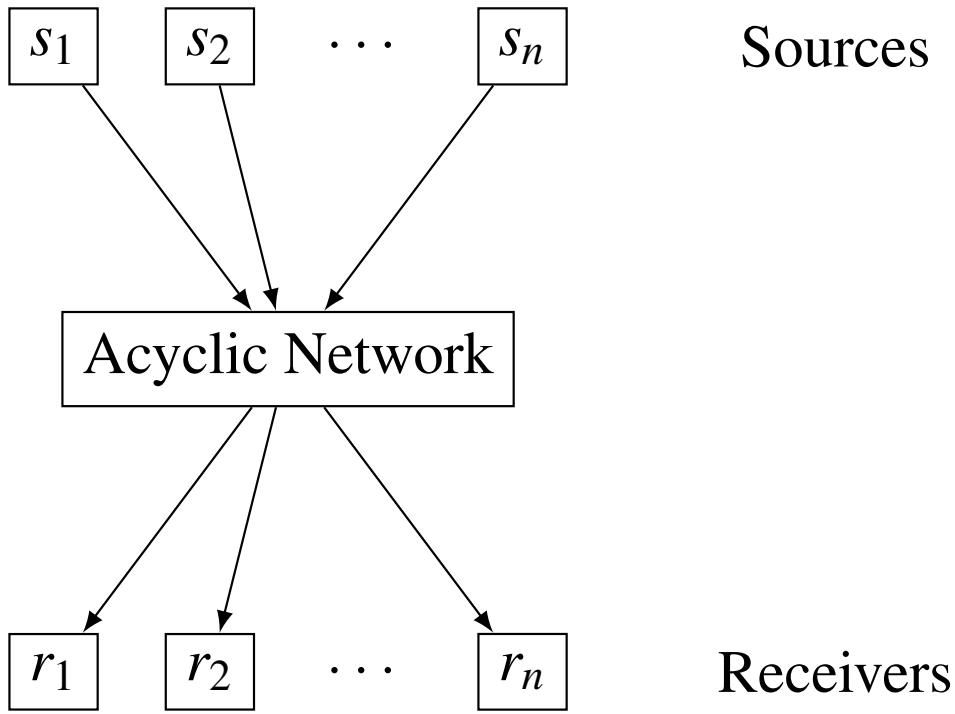
- Not yet! Many obstacles still to face; a huge infrastructure to change
- Further possible applications:
  - File distribution [Microsoft Avalanche]. Idea: send linear combinations of k messages. Coded coupon collector: need $k + o(1)$ randomly chosen linear combinations
  - Distributed storage [Dimakis et. al. '11]. Idea: store linear combinations (coding theoretic problem)
  - Fountain codes: again send linear combinations over a channel which can erase messages. Distribution of combinations finely tailored for efficient decoding (solitons) Raptor codes [Shokrollahi '06]
  - Memoryless computation [Burckel et. al. '96 onwards, G and Riis '12]

Network coding solvability

- Problem: given a network, sources, receivers, how much information can be transmitted?
- Four main approaches
  - Logic: removes network, leaves terms from formal logic
  - Information theory: removes source/intermediate/receiver hierarchy
  - Combinatorics: stability number of a related graph, problem on messages instead of combinations
  - Reverse engineering: design of networks where NC is efficient

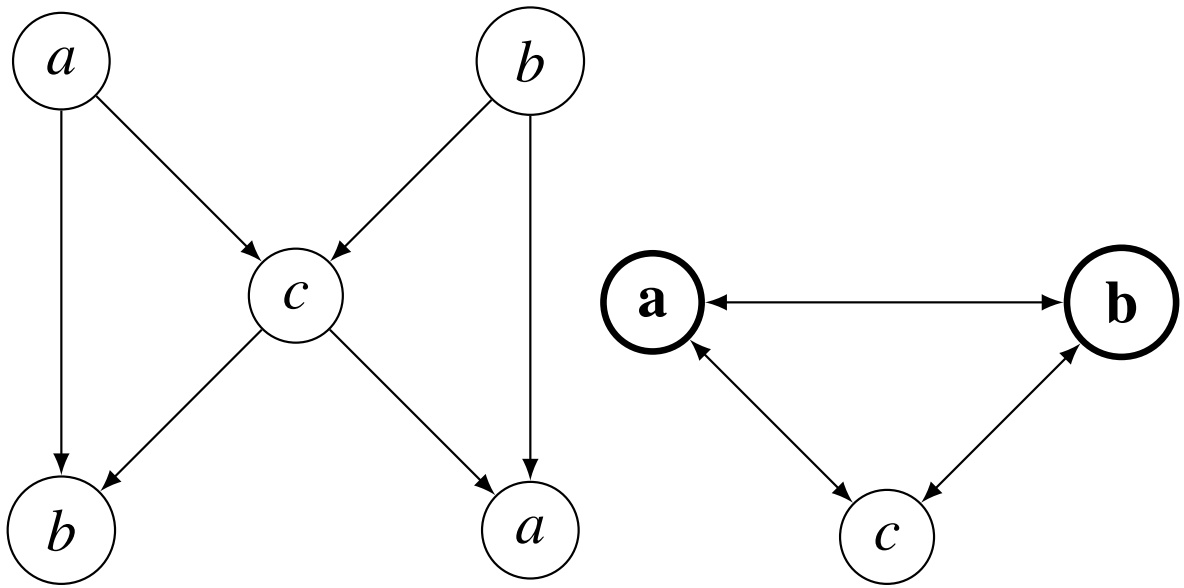
Network coding solvability

Canonical form: multiple unicast.



Each arc carries a symbol from the same finite alphabet A
Each source s_i sends a message $m_i \in A$ to r_i

Network coding and guessing games



All nodes are assigned a value from 0 to $s-1$. They have a common strategy to guess all the values simultaneously

No cooperation: can only guess one assignment. We can do better!

Guessing game

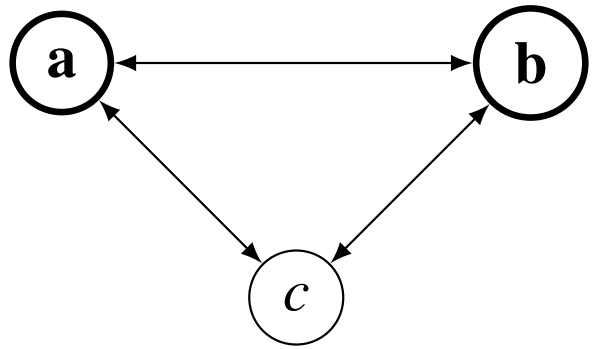
More formally:

- D digraph on n vertices, $A = \{0, 1, \ldots, s-1\}$
- $x = (x_1, \ldots, x_n) \in A^n$ is an assignment of values to the vertices of D
- A strategy F is a set of n functions $f_v(x_{N(v)}) : A^{d_v} \to A$
- x is guessed by F iff $F(x) = x$
- Guessing number

$$g(D, s) = \max_F \log_s |\text{guess}(F)|$$

Guessing game

Thm: $g(D,s) \leq n - mias(D)$, solvable iff equality



Ex: $g(K_3,s) = 2$. Strategy:

$$\begin{aligned} f_a(x_b,x_c) &= -x_b - x_c \mod s \\ f_b(x_a,x_c) &= -x_a - x_c \mod s \\ f_c(x_a,x_b) &= -x_a - x_b \mod s \end{aligned}$$

Guesses all assignments where $x_a + x_b + x_c = 0$

Network design

- Idea: fix the guesses and design the network accordingly
- Parity-check protocol: over $\mathrm{GF}(2)$, see $x_1, \ldots, x_d$ and guess

$$\hat{x_v} = x_1 + \ldots + x_d$$

  sum of the values in the neighbourhood
- The correct guesses form a linear code, whose parity-check matrix is given by $\mathbf{I}_n + \mathbf{A}$
- This achieves a guessing number of $n - \mathrm{rk}(\mathbf{I}_n + \mathbf{A})$, where A is the adjacency matrix of the graph
- Example: on K_3,

$$\mathbf{I}_3 + \mathbf{A} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Cyclic codes

- Cyclic code: all multiples of $p(x)$, where $p(x)|x^n + 1$
- Automatically yields guessing number $\deg(p)$ and a corresponding solvable NC
- Example: $(7, 3)$ simplex code (dual of Hamming code)

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

Cyclic codes

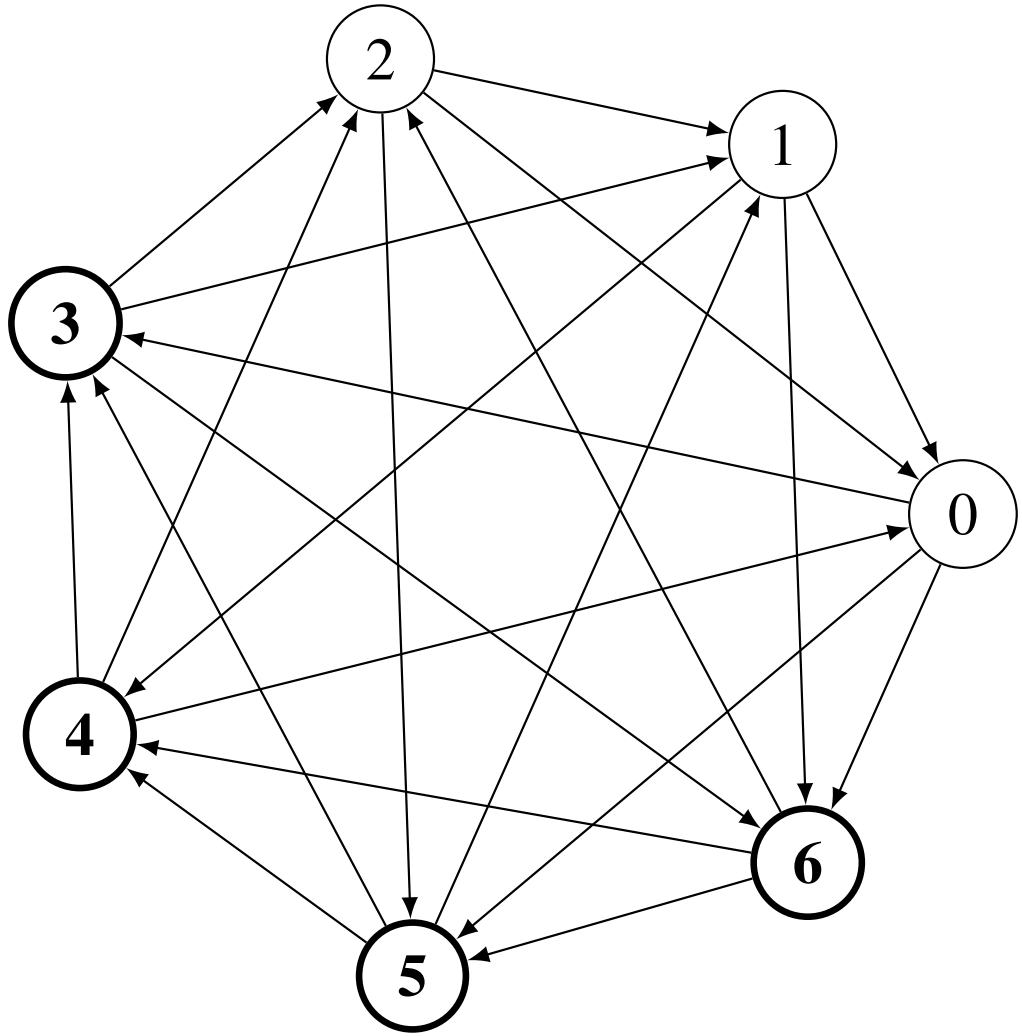


Figure: Digraph P_7 on 7 vertices generated by $x^4 + x^2 + x + 1$ with binary linear guessing number 4.

Not there yet

In general, we want a "bad" code:

- Small minimum distance (for sparse graph)
- Low dimension (for high guessing number)
- Minimum distance of the dual > 1 (for the $\mathbf{I} + \mathbf{A}$ form)

Bad cyclic codes are hard to find!
How to control the girth?
Simplex codes yield dense graphs, Hammnig codes yield graphs with small girth.

Combining graphs

Idea: taking Kronecker product of matrices $\mathbf{I} + \mathbf{A}$.

$$
\begin{aligned}
D &:= D_1 \boxtimes D_2 \\
n &= n_1 n_2 \\
\mathbf{I}_n + \mathbf{A} &= (\mathbf{I}_{n_1} + \mathbf{B}_1) \boxtimes (\mathbf{I}_{n_2} + \mathbf{B}_2) \\
mias &= mias_1 mias_2 \\
n - g &= (n_1 - g_1)(n_2 - g_2) \\
d + 1 &= (d_1 + 1)(d_2 + 1) \\
\gamma &= \min\{\gamma_1, \gamma_2\}
\end{aligned}
$$

E.g. C_3^2

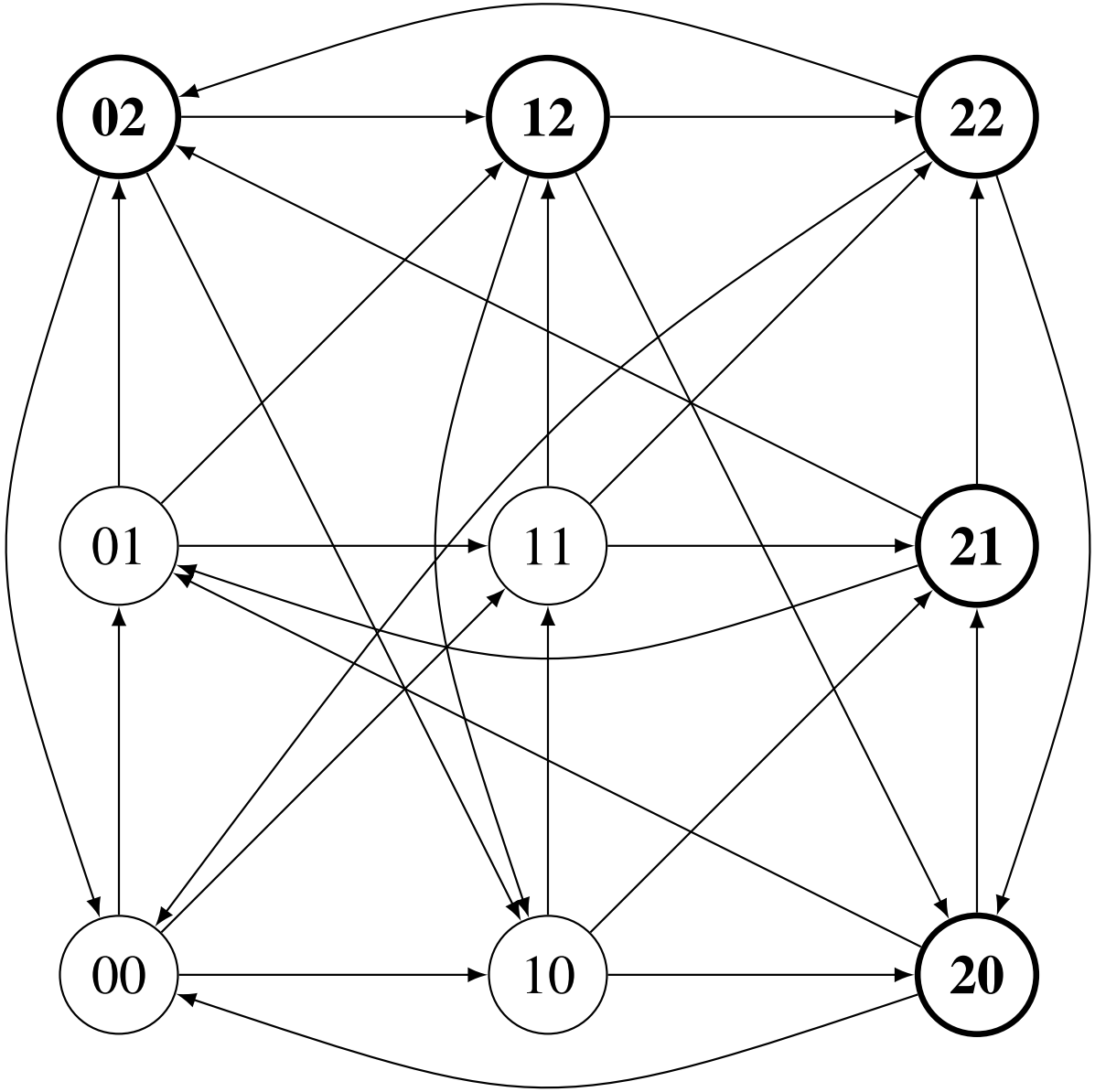


Figure: The digraph C_3^2 with dimension n equal to 5

Product of cycles

In general, for C_l^k:

$$\begin{aligned} n &= l^k \\ mias &= (l-1)^k \\ g &= l^k - (l-1)^k \\ d+1 &= 2^k \\ \gamma &= l \end{aligned}$$

Therefore, γ can be arbitrarily large, and $\lim_{k\to\infty} \frac{g}{n} = 1$.

Conclusion

Where do we go from here?

- Applicability: a lot more work remains
- Solvability: a hard problem with many connections (a lot still unknown or undeveloped)