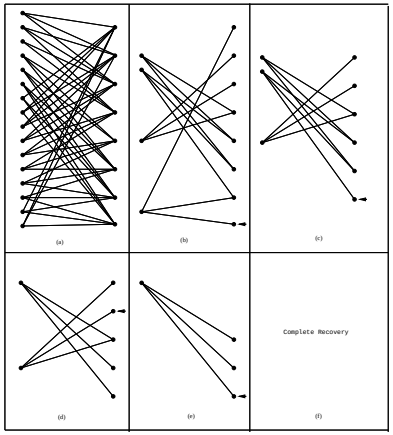
Introduction to Low-Density Parity-Check Codes



Amin Shokrollahi

Motivation

Design of codes for LIVE streaming:

1. Cannot afford to wait to receive everything,
2. Want to be no worse than no coding,
3. Want user defined protection,
4. Want fast encoding/decoding algorithms.

LT-codes contradict points 1 and 2 if many losses.

Reed-Solomon Codes

Can be used to fulfill points 1–3, but not necessarily 4 (in many scenarios).

Low-Density Parity-Check Codes

Were discovered in early 1960's by R. Gallager.

Rediscovered many times by different people.

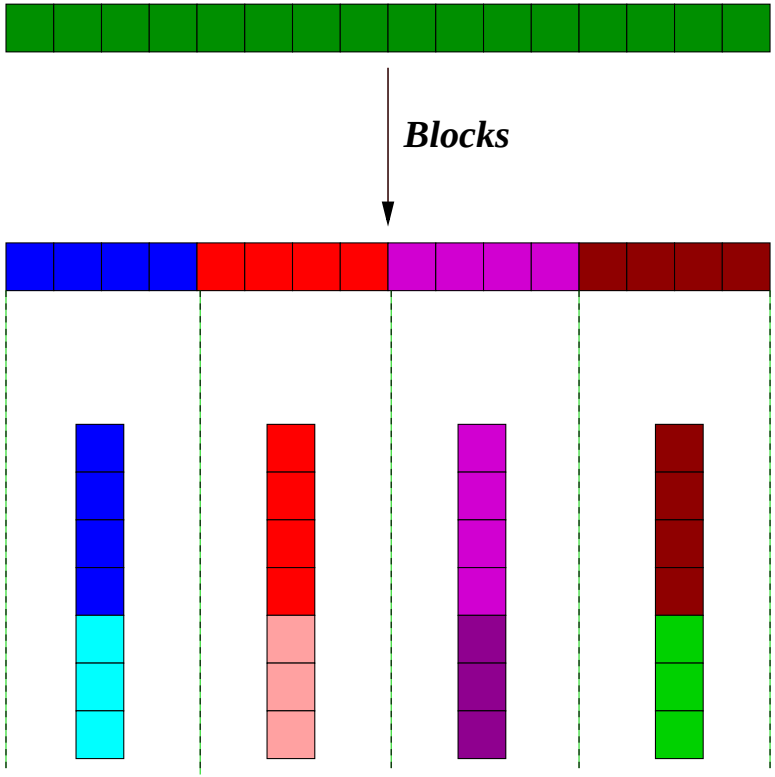
Predecessors of LT codes.

Traditional Error Correcting Codes

1. Block original content into blocks of of packets.

2. Add fixed amount of redundancy to each block so that recovery possible after erasures.

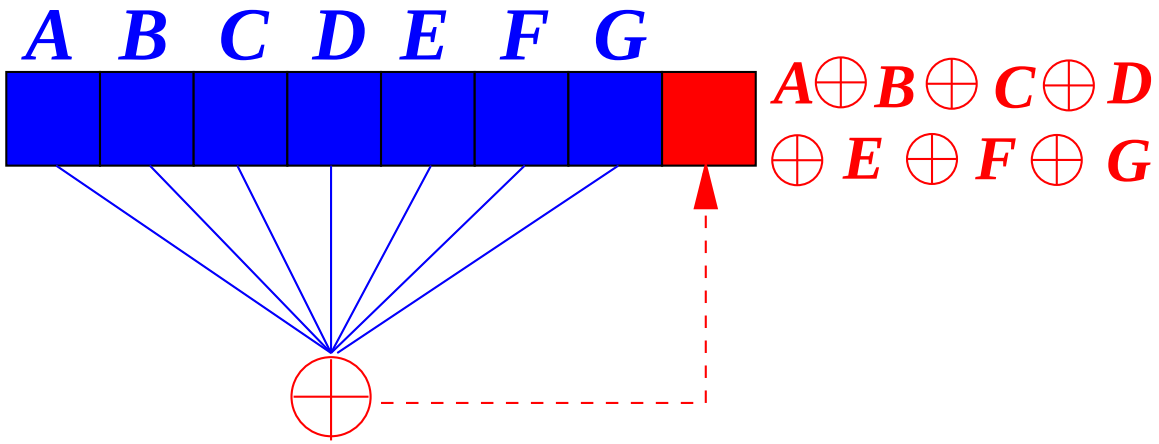
FEC



FEC

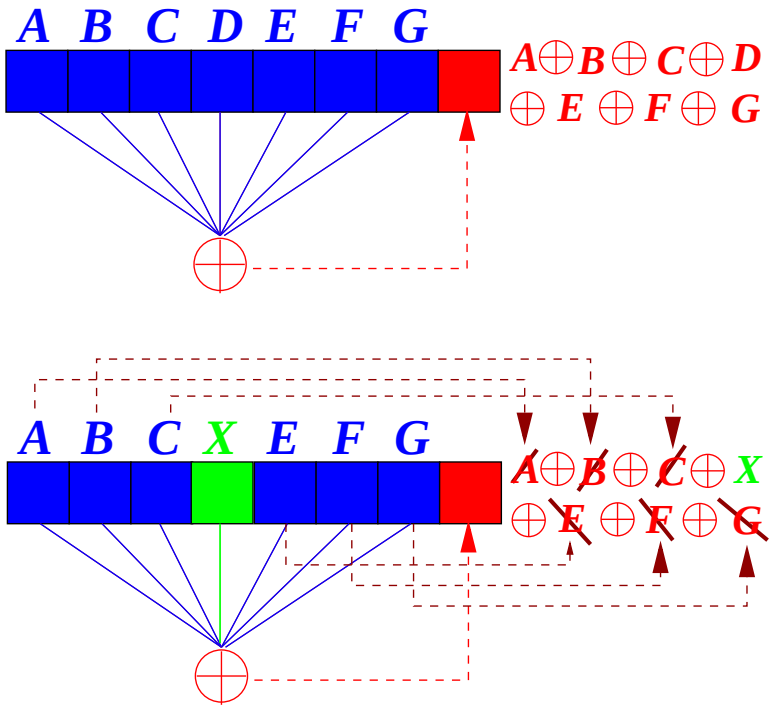
Example: Parity Check

Parity check can correct one erasure!



How to Correct one Erasure

Need to know: $A \oplus A = 0$.



Hamming Code: 2 Erasures

$$(x_1, x_2, x_3, x_4, x_1 + x_2 + x_4, x_1 + x_3 + x_4, x_2 + x_3 + x_4).$$

No matter which two information positions erased, there is a redundant position that has two non-erased information positions and one erased one.

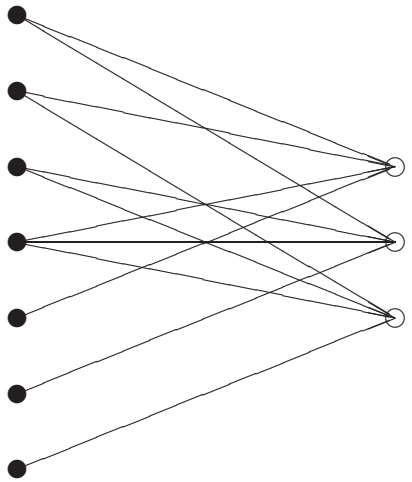
Dual Construction

Hamming code consists of all vectors

$$(x_1, x_2, \ldots, x_7)$$

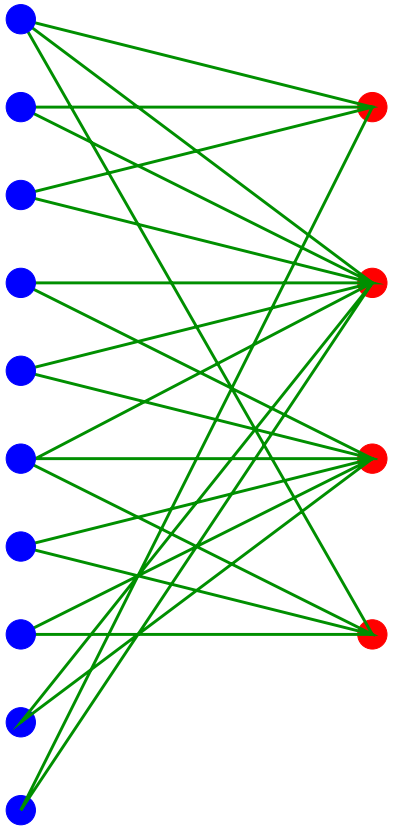
such that

$$\begin{aligned} x_1 + x_2 + x_4 + x_5 &= 0 \\ x_1 + x_3 + x_4 + x_6 &= 0 \\ x_2 + x_3 + x_4 + x_7 &= 0 \end{aligned}$$

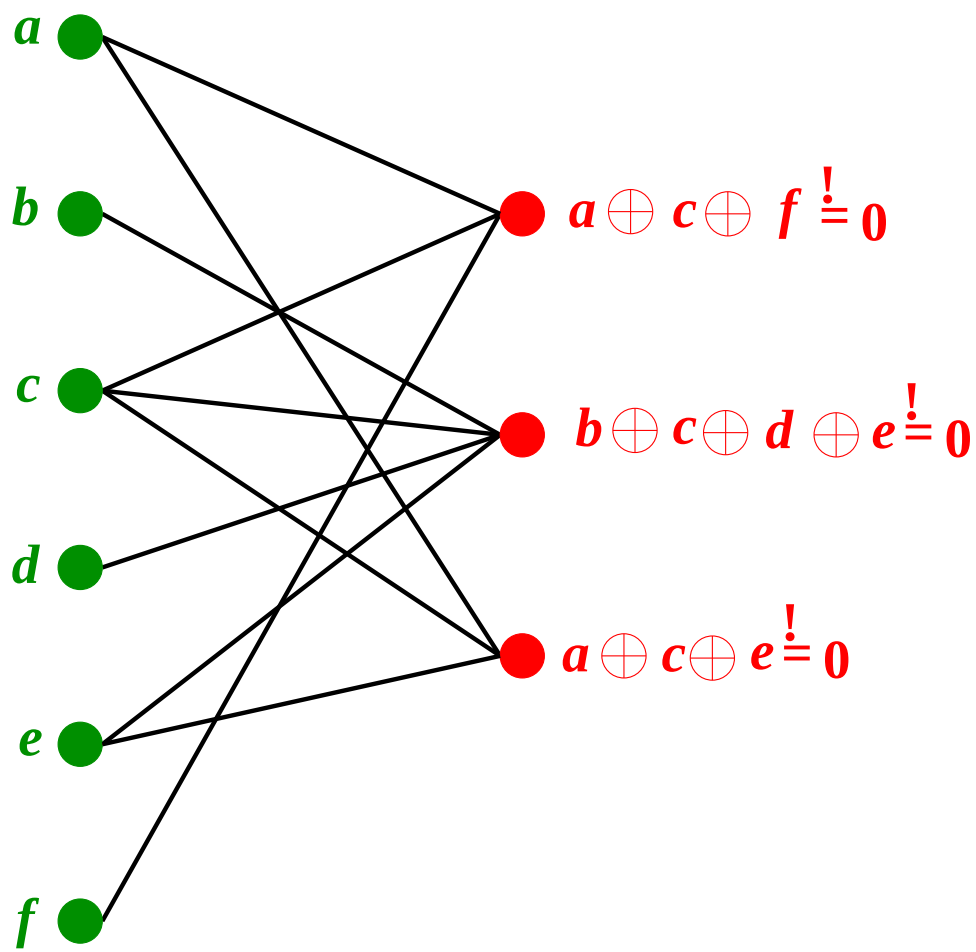


Low-Density Parity-Check Codes

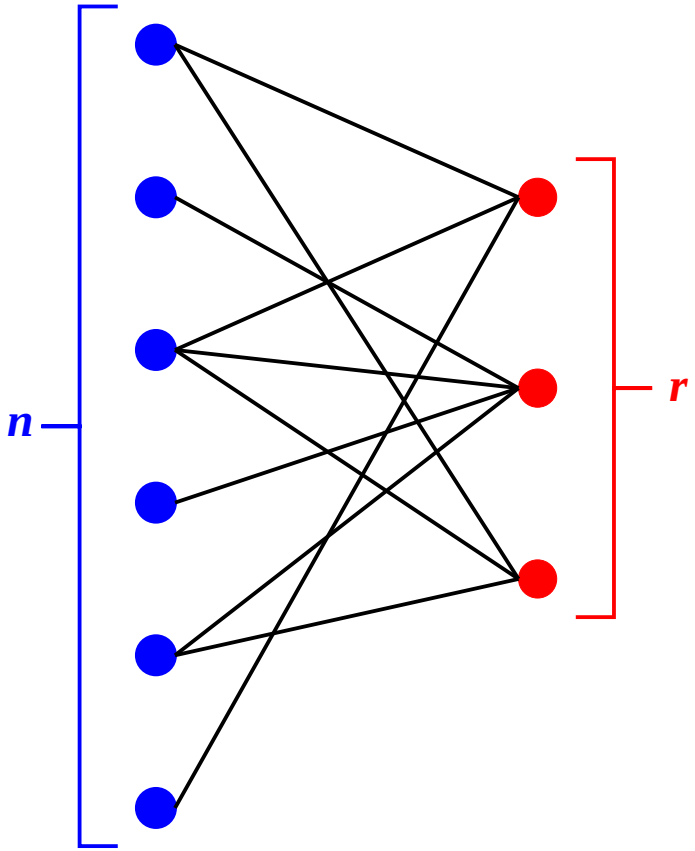
Codes are constructed from sparse bipartite graphs.



Code Construction



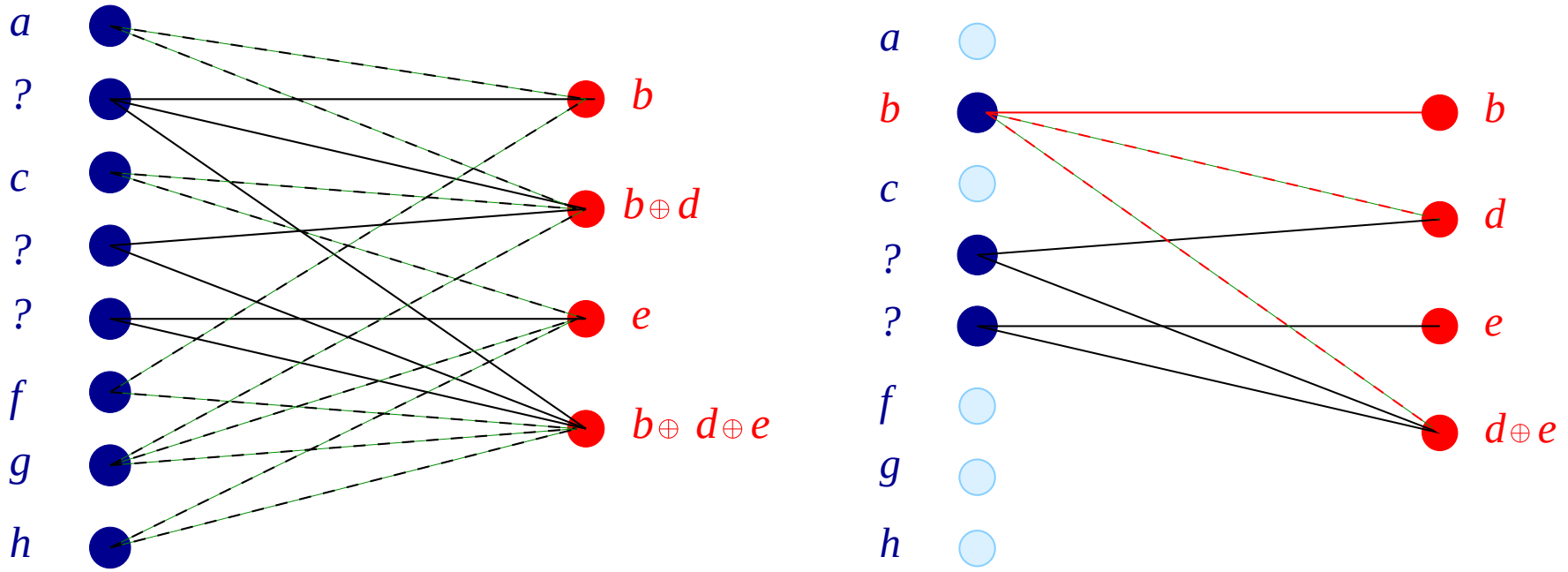
Parameters



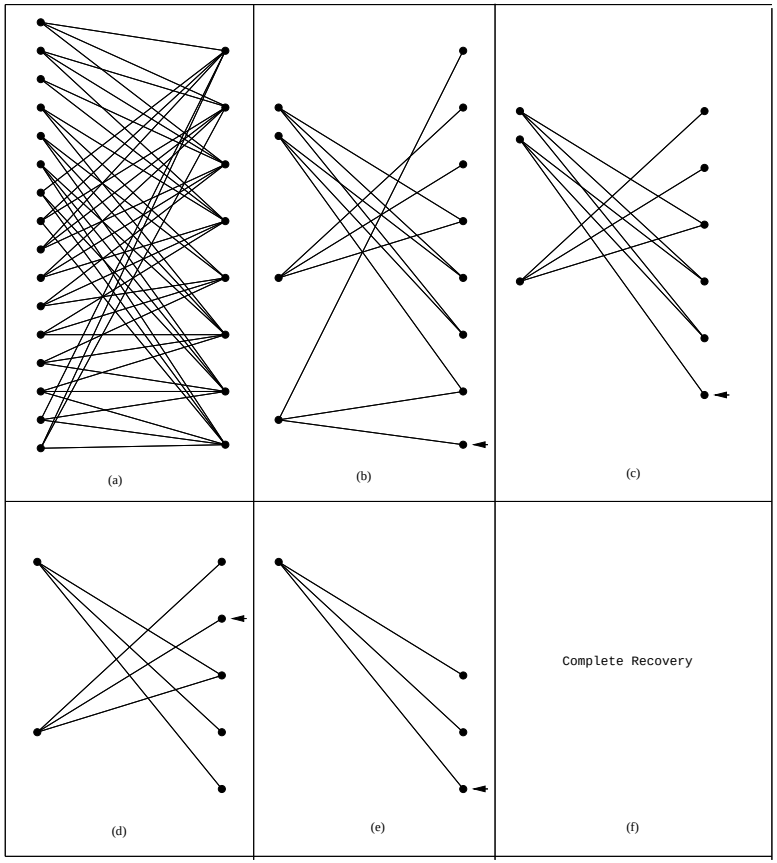
$$Rate \geq \frac{n-r}{n}$$

$$Rate \geq 1 - \frac{average\ left\ degree}{average\ right\ degree}$$

Erasure Decoding



Example



Requirements

Want:

- Small number of edges in graph.
- Be able to correct large fraction of erasures

Full theory to analyze these codes was developed by LMSSS in 1997.

Example

Want code with 10% redundancy and few edges.

Choose biregular graphs with degree 3 on the left and degree 30 on the right.

Analysis gives maximum fraction of $\delta = 0.082835$ correctable errors (OK for very large block lengths).

If we choose 5% erasures we are fine, even for codes of length 1000.

Extremely fast decoding (about 10 times faster than TREVI).

Example

23 kbps stream, 10 seconds latency; gives 230 kb, which is 28750 bytes.

Choose 32 bytes per symbol.

This is roughly 900 symbols.

Add to each packet of 900 symbols another 100 redundant symbols, so that packets are 1000 symbols long.

Can protect from almost any set of 50 erased symbols.

Example

380 kbps, 30 seconds latency, 14% erasure protection.

Choose code with degree 3 on the right, and degree 15 on the left.

Analysis: can protect against 16.75% erasures.

Choose block length 57000 symbols, with 20% redundancy.

Encoding?

$$(x_1, x_2, \ldots, x_n) \cdot \left[\begin{array}{cccc|c} 1 & 0 & \cdots & 0 & \\ * & 1 & \cdots & 0 & \\ \vdots & \vdots & \ddots & \vdots & C \\ * & * & \cdots & 1 & \\ \hline & & D & & E \\ \hline & & & & \\ & & A & & \\ & & & & \\ & & & & \end{array}\right] = 0.$$

Have to compute inverse of E.

Encoding

Computational burden only on server side.

Ongoing research. Here are some results.

- Length 1000 code of rate 1/10: scheduling + inversion of matrix of size 40.

- Length 57000 code of rate 4/5: scheduling + inversion of (sparse) matrix of size 1000. Not feasible at this point.

The higher the rate, the faster the encoder.

Other degree distributions for the graph make encoding much faster.