Efficient Recovery from Multiple Erasures by Accessing Small Number of Disks in Distributed Data Storage

Ganesh R. Kini and Balaji S.B.
Codes and Signal Design Lab
Advisor: Prof Vijay Kumar

17 May 2017

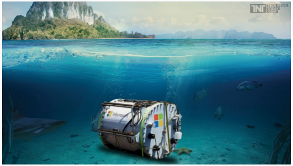
Students' Seminar Series
Department of ECE
Indian Institute of Science

Outline

Codes with Sequential Local Recovery

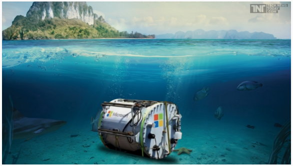
Outline

Distributed Storage



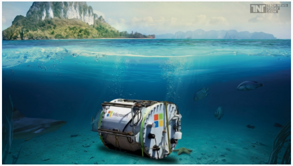
- Data is stored by distributing across disks (nodes).
- Requirements:
  - High reliability i.e. protection from data loss due to disk failures
    want to correct large number of erasures

Distributed Storage

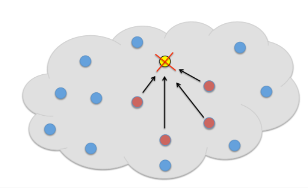


- Data is stored by distributing across disks (nodes).
- Requirements:
  - High reliability i.e. protection from data loss due to disk failures
    want to correct large number of erasures
  - Low storage overhead
    i.e. want high-rate codes

Distributed Storage



- Data is stored by distributing across disks (nodes).
- Requirements:
  - High reliability i.e. protection from data loss due to disk failures
    want to correct large number of erasures
  - Low storage overhead
    i.e. want high-rate codes
  - Efficient repair of a disk when it fails
    want to contact very few surviving nodes



Sequential Recovery

A length 7 code with
code-symbols $c_1, c_2, c_3, ..., c_7$.

Sequential Recovery

A length 7 code with code-symbols $c_1, c_2, c_3, ..., c_7$.
Suppose c_1, c_2 and c_3 (in general some t) are lost.

Sequential Recovery

A length 7 code with code-symbols $c_1, c_2, c_3, ..., c_7$. Suppose c_1, c_2 and c_3 (in general some t) are lost. Suppose can access at most 2 (in general r) other symbols to recover each lost symbol.

Sequential Recovery

A length 7 code with code-symbols $c_1, c_2, c_3, ..., c_7$.

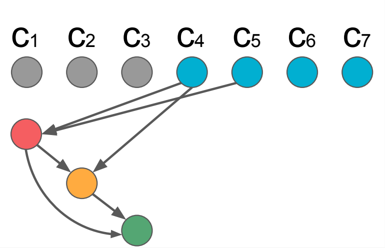
Suppose c_1, c_2 and c_3 (in general some t) are lost.

Suppose can access at most 2 (in general r) other symbols to recover each lost symbol.

Property:

$c_1 = f_1(c_4, c_5)$, $c_2 = f_2(c_1, c_4)$, $c_3 = f_3(c_1, c_2)$

So that, can recover the lost symbols in the sequence $c_1 - -c_2 - -c_3$.



Sequential Recovery

A length 7 code with code-symbols $c_1, c_2, c_3, ..., c_7$.

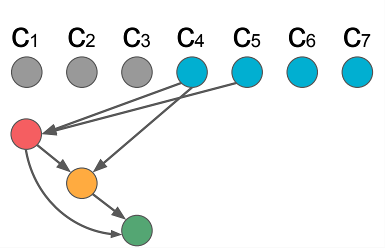
Suppose c_1, c_2 and c_3 (in general some t) are lost.

Suppose can access at most 2 (in general r) other symbols to recover each lost symbol.

Property:

$c_1 = f_1(c_4, c_5)$, $c_2 = f_2(c_1, c_4)$, $c_3 = f_3(c_1, c_2)$

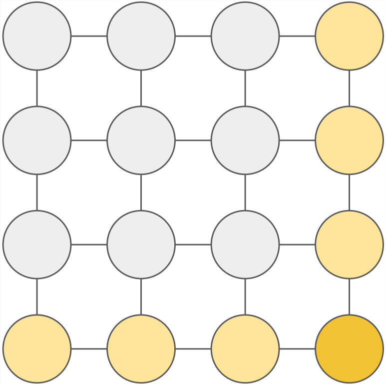
So that, can recover the lost symbols in the sequence $c_1 - -c_2 - -c_3$.



Questions: What is the highest "rate" achievable by such codes? How to design such rate-optimal codes with low blocklength, low field-size?

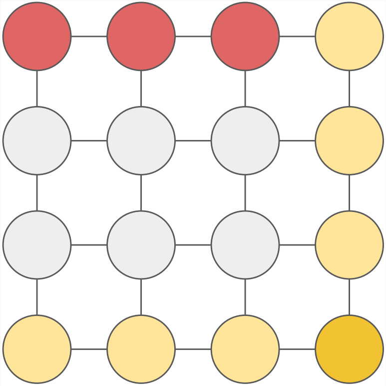
Outline

A Simple Code with Sequential Recovery: 2D Product Code



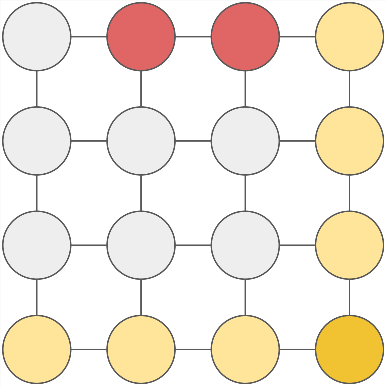
- $(n = 16, k = 9, r = 3, t = 3)_{\text{seq}}$ code
- Rate of the code for general r is $\frac{k}{n} = \frac{r^2}{(r+1)^2}$
- Every row is a codeword of SPC code, every column is a codeword of SPC code
- Parity is the sum of r symbols

A Simple Code with Sequential Recovery: 2D Product Code



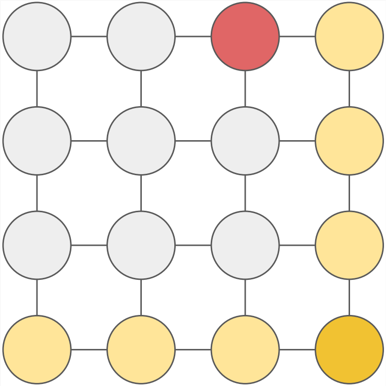
Can correct this erasure-pattern in any sequence

A Simple Code with Sequential Recovery: 2D Product Code



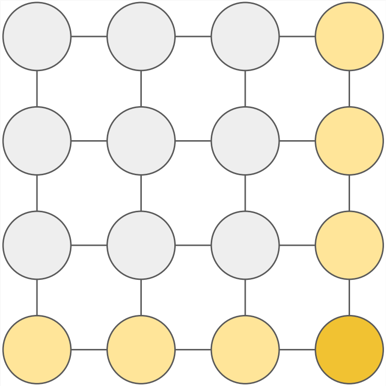
Can correct this erasure-pattern in any sequence

A Simple Code with Sequential Recovery: 2D Product Code



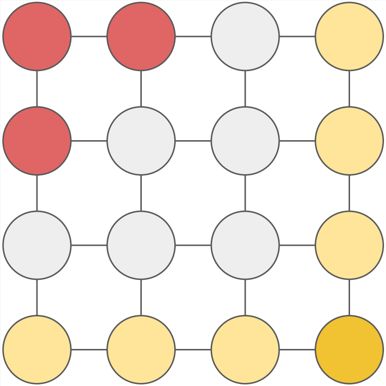
Can correct this erasure-pattern in any sequence

A Simple Code with Sequential Recovery: 2D Product Code



Can it correct all 3-erasure patterns?
In any sequence?

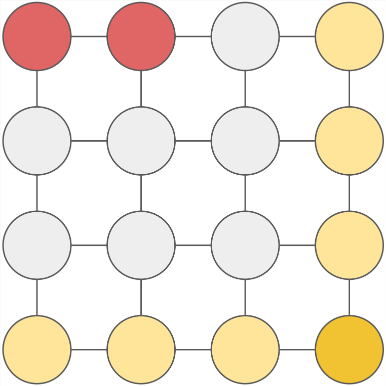
A Simple Code with Sequential Recovery: 2D Product Code



Can it correct all 3-erasure patterns? In any sequence?

Not in any arbitrary sequence; but can correct any 3-erasures

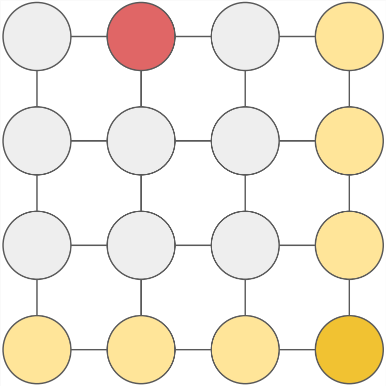
A Simple Code with Sequential Recovery: 2D Product Code



Can it correct all 3-erasure patterns? In any sequence?

Not in any arbitrary sequence; but can correct any 3-erasures

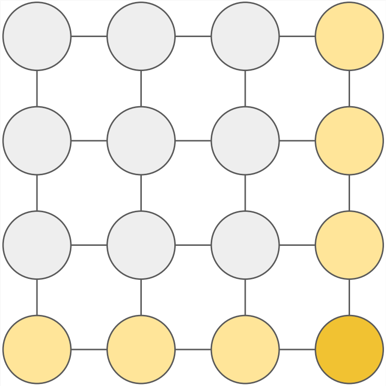
A Simple Code with Sequential Recovery: 2D Product Code



Can it correct all 3-erasure patterns? In any sequence?

Not in any arbitrary sequence; but can correct any 3-erasures

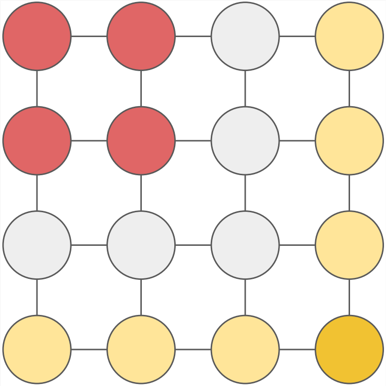
A Simple Code with Sequential Recovery: 2D Product Code



Can it correct all 3-erasure patterns? In any sequence?

Not in any arbitrary sequence; but can correct any 3-erasures

A Simple Code with Sequential Recovery: 2D Product Code



- Hence can correct any 3-erasure-pattern
- But some 4-erasure-patterns are uncorrectable

Definition of Code with Sequential Local Recovery(Sequential LRC)

Definition

Code with Sequential Local Recovery

An $[n, k]$ code is said to be a locally recoverable code with sequential recovery from t erasures, if for any set of $s \leq t$ erasures, there is an s-step sequential recovery process, in which at each step, a single erased symbol is recovered by accessing at most r other code symbols.

Definition of Code with Sequential Local Recovery(Sequential LRC)

Definition

Code with Sequential Local Recovery

An $[n, k]$ code is said to be a locally recoverable code with sequential recovery from t erasures, if for any set of $s \leq t$ erasures, there is an s-step sequential recovery process, in which at each step, a single erased symbol is recovered by accessing at most r other code symbols.

This is equivalent to the requirement that for any set of $s \leq t$ erasures, the dual code contain a codeword whose support contains the coordinate of precisely one of the s erased symbols.

Definition of Code with Sequential Local Recovery(Sequential LRC)

Definition

Code with Sequential Local Recovery

An $[n, k]$ code is said to be a locally recoverable code with sequential recovery from t erasures, if for any set of $s \leq t$ erasures, there is an s-step sequential recovery process, in which at each step, a single erased symbol is recovered by accessing at most r other code symbols.

This is equivalent to the requirement that for any set of $s \leq t$ erasures, the dual code contain a codeword whose support contains the coordinate of precisely one of the s erased symbols.

We will formally refer to this class of codes as $(n, k, r, t)_{seq}$ codes.

Outline

Question of Code-Rate

Given locality parameter r and erasure correctability parameter t, what is the maximum achievable code-rate?

Question of Code-Rate

Given locality parameter r and erasure correctability parameter t, what is the maximum achievable code-rate?

Theorem

Rate Bound[1] *: Let $\mathcal{C}$ be an $(n, k, r, t)_{seq}$ code over a field $\mathbb{F}_q$. Let $r \geq 3$. Then*

$$\frac{k}{n} \leq \frac{r^{\frac{t}{2}}}{r^{\frac{t}{2}} + 2\sum_{i=0}^{\frac{t}{2}-1} r^i} \quad \text{for even } t, \qquad (1)$$

$$\frac{k}{n} \leq \frac{r^s}{r^s + 2\sum_{i=1}^{s-1} r^i + 1} \quad \text{for odd } t, \qquad (2)$$

where $s = \frac{t+1}{2}$.

[1] S. B. Balaji, G. R. Kini, and P. V. Kumar, "A tight rate bound and a matching construction for locally recoverable codes with sequential recovery from any number of multiple erasures, 2017. [Online]. Available: http://arxiv.org/abs/1611.08561

Question of Code-Rate

Given locality parameter r and erasure correctability parameter t, what is the maximum achievable code-rate?

Theorem

Rate Bound*: Let $\mathcal{C}$ be an $(n, k, r, t)_{seq}$ code over a field $\mathbb{F}_q$. Let $r \geq 3$. Then*

$$\frac{k}{n} \leq \frac{r^{\frac{t}{2}}}{r^{\frac{t}{2}} + 2\sum_{i=0}^{\frac{t}{2}-1} r^i} \quad \text{for even } t, \qquad (1)$$

$$\frac{k}{n} \leq \frac{r^s}{r^s + 2\sum_{i=1}^{s-1} r^i + 1} \quad \text{for odd } t, \qquad (2)$$

where $s = \frac{t+1}{2}$.

Proof.

We investigate the structure of the parity check matrix

Parity-Check Matrix

- Let $\mathcal{S} = \text{span}(\underline{c} \in \mathcal{C}^\perp : w_H(\underline{c}) \leq r + 1)$, where $\underline{c}$ is a row-vector

Parity-Check Matrix

- Let $\mathcal{S} = \text{span}(\underline{c} \in \mathcal{C}^{\perp} : w_H(\underline{c}) \leq r + 1)$, where $\underline{c}$ is a row-vector
- Let m be dimension of $\mathcal{S}$, and $\underline{c_1}, ..., \underline{c_m}$ be a basis of $\mathcal{S}$ s.t. $w_H(\underline{c_i}) \leq r + 1$

Parity-Check Matrix

- Let $\mathcal{S} = \text{span}(\underline{c} \in \mathcal{C}^{\perp} : w_H(\underline{c}) \leq r + 1)$, where $\underline{c}$ is a row-vector
- Let m be dimension of $\mathcal{S}$, and $\underline{c_1}, ..., \underline{c_m}$ be a basis of $\mathcal{S}$ s.t. $w_H(\underline{c_i}) \leq r + 1$
- Let $H_1 = \begin{bmatrix} \underline{c_1} \\ \underline{c_2} \\ \vdots \\ \underline{c_m} \end{bmatrix}$

Parity-Check Matrix

- Let $\mathcal{S} = \text{span}(\underline{c} \in \mathcal{C}^\perp : w_H(\underline{c}) \leq r+1)$, where $\underline{c}$ is a row-vector
- Let m be dimension of $\mathcal{S}$, and $\underline{c_1}, ..., \underline{c_m}$ be a basis of $\mathcal{S}$ s.t. $w_H(\underline{c_i}) \leq r+1$
- Let $H_1 = \begin{bmatrix} \underline{c_1} \\ \underline{c_2} \\ \vdots \\ \underline{c_m} \end{bmatrix}$
- H_1 is a parity-check matrix of an $(n, n-m, r, t)_{seq}$ code

Parity Check Matrix of a Linear Code

Suppose an n-length code has code-symbols $c_1, ..., c_n$.
The rows of a parity check matrix of the code are nothing but the linear equations that the code-symbols satisfy.

Suppose $[a_1, ..., a_n]$ is one row, then $\sum_{i=1}^{n} a_i c_i = 0$

If the dimension of the code is k, then the parity check matrices have rank $n - k$.

P-C Matrix for Sequentially Correcting t Erasures Locally

We'll now see the case of even t

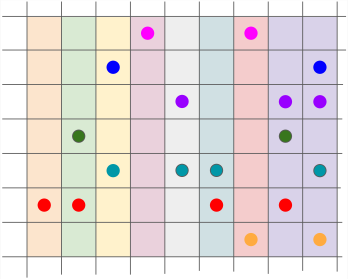
P-C Matrix for Sequentially Correcting t Erasures Locally

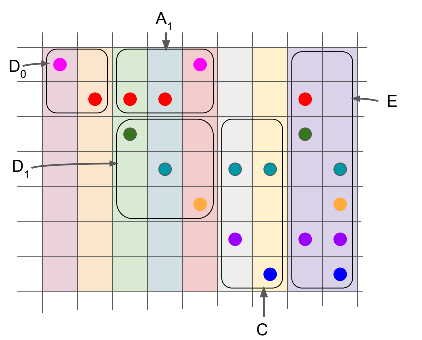
Start with any (n, k, r, t) code, consider the matrix H_1 for it, with row and column permutations it looks like this:

$$H_1 = \left[\begin{array}{c|c|c|c|c|c|c|c|c}
D_0 & A_1 & 0 & 0 & \dots & 0 & 0 & 0 & \\
\cline{1-8}
0 & D_1 & A_2 & 0 & \dots & 0 & 0 & 0 & \\
\cline{1-8}
0 & 0 & D_2 & A_3 & \dots & 0 & 0 & 0 & \\
\cline{1-8}
0 & 0 & 0 & D_3 & \dots & 0 & 0 & 0 & \\
\cline{1-8}
\vdots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots & E \\
\cline{1-8}
0 & 0 & 0 & 0 & \dots & A_{\frac{t}{2}-2} & 0 & 0 & \\
\cline{1-8}
0 & 0 & 0 & 0 & \dots & D_{\frac{t}{2}-2} & A_{\frac{t}{2}-1} & 0 & \\
\cline{1-8}
0 & 0 & 0 & 0 & \dots & 0 & D_{\frac{t}{2}-1} & & \\
\cline{1-7}
0 & 0 & 0 & 0 & \dots & 0 & 0 & C &
\end{array}\right]$$

P-C Matrix: Row and Column Permutation

Take H_1 matrix of any $(n, k, r, t)_{seq}$ code. Permute rows and columns to get the staircase form:





P-C Matrix for Sequentially Correcting t Erasures Locally

$$H_1 = \left[\begin{array}{c|c|c|c|c|c|c|c|c}
D_0 & A_1 & 0 & 0 & \dots & 0 & 0 & 0 & \\ \hline
0 & D_1 & A_2 & 0 & \dots & 0 & 0 & 0 & \\ \hline
0 & 0 & D_2 & A_3 & \dots & 0 & 0 & 0 & \\ \hline
0 & 0 & 0 & D_3 & \dots & 0 & 0 & 0 & \\ \hline
\vdots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots & E \\ \hline
0 & 0 & 0 & 0 & \dots & A_{\frac{t}{2}-2} & 0 & 0 & \\ \hline
0 & 0 & 0 & 0 & \dots & D_{\frac{t}{2}-2} & A_{\frac{t}{2}-1} & 0 & \\ \hline
0 & 0 & 0 & 0 & \dots & 0 & D_{\frac{t}{2}-1} & & \\ \hline
0 & 0 & 0 & 0 & \dots & 0 & 0 & C &
\end{array}\right]$$

- A_i's are $\rho_{i-1} \times a_i$ and D_i's are $\rho_i \times a_i$ for some ρ_i's and a_i's

P-C Matrix for Sequentially Correcting t Erasures Locally

$$H_1 = \left[\begin{array}{c|c|c|c|c|c|c|c|c}
D_0 & A_1 & 0 & 0 & \dots & 0 & 0 & 0 & \\
\hline
0 & D_1 & A_2 & 0 & \dots & 0 & 0 & 0 & \\
\hline
0 & 0 & D_2 & A_3 & \dots & 0 & 0 & 0 & \\
\hline
0 & 0 & 0 & D_3 & \dots & 0 & 0 & 0 & \\
\hline
\vdots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots & E \\
\hline
0 & 0 & 0 & 0 & \dots & A_{\frac{t}{2}-2} & 0 & 0 & \\
\hline
0 & 0 & 0 & 0 & \dots & D_{\frac{t}{2}-2} & A_{\frac{t}{2}-1} & 0 & \\
\hline
0 & 0 & 0 & 0 & \dots & 0 & D_{\frac{t}{2}-1} & & \\
\cline{1-7}
0 & 0 & 0 & 0 & \dots & 0 & 0 & C &
\end{array}\right]$$

- A_i's are $\rho_{i-1} \times a_i$ and D_i's are $\rho_i \times a_i$ for some ρ_i's and a_i's
- D_0: columns have weight 1 and rows have weight at least 1

P-C Matrix for Sequentially Correcting t Erasures Locally

$$
H_1 = \left[\begin{array}{c|c|c|c|c|c|c|c|c}
D_0 & A_1 & 0 & 0 & \dots & 0 & 0 & 0 & \\
\hline
0 & D_1 & A_2 & 0 & \dots & 0 & 0 & 0 & \\
\hline
0 & 0 & D_2 & A_3 & \dots & 0 & 0 & 0 & \\
\hline
0 & 0 & 0 & D_3 & \dots & 0 & 0 & 0 & \\
\hline
\vdots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots & E \\
\hline
0 & 0 & 0 & 0 & \dots & A_{\frac{t}{2}-2} & 0 & 0 & \\
\hline
0 & 0 & 0 & 0 & \dots & D_{\frac{t}{2}-2} & A_{\frac{t}{2}-1} & 0 & \\
\hline
0 & 0 & 0 & 0 & \dots & 0 & D_{\frac{t}{2}-1} & & \\
\hline
0 & 0 & 0 & 0 & \dots & 0 & 0 & C &
\end{array}\right]
$$

- A_i's are $\rho_{i-1} \times a_i$ and D_i's are $\rho_i \times a_i$ for some ρ_i's and a_i's
- D_0: columns have weight 1 and rows have weight at least 1
- $\begin{bmatrix} A_i \\ D_i \end{bmatrix}$: for $i \geq 1$, columns have weight 2

P-C Matrix for Sequentially Correcting t Erasures Locally

$$H_1 = \left[\begin{array}{c|c|c|c|c|c|c|c|c}
D_0 & A_1 & 0 & 0 & \dots & 0 & 0 & 0 & \\
\hline
0 & D_1 & A_2 & 0 & \dots & 0 & 0 & 0 & \\
\hline
0 & 0 & D_2 & A_3 & \dots & 0 & 0 & 0 & \\
\hline
0 & 0 & 0 & D_3 & \dots & 0 & 0 & 0 & \\
\hline
\vdots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots & E \\
\hline
0 & 0 & 0 & 0 & \dots & A_{\frac{t}{2}-2} & 0 & 0 & \\
\hline
0 & 0 & 0 & 0 & \dots & D_{\frac{t}{2}-2} & A_{\frac{t}{2}-1} & 0 & \\
\hline
0 & 0 & 0 & 0 & \dots & 0 & D_{\frac{t}{2}-1} & & \\
\hline
0 & 0 & 0 & 0 & \dots & 0 & 0 & C &
\end{array}\right]$$

- A_i's are $\rho_{i-1} \times a_i$ and D_i's are $\rho_i \times a_i$ for some ρ_i's and a_i's
- D_0: columns have weight 1 and rows have weight at least 1
- $\begin{bmatrix} A_i \\ D_i \end{bmatrix}$: for $i \geq 1$, columns have weight 2
- A_i: columns have weight at least 1

P-C Matrix for Sequentially Correcting t Erasures Locally

$$H_1 = \left[\begin{array}{c|c|c|c|c|c|c|c|c}
D_0 & A_1 & 0 & 0 & \dots & 0 & 0 & 0 & \\ \hline
0 & D_1 & A_2 & 0 & \dots & 0 & 0 & 0 & \\ \hline
0 & 0 & D_2 & A_3 & \dots & 0 & 0 & 0 & \\ \hline
0 & 0 & 0 & D_3 & \dots & 0 & 0 & 0 & \\ \hline
\vdots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots & E \\ \hline
0 & 0 & 0 & 0 & \dots & A_{\frac{t}{2}-2} & 0 & 0 & \\ \hline
0 & 0 & 0 & 0 & \dots & D_{\frac{t}{2}-2} & A_{\frac{t}{2}-1} & 0 & \\ \hline
0 & 0 & 0 & 0 & \dots & 0 & D_{\frac{t}{2}-1} & & \\ \cline{1-7}
0 & 0 & 0 & 0 & \dots & 0 & 0 & C &
\end{array}\right]$$

- A_i's are $\rho_{i-1} \times a_i$ and D_i's are $\rho_i \times a_i$ for some ρ_i's and a_i's
- D_0: columns have weight 1 and rows have weight at least 1
- $\begin{bmatrix} A_i \\ D_i \end{bmatrix}$: for $i \geq 1$, columns have weight 2
- A_i: columns have weight at least 1
- D_i: rows have weight at least 1 and columns have weight at most 1

P-C Matrix for Sequentially Correcting t Erasures Locally

$$H_1 = \left[\begin{array}{c|c|c|c|c|c|c|c|c}
D_0 & A_1 & 0 & 0 & \dots & 0 & 0 & 0 & \\
0 & D_1 & A_2 & 0 & \dots & 0 & 0 & 0 & \\
0 & 0 & D_2 & A_3 & \dots & 0 & 0 & 0 & \\
0 & 0 & 0 & D_3 & \dots & 0 & 0 & 0 & \\
\vdots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots & E \\
0 & 0 & 0 & 0 & \dots & A_{\frac{t}{2}-2} & 0 & 0 & \\
0 & 0 & 0 & 0 & \dots & D_{\frac{t}{2}-2} & A_{\frac{t}{2}-1} & 0 & \\
0 & 0 & 0 & 0 & \dots & 0 & D_{\frac{t}{2}-1} & & \\
0 & 0 & 0 & 0 & \dots & 0 & 0 & C &
\end{array}\right]$$

- A_i's are $\rho_{i-1} \times a_i$ and D_i's are $\rho_i \times a_i$ for some ρ_i's and a_i's
- D_0: columns have weight 1 and rows have weight at least 1
- $\begin{bmatrix} A_i \\ D_i \end{bmatrix}$: for $i \geq 1$, columns have weight 2
- A_i: columns have weight at least 1
- D_i: rows have weight at least 1 and columns have weight at most 1
- C: columns have weight exactly 2

P-C Matrix for Sequentially Correcting t Erasures Locally

$$H_1 = \left[\begin{array}{c|c|c|c|c|c|c|c|c}
D_0 & A_1 & 0 & 0 & \ldots & 0 & 0 & 0 & \\
\hline
0 & D_1 & A_2 & 0 & \ldots & 0 & 0 & 0 & \\
\hline
0 & 0 & D_2 & A_3 & \ldots & 0 & 0 & 0 & \\
\hline
0 & 0 & 0 & D_3 & \ldots & 0 & 0 & 0 & \\
\hline
\vdots & \vdots & \vdots & \vdots & \ldots & \vdots & \vdots & \vdots & E \\
\hline
0 & 0 & 0 & 0 & \ldots & A_{\frac{t}{2}-2} & 0 & 0 & \\
\hline
0 & 0 & 0 & 0 & \ldots & D_{\frac{t}{2}-2} & A_{\frac{t}{2}-1} & 0 & \\
\hline
0 & 0 & 0 & 0 & \ldots & 0 & D_{\frac{t}{2}-1} & & \\
\cline{1-7}
0 & 0 & 0 & 0 & \ldots & 0 & 0 & C &
\end{array}\right]$$

- A_i's are $\rho_{i-1} \times a_i$ and D_i's are $\rho_i \times a_i$ for some ρ_i's and a_i's
- D_0: columns have weight 1 and rows have weight at least 1
- $\begin{bmatrix} A_i \\ D_i \end{bmatrix}$: for $i \geq 1$, columns have weight 2
- A_i: columns have weight at least 1
- D_i: rows have weight at least 1 and columns have weight at most 1
- C: columns have weight exactly 2
- E: columns have weight at least 3

P-C Matrix: Structure

Claim

A_i's are matrices with each column having weight 1 and D_i's are matrices with each row and each column having weight 1.

P-C Matrix: Structure

Claim

A_i's are matrices with each column having weight 1 and D_i's are matrices with each row and each column having weight 1.

Proof.

Fact: $d_{min}(\mathcal{C}) \geq t + 1$; hence no $x(\leq t)$ columns of H_1 can be linearly dependent.

P-C Matrix: Structure

Claim

A_i's are matrices with each column having weight 1 and D_i's are matrices with each row and each column having weight 1.

Proof.

Fact: $d_{min}(\mathcal{C}) \geq t + 1$; hence no $x(\leq t)$ columns of H_1 can be linearly dependent.

Will show that columns of A_i have weight exactly 1 and rows of D_i have weight exactly 1.

P-C Matrix: Structure

Claim

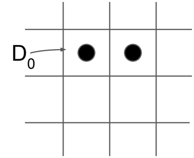
A_i's are matrices with each column having weight 1 and D_i's are matrices with each row and each column having weight 1.

Proof.

Fact: $d_{min}(\mathcal{C}) \geq t+1$; hence no $x(\leq t)$ columns of H_1 can be linearly dependent.

Will show that columns of A_i have weight exactly 1 and rows of D_i have weight exactly 1.

If D_0 has a row with at least 2 non-zero entries, then 2 columns become linearly dependent, a contradiction to $d_{min}(\mathcal{C}) \geq t+1$.



P-C Matrix: Structure

Claim

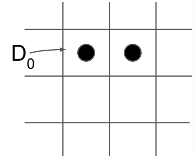
A_i's are matrices with each column having weight 1 and D_i's are matrices with each row and each column having weight 1.

Proof.

Fact: $d_{min}(\mathcal{C}) \geq t + 1$; hence no $x(\leq t)$ columns of H_1 can be linearly dependent.

Will show that columns of A_i have weight exactly 1 and rows of D_i have weight exactly 1.

If D_0 has a row with at least 2 non-zero entries, then 2 columns become linearly dependent, a contradiction to $d_{min}(\mathcal{C}) \geq t + 1$.



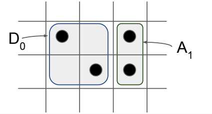
Therefore, with column permutation, D_0 is diagonal.

P-C Matrix: Structure

A_1: columns have weight exactly 1

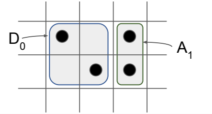
P-C Matrix: Structure

A_1: columns have weight exactly 1



P-C Matrix: Structure

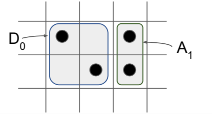
A_1: columns have weight exactly 1



3 columns linearly dependent.

P-C Matrix: Structure

A_1: columns have weight exactly 1

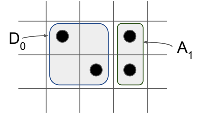


3 columns linearly dependent.

D_2: rows have weight exactly 1

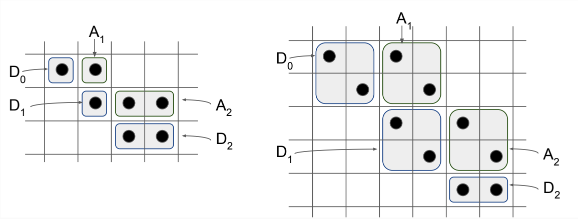
P-C Matrix: Structure

A_1: columns have weight exactly 1



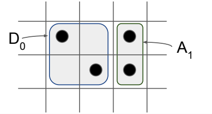
3 columns linearly dependent.

D_2: rows have weight exactly 1



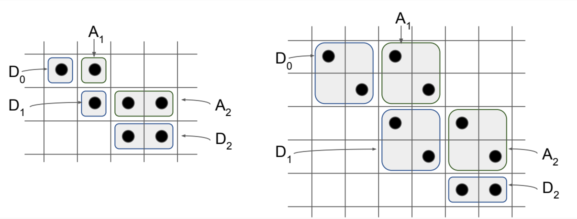
P-C Matrix: Structure

A_1: columns have weight exactly 1



3 columns linearly dependent.

D_2: rows have weight exactly 1



Upto 6 columns linearly dependent.

P-C Matrix: Structure

For some $0 \leq i \leq \frac{t}{2} - 1$, upto $2(i+1) \leq t$ columns become linearly dependent, which is a contradiction to $d_{min}(\mathcal{C}) \geq t + 1$.

Thus, the Claim is true. i.e.
A_i's are matrices with each column having weight 1 and D_i's are matrices with each row and each column having weight 1.

Therefore, D_i's are diagonal (identity, after scaling) matrices with number of rows ρ_i and number of columns a_i equal.

$$\rho_i = a_i$$

Let's recall...

$$
m \quad
\begin{array}{c}
\\
a_0 \\ a_1 \\ \\ \\ \\ \\ \\ a_{t/2-1} \\ p
\end{array}
\quad
H_1 =
\begin{array}{c}
\begin{array}{cccccccc}
a_0 & a_1 & & & & & a_{t/2-1} & a_{t/2}
\end{array}
\\
\left[
\begin{array}{cccccccc|c}
D_0 & A_1 & 0 & 0 & \dots & 0 & 0 & 0 & \\
0 & D_1 & A_2 & 0 & \dots & 0 & 0 & 0 & \\
0 & 0 & D_2 & A_3 & \dots & 0 & 0 & 0 & \\
0 & 0 & 0 & D_3 & \dots & 0 & 0 & 0 & \\
\vdots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots & E \\
0 & 0 & 0 & 0 & \dots & A_{\frac{t}{2}-2} & 0 & 0 & \\
0 & 0 & 0 & 0 & \dots & D_{\frac{t}{2}-2} & A_{\frac{t}{2}-1} & 0 & \\
0 & 0 & 0 & 0 & \dots & 0 & D_{\frac{t}{2}-1} & & \\
0 & 0 & 0 & 0 & \dots & 0 & 0 & C &
\end{array}
\right]
\\
n
\end{array}
$$

Now we count...

Equating sum of row-weights and sum of column-weights of A_i:

$$
H_1 = \begin{array}{cc}
 & \begin{array}{ccccccccc} \quad & \quad & \quad & a_3 & \quad & \quad & \quad & \quad & \quad \end{array} \\
\begin{array}{c} \\ \\ a_2 \\ \\ \\ \\ \\ \\ \\ \end{array} &
\left[\begin{array}{c|c|c|c|c|c|c|c|c}
D_0 & A_1 & 0 & 0 & \dots & 0 & 0 & 0 & \\
\hline
0 & D_1 & A_2 & 0 & \dots & 0 & 0 & 0 & \\
\hline
0 & 0 & D_2 & A_3 & \dots & 0 & 0 & 0 & \\
\hline
0 & 0 & 0 & D_3 & \dots & 0 & 0 & 0 & \\
\hline
\vdots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots & E \\
\hline
0 & 0 & 0 & 0 & \dots & A_{\frac{t}{2}-2} & 0 & 0 & \\
\hline
0 & 0 & 0 & 0 & \dots & D_{\frac{t}{2}-2} & A_{\frac{t}{2}-1} & 0 & \\
\hline
0 & 0 & 0 & 0 & \dots & 0 & D_{\frac{t}{2}-1} & & \\
\cline{1-7}
0 & 0 & 0 & 0 & \dots & 0 & 0 & C &
\end{array}\right]
\end{array}
$$

$$a_{i-1} r \geq a_i$$

Now we count...

Equating sum of row-weights and sum of column-weights of C:

$$H_1 = \begin{array}{ll}
 & \begin{array}{cccccccccc} & & & & & \phantom{A_{\frac{t}{2}-2}} & a_{t/2-1} & a_{t/2} & \end{array} \\
\begin{array}{r} \\ \\ \\ \\ \\ \\ \\ a_{t/2-1} \\ p \end{array} & \left[\begin{array}{cccccccc|c}
D_0 & A_1 & 0 & 0 & \dots & 0 & 0 & 0 & \\
0 & D_1 & A_2 & 0 & \dots & 0 & 0 & 0 & \\
0 & 0 & D_2 & A_3 & \dots & 0 & 0 & 0 & \\
0 & 0 & 0 & D_3 & \dots & 0 & 0 & 0 & \\
\vdots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots & E \\
0 & 0 & 0 & 0 & \dots & A_{\frac{t}{2}-2} & 0 & 0 & \\
0 & 0 & 0 & 0 & \dots & D_{\frac{t}{2}-2} & A_{\frac{t}{2}-1} & 0 & \\
0 & 0 & 0 & 0 & \dots & 0 & D_{\frac{t}{2}-1} & & \\
0 & 0 & 0 & 0 & \dots & 0 & 0 & C &
\end{array}\right]
\end{array}$$

$$2a_{\frac{t}{2}} \leq (a_{\frac{t}{2}-1} + p)(r+1) - a_{\frac{t}{2}-1}$$

Now we count...

Equating sum of row-weights and sum of column-weights of H_1:

$$
H_1 = \left[\begin{array}{c|c|c|c|c|c|c|c|c}
D_0 & A_1 & 0 & 0 & \dots & 0 & 0 & 0 & \\ \hline
0 & D_1 & A_2 & 0 & \dots & 0 & 0 & 0 & \\ \hline
0 & 0 & D_2 & A_3 & \dots & 0 & 0 & 0 & \\ \hline
0 & 0 & 0 & D_3 & \dots & 0 & 0 & 0 & \\ \hline
\vdots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots & E \\ \hline
0 & 0 & 0 & 0 & \dots & A_{\frac{t}{2}-2} & 0 & 0 & \\ \hline
0 & 0 & 0 & 0 & \dots & D_{\frac{t}{2}-2} & A_{\frac{t}{2}-1} & 0 & \\ \hline
0 & 0 & 0 & 0 & \dots & 0 & D_{\frac{t}{2}-1} & & \\ \cline{1-7}
0 & 0 & 0 & 0 & \dots & 0 & 0 & C &
\end{array}\right]
$$

$$
m(r+1) \geq a_0 + 2\left(\sum_{i=1}^{\frac{t}{2}} a_i\right) + 3\left(n - \sum_{i=0}^{\frac{t}{2}} a_i\right)
$$

The Inequalities

We now have the following set of inequalities:

$$a_{i-1}r \geq a_i \tag{3}$$

$$2a_{\frac{t}{2}} \leq (a_{\frac{t}{2}-1} + p)(r+1) - a_{\frac{t}{2}-1} \tag{4}$$

$$m(r+1) \geq a_0 + 2(\sum_{i=1}^{\frac{t}{2}} a_i) + 3(n - \sum_{i=0}^{\frac{t}{2}} a_i) \tag{5}$$

The Inequalities

We now have the following set of inequalities:

$$a_{i-1}r \geq a_i \tag{3}$$

$$2a_{\frac{t}{2}} \leq (a_{\frac{t}{2}-1} + p)(r+1) - a_{\frac{t}{2}-1} \tag{4}$$

$$m(r+1) \geq a_0 + 2(\sum_{i=1}^{\frac{t}{2}} a_i) + 3(n - \sum_{i=0}^{\frac{t}{2}} a_i) \tag{5}$$

Also,

$$\sum_{i=0}^{\frac{t}{2}-1} a_i + p = m \tag{6}$$

The Inequalities

We now have the following set of inequalities:

$$a_{i-1}r \geq a_i \tag{3}$$

$$2a_{\frac{t}{2}} \leq (a_{\frac{t}{2}-1} + p)(r+1) - a_{\frac{t}{2}-1} \tag{4}$$

$$m(r+1) \geq a_0 + 2(\sum_{i=1}^{\frac{t}{2}} a_i) + 3(n - \sum_{i=0}^{\frac{t}{2}} a_i) \tag{5}$$

Also,

$$\sum_{i=0}^{\frac{t}{2}-1} a_i + p = m \tag{6}$$

Now we obtain a lower bound on m

Upper Bound on Rate

- either by manipulating the inequalities

Upper Bound on Rate

- either by manipulating the inequalities
- or by observing that the inequalities are linear in $a_0, ..., a_{\frac{t}{2}}, p$; hence formulating a linear programming problem

Upper Bound on Rate

- either by manipulating the inequalities
- or by observing that the inequalities are linear in $a_0, ..., a_{\frac{t}{2}}, p$; hence formulating a linear programming problem

We obtain:

$$m \geq \frac{2n \sum_{i=0}^{\frac{t}{2}-1} r^i}{r^{\frac{t}{2}} + 2 \sum_{i=0}^{\frac{t}{2}-1} r^i} \tag{7}$$

Upper Bound on Rate

- either by manipulating the inequalities
- or by observing that the inequalities are linear in $a_0, ..., a_{\frac{t}{2}}, p$; hence formulating a linear programming problem

We obtain:

$$m \geq \frac{2n \sum_{i=0}^{\frac{t}{2}-1} r^i}{r^{\frac{t}{2}} + 2 \sum_{i=0}^{\frac{t}{2}-1} r^i} \tag{7}$$

Now, $n - k \geq m$
(Recall m is the number of independent "local" parity checks only)

Upper Bound on Rate

- either by manipulating the inequalities
- or by observing that the inequalities are linear in $a_0, ..., a_{\frac{t}{2}}, p$; hence formulating a linear programming problem

We obtain:

$$m \geq \frac{2n\sum_{i=0}^{\frac{t}{2}-1} r^i}{r^{\frac{t}{2}} + 2\sum_{i=0}^{\frac{t}{2}-1} r^i} \tag{7}$$

Now, $n - k \geq m$

(Recall m is the number of independent "local" parity checks only)

Therefore we get,

$$\frac{k}{n} \leq 1 - \frac{m}{n} \leq \frac{r^{\frac{t}{2}}}{r^{\frac{t}{2}} + 2\sum_{i=0}^{\frac{t}{2}-1} r^i} \tag{8}$$

Upper Bound on Rate

- either by manipulating the inequalities
- or by observing that the inequalities are linear in $a_0, ..., a_{\frac{t}{2}}, p$; hence formulating a linear programming problem

We obtain:

$$m \geq \frac{2n\sum_{i=0}^{\frac{t}{2}-1} r^i}{r^{\frac{t}{2}} + 2\sum_{i=0}^{\frac{t}{2}-1} r^i} \tag{7}$$

Now, $n - k \geq m$

(Recall m is the number of independent “local” parity checks only)

Therefore we get,

$$\frac{k}{n} \leq 1 - \frac{m}{n} \leq \frac{r^{\frac{t}{2}}}{r^{\frac{t}{2}} + 2\sum_{i=0}^{\frac{t}{2}-1} r^i} \tag{8}$$

Proof for odd t proceeds along similar lines

Conditions for Equality

- $a_i = \frac{2nr^i}{r^{\frac{t}{2}} + 2\sum_{j=0}^{\frac{t}{2}-1} r^j}$, for $0 \leq i \leq \frac{t}{2} - 1$,
- $a_{\frac{t}{2}} = \frac{nr^{\frac{t}{2}}}{r^{\frac{t}{2}} + 2\sum_{j=0}^{\frac{t}{2}-1} r^j}$,
- $p = 0$
- Note that $\sum_{i=0}^{\frac{t}{2}} a_i = n$, therefore E is an empty matrix.

The parity-check matrix then is

$$H_1 = \left[\begin{array}{c|c|c|c|c|c|c|c}
D_0 & A_1 & 0 & 0 & \dots & 0 & 0 & 0 \\ \hline
0 & D_1 & A_2 & 0 & \dots & 0 & 0 & 0 \\ \hline
0 & 0 & D_2 & A_3 & \dots & 0 & 0 & 0 \\ \hline
0 & 0 & 0 & D_3 & \dots & 0 & 0 & 0 \\ \hline
\vdots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots \\ \hline
0 & 0 & 0 & 0 & \dots & A_{\frac{t}{2}-2} & 0 & 0 \\ \hline
0 & 0 & 0 & 0 & \dots & D_{\frac{t}{2}-2} & A_{\frac{t}{2}-1} & 0 \\ \hline
0 & 0 & 0 & 0 & \dots & 0 & D_{\frac{t}{2}-1} & C
\end{array}\right]$$

Outline

1. Codes with Sequential Local Recovery
   - Introduction
   - An Example: 2D Product Code
   - Upper Bound on Code Rate
   - A Rate-Optimal Binary Code Construction
2. Codes with Availability
   - A Greedy Algorithm for Rate-Bound

Rate-Optimal Binary Code Construction

- A graph-based construction

Rate-Optimal Binary Code Construction

- A graph-based construction
- An iterative procedure for constructing a graph $G_{\frac{t}{2}-1}$ starting from a regular graph G_0

Rate-Optimal Binary Code Construction

- A graph-based construction
- An iterative procedure for constructing a graph $G_{\frac{t}{2}-1}$ starting from a regular graph G_0
- Add nodes to the graph in every step in a layer-by-layer fashion, each time maintaining the girth of graph to be at least $t+1$

Graph Construction

$t = 4, r = 3$ construction

Pick G_0, r-regular, girth $\geq (t+1)$,
$U_0 = V(G_0)$, $|U_0| = u_0$,

G_0 

$3-$ regular graph
with $u_0 = 10$ nodes
and girth $= 5$

Graph Construction

$t = 4, r = 3$ construction

Pick G_0, r-regular, girth $\geq (t+1)$,
$U_0 = V(G_0)$, $|U_0| = u_0$,

$\downarrow$

$i = 1$

G_0

$3 -$ regular graph
with $u_0 = 10$ nodes
and girth $= 5$

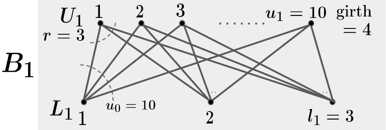
Graph Construction

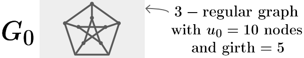
Pick G_0, r-regular, girth $\geq (t+1)$,
$U_0 = V(G_0)$, $|U_0| = u_0$,

$i = 1$

Pick bipartite graph B_i: (r, u_{i-1})-biregular,
$V(B_i) = U_i \cup L_i$, girth $\geq \lceil (t+1)/(i+1/2) \rceil$

$t = 4, r = 3$ construction





Graph Construction

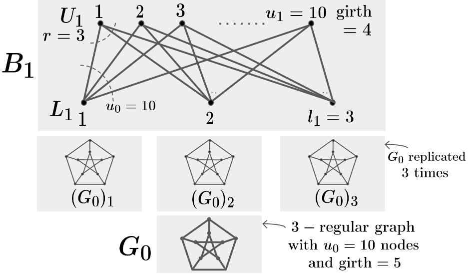
Pick G_0, r-regular, girth $\geq (t+1)$,
$U_0 = V(G_0)$, $|U_0| = u_0$,

$i = 1$

Pick bipartite graph B_i: (r, u_{i-1})-biregular,
$V(B_i) = U_i \cup L_i$, girth $\geq \lceil (t+1)/(i+1/2) \rceil$

Replicate the graph G_{i-1} l_i times (thus each upper node in U_{i-1} is also replicated l_i times)

$t = 4, r = 3$ construction



Graph Construction

Pick G_0, r-regular, girth $\geq (t+1)$,
$U_0 = V(G_0)$, $|U_0| = u_0$,

↓

$i = 1$

↓

Pick bipartite graph B_i: (r, u_{i-1})-biregular,
$V(B_i) = U_i \cup L_i$, girth $\geq \lceil (t+1)/(i+1/2) \rceil$

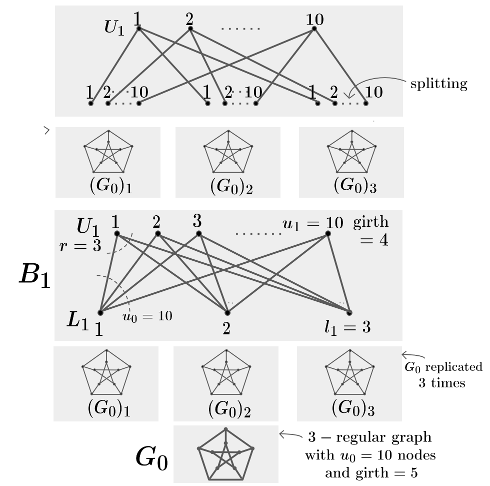
↓

Replicate the graph G_{i-1} l_i times (thus each upper node in U_{i-1} is also replicated l_i times)

↓

Split each of the ℓ_i lower nodes in B_i, each of degree u_{i-1}, into u_{i-1} degree-1 nodes

$t = 4, r = 3$ construction



Graph Construction

Pick G_0, r-regular, girth $\geq (t+1)$, $U_0 = V(G_0)$, $|U_0| = u_0$,

↓

$i = 1$

↓

Pick bipartite graph B_i: (r, u_{i-1})-biregular, $V(B_i) = U_i \cup L_i$, girth $\geq \lceil (t+1)/(i+1/2) \rceil$

↓

Replicate the graph G_{i-1} l_i times (thus each upper node in U_{i-1} is also replicated l_i times)

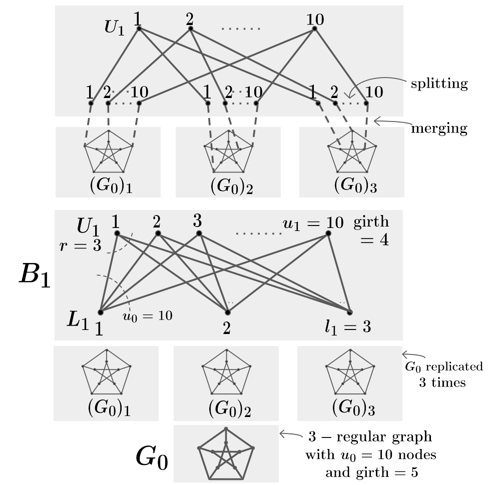
↓

Split each of the ℓ_i lower nodes in B_i, each of degree u_{i-1}, into u_{i-1} degree-1 nodes

↓

Merge the u_{i-1}, degree-1 nodes obtained from splitting a single node of L_i with the u_{i-1} nodes lying in U_{i-1} and corresponding to a single copy of G_{i-1}

$t = 4, r = 3$ construction



Graph Construction

Pick G_0, r-regular, girth $\geq (t+1)$, $U_0 = V(G_0)$, $|U_0| = u_0$,

$i = 1$

Pick bipartite graph B_i: (r, u_{i-1})-biregular, $V(B_i) = U_i \cup L_i$, girth $\geq \lceil (t+1)/(i+1/2) \rceil$

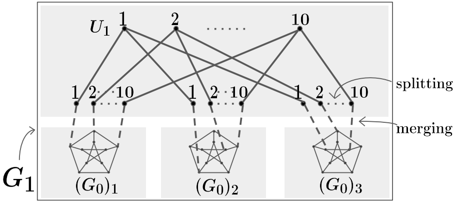
Replicate the graph G_{i-1} l_i times (thus each upper node in U_{i-1} is also replicated l_i times)

Split each of the ℓ_i lower nodes in B_i, each of degree u_{i-1}, into u_{i-1} degree-1 nodes

Merge the u_{i-1}, degree-1 nodes obtained from splitting a single node of L_i with the u_{i-1} nodes lying in U_{i-1} and corresponding to a single copy of G_{i-1}

The resulting graph is G_i; can be verified that G_i has girth $\geq t+1$, the nodes $U_i \subset V(G_i)$ now form the upper layer of the graph G_i and these are the nodes in G_i that participate in the next iterative step

$t = 4, r = 3$ construction



Graph Construction

Pick G_0, r-regular, girth $\geq (t+1)$, $U_0 = V(G_0)$, $|U_0| = u_0$,

$i = 1$

Pick bipartite graph B_i: (r, u_{i-1})-biregular, $V(B_i) = U_i \cup L_i$, girth $\geq \lceil (t+1)/(i+1/2) \rceil$

Replicate the graph G_{i-1} l_i times (thus each upper node in U_{i-1} is also replicated l_i times)

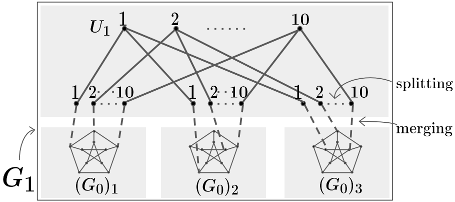
Split each of the ℓ_i lower nodes in B_i, each of degree u_{i-1}, into u_{i-1} degree-1 nodes

Merge the u_{i-1}, degree-1 nodes obtained from splitting a single node of L_i with the u_{i-1} nodes lying in U_{i-1} and corresponding to a single copy of G_{i-1}

The resulting graph is G_i; can be verified that G_i has girth $\geq t+1$, the nodes $U_i \subset V(G_i)$ now form the upper layer of the graph G_i and these are the nodes in G_i that participate in the next iterative step

is $i = \frac{t}{2} - 1$?

$t = 4, r = 3$ construction



Graph Construction

Pick G_0, r-regular, girth $\geq (t+1)$, $U_0 = V(G_0)$, $|U_0| = u_0$,

$i = 1$

Pick bipartite graph B_i: (r, u_{i-1})-biregular, $V(B_i) = U_i \cup L_i$, girth $\geq \lceil (t+1)/(i+1/2) \rceil$

Replicate the graph G_{i-1} l_i times (thus each upper node in U_{i-1} is also replicated l_i times)

Split each of the ℓ_i lower nodes in B_i, each of degree u_{i-1}, into u_{i-1} degree-1 nodes

Merge the u_{i-1}, degree-1 nodes obtained from splitting a single node of L_i with the u_{i-1} nodes lying in U_{i-1} and corresponding to a single copy of G_{i-1}

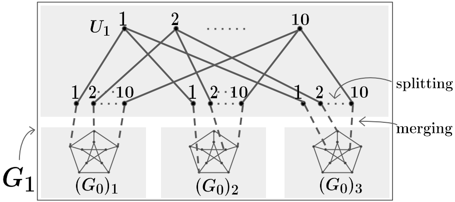
The resulting graph is G_i; can be verified that G_i has girth $\geq t+1$, the nodes $U_i \subset V(G_i)$ now form the upper layer of the graph G_i and these are the nodes in G_i that participate in the next iterative step

is $i = \frac{t}{2} - 1$?

NO: $i = i + 1$

YES: stop

$t = 4, r = 3$ construction



Girth of the Graph

Suppose we are constructing G_i using copies of G_{i-1} and B_i, a bipartite graph with girth at least $g_{B_i} \geq \frac{t+1}{i+\frac{1}{2}}$

Girth of the Graph

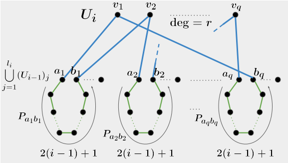
Suppose we are constructing G_i using copies of G_{i-1} and B_i, a bipartite graph with girth at least $g_{B_i} \geq \frac{t+1}{i+\frac{1}{2}}$

Assumption: G_{i-1} has girth at least $t+1$

Girth of the Graph

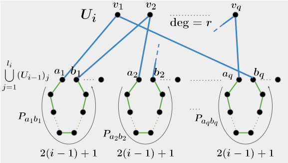
Suppose we are constructing G_i using copies of G_{i-1} and B_i, a bipartite graph with girth at least $g_{B_i} \geq \frac{t+1}{i+\frac{1}{2}}$

Assumption: G_{i-1} has girth at least $t+1$



Girth of the Graph

Suppose we are constructing G_i using copies of G_{i-1} and B_i, a bipartite graph with girth at least $g_{B_i} \geq \frac{t+1}{i+\frac{1}{2}}$
Assumption: G_{i-1} has girth at least $t+1$

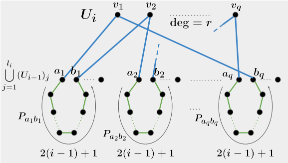


$$\begin{aligned}
\text{Length of any cycle } &\geq 2q + q(2(i-1)+1) \qquad (9)\\
&\geq g_{B_i} + \frac{g_{B_i}}{2}(2(i-1)+1)\\
&\geq g_{B_i}(i+\frac{1}{2}) \geq \frac{t+1}{i+\frac{1}{2}}(i+\frac{1}{2}) = t+1
\end{aligned}$$

Girth of the Graph

Suppose we are constructing G_i using copies of G_{i-1} and B_i, a bipartite graph with girth at least $g_{B_i} \geq \frac{t+1}{i+\frac{1}{2}}$

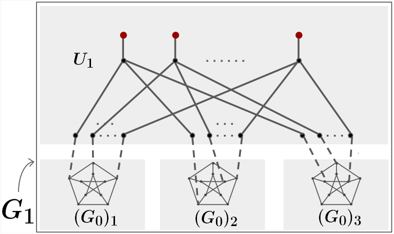
Assumption: G_{i-1} has girth at least $t+1$



$$
\begin{aligned}
\text{Length of any cycle } &\geq 2q + q(2(i-1)+1) \qquad (9)\\
&\geq g_{B_i} + \frac{g_{B_i}}{2}(2(i-1)+1)\\
&\geq g_{B_i}(i+\frac{1}{2}) \geq \frac{t+1}{i+\frac{1}{2}}(i+\frac{1}{2}) = t+1
\end{aligned}
$$

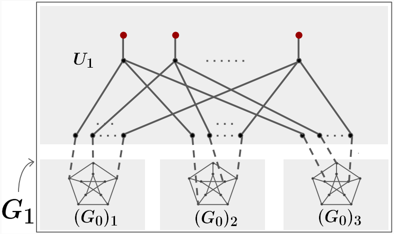
Hence for every i, girth of G_i is at least $t+1$

Code Defined on the Graph: Tanner Graph



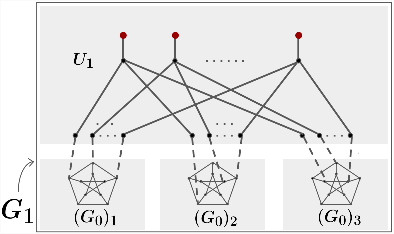
- To every node in top-most layer, attach an edge (with a dummy node)

Code Defined on the Graph: Tanner Graph



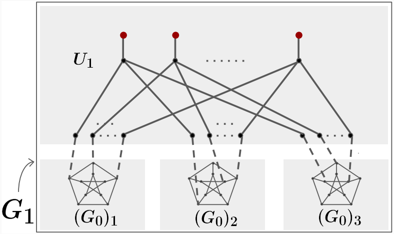
- To every node in top-most layer, attach an edge (with a dummy node)
- Now every edge represents a code symbol

Code Defined on the Graph: Tanner Graph



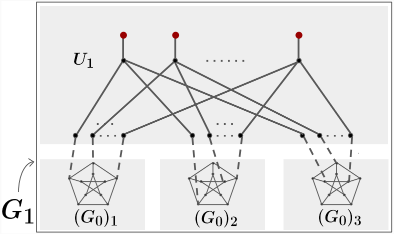
- To every node in top-most layer, attach an edge (with a dummy node)
- Now every edge represents a code symbol
- Every node (except the dummy nodes) represents a parity check of symbols represented by the $r+1$ edges incident on it

Code Defined on the Graph: Tanner Graph



- To every node in top-most layer, attach an edge (with a dummy node)
- Now every edge represents a code symbol
- Every node (except the dummy nodes) represents a parity check of symbols represented by the $r+1$ edges incident on it
- The graph has girth at least $t+1$

Code Defined on the Graph: Tanner Graph



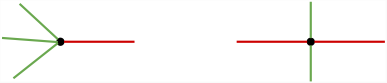
- To every node in top-most layer, attach an edge (with a dummy node)
- Now every edge represents a code symbol
- Every node (except the dummy nodes) represents a parity check of symbols represented by the $r + 1$ edges incident on it
- The graph has girth at least $t + 1$
- Any two dummy nodes are separated by a path of length at least $t + 1$

Erasure Correction

Suppose there are multiple erasures.

Erasure Correction

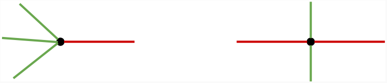
Suppose there are multiple erasures.
If among the edges incident on one node, only one is erased, then it can be recovered. But, if two edges incident on a node are erased, then cannot recover using that parity check.



Hence, for correcting multiple erasures, at every step there should exist at least one parity check, with exactly one erased symbol.

Erasure Correction

Suppose there are multiple erasures.
If among the edges incident on one node, only one is erased, then it can be recovered. But, if two edges incident on a node are erased, then cannot recover using that parity check.



Hence, for correcting multiple erasures, at every step there should exist at least one parity check, with exactly one erased symbol.
When do we run into trouble?

t Erasure Correctability

Suppose there are e erasures.

t Erasure Correctability

Suppose there are e erasures.

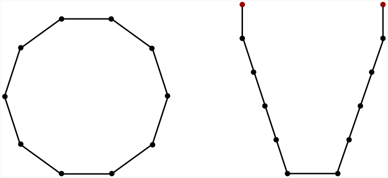
In the graph retain only those edges representing the erased symbols, and the nodes they are connected to.

t Erasure Correctability

Suppose there are e erasures.

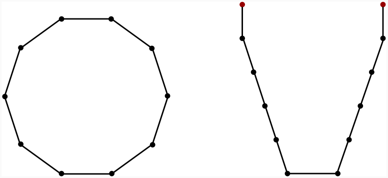
In the graph retain only those edges representing the erased symbols, and the nodes they are connected to.

Two types of erasure-patterns can cause the decoder to stop:



t Erasure Correctability

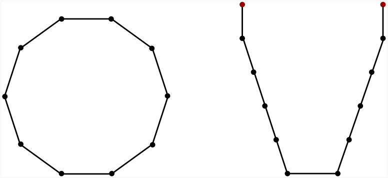
Suppose there are e erasures.
In the graph retain only those edges representing the erased symbols, and the nodes they are connected to.
Two types of erasure-patterns can cause the decoder to stop:



- The graph does not have cycles of length less than $t+1$.
  Therefore the first case does not arise

t Erasure Correctability

Suppose there are e erasures.
In the graph retain only those edges representing the erased symbols, and the nodes they are connected to.
Two types of erasure-patterns can cause the decoder to stop:



- The graph does not have cycles of length less than $t+1$. Therefore the first case does not arise
- Any two dummy nodes are separated by a path of length at least $t+1$. Hence the second case is also avoided

Rate-Optimality of the construction

Upon counting the number of code symbols and number of parity checks, one can see that this construction yields rate-optimal codes for any even t and any $r \geq 3$.

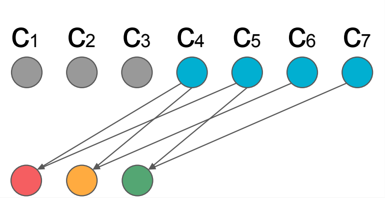
Codes with Availability

Outline

Availability

Same setting as earlier

Desirable:
$c_1 = f_1(c_4, c_5)$, $c_2 = f_2(c_4, c_6)$,
$c_3 = f_3(c_5, c_7)$
So that, can recover the lost symbols in any order.



An additional property called "majority-logic decodability".

Questions:
What is the highest "rate" achievable by such codes? Highest "minimum-distance"? How to design such rate-optimal codes with low blocklength, low field-size?

Definition: Strict Availability via Orthogonal Parity Checks

$$\underbrace{H}_{(m\times n)} = \underbrace{\begin{bmatrix} 1 & 1 & & 1 & & & \\ & 1 & 1 & & 1 & & \\ & & 1 & 1 & & 1 & \\ & & & 1 & 1 & & 1 \\ 1 & & & & 1 & 1 & \\ & 1 & & & & 1 & 1 \\ 1 & & 1 & & & & 1 \end{bmatrix}}_{\Updownarrow \ w_H(\text{each column})=t \ \Updownarrow} \Leftrightarrow w_H(\text{each row}) = (r+1)$$

- Let $S_j^{(i)}$ be the support of the jth row having a 1 in column i
- Then

$$S_j^{(i)} \cap S_l^{(i)} = \{i\}, \quad j = 1, 2, \cdots, t, \quad j \neq l.$$

- t orthogonal parity checks per code symbol
- Terminology: $(n, k, r, t)_{\text{sa}}$ codes.

Outline

Rate-Bound for Strict Availability for t $= 3$

Theorem

Rate-Bound[1] *Let $\mathcal{C}$ be an $(n, k, r, 3)_{sa}$ code over the field $\mathbb{F}_q$ having connected Tanner graph, then its rate is upper bounded by the following expression:*

$$\frac{k}{n} \leq 1 - \frac{3(1 + L_1 + L_2)}{(r+1)(3 + L_1 + 2L_2)}, \tag{10}$$

$$\text{where: } m = \frac{3n}{r+1}, \quad L_1 = \left\lceil \frac{(2r-1)m}{3(r+2)} - \frac{1}{r+2} - 1 \right\rceil,$$
$$L_2 = \left\lfloor \frac{m - 3 - L_1}{2} \right\rfloor,$$

[1] S. B. Balaji and P. Vijay Kumar, “Bounds on Codes with Locality and Availability”,2017. [Online] .Available: https://arxiv.org/abs/1611.00159

Greedy Algorithm : Step 1

$$H = \begin{bmatrix} 1 & 1 & & 1 & & & \\ & 1 & 1 & & 1 & & \\ & & 1 & 1 & & 1 & \\ & & & 1 & 1 & & 1 \\ 1 & & & & 1 & 1 & \\ & 1 & & & & 1 & 1 \\ 1 & & 1 & & & & 1 \end{bmatrix}$$

$S = \{1\}$,

$$P = \begin{bmatrix} 1 & 1 & & 1 & & & \\ & & & & & & \\ & & & & & & \\ & & & & & & \\ 1 & & & & 1 & 1 & \\ & & & & & & \\ 1 & & 1 & & & & 1 \end{bmatrix}$$

Greedy Algorithm: Step 2

$$H = \left[\begin{array}{c|c|c|c|c|c|c}
1 & 1 & & 1 & & & \\ \hline
 & 1 & 1 & & 1 & & \\ \hline
 & & 1 & 1 & & 1 & \\ \hline
 & & & 1 & 1 & & 1 \\ \hline
1 & & & & 1 & 1 & \\ \hline
 & 1 & & & & 1 & 1 \\ \hline
1 & & 1 & & & & 1
\end{array}\right]$$

$S = \{1, 2\}$,

$$P = \left[\begin{array}{c|c|c|c|c|c|c}
1 & 1 & & 1 & & & \\ \hline
 & 1 & 1 & & 1 & & \\ \hline
 & & & & & & \\ \hline
 & & & & & & \\ \hline
1 & & & & 1 & 1 & \\ \hline
 & 1 & & & & 1 & 1 \\ \hline
1 & & 1 & & & & 1
\end{array}\right]$$

Greedy Algorithm: Step 3

$$H = \begin{bmatrix} 1 & 1 & & 1 & & & \\ & 1 & 1 & & 1 & & \\ & & 1 & 1 & & 1 & \\ & & & 1 & 1 & & 1 \\ 1 & & & & 1 & 1 & \\ & 1 & & & & 1 & 1 \\ 1 & & 1 & & & & 1 \end{bmatrix}$$

$S = \{1, 2, 5\}$,

$$P = \begin{bmatrix} 1 & 1 & & 1 & & & \\ & 1 & 1 & & 1 & & \\ & & & & & & \\ & & & 1 & 1 & & 1 \\ 1 & & & & 1 & 1 & \\ & 1 & & & & 1 & 1 \\ 1 & & 1 & & & & 1 \end{bmatrix}$$

Greedy Algorithm: Step 4

$$H = \left[\begin{array}{c|c|c|c|c|c|c} 1 & 1 & & 1 & & & \\ \hline & 1 & 1 & & 1 & & \\ \hline & & 1 & 1 & & 1 & \\ \hline & & & 1 & 1 & & 1 \\ \hline 1 & & & & 1 & 1 & \\ \hline & 1 & & & & 1 & 1 \\ \hline 1 & & 1 & & & & 1 \end{array}\right]$$

$S = \{1, 2, 5, 6\}$, Hence $P = H$ and $k \leq n - |S| = 3$

$$P = \left[\begin{array}{c|c|c|c|c|c|c} 1 & 1 & & 1 & & & \\ \hline & 1 & 1 & & 1 & & \\ \hline & & 1 & 1 & & 1 & \\ \hline & & & 1 & 1 & & 1 \\ \hline 1 & & & & 1 & 1 & \\ \hline & 1 & & & & 1 & 1 \\ \hline 1 & & 1 & & & & 1 \end{array}\right]$$

Greedy Algorithm

1. Let $S = \emptyset, P = \emptyset$.
2. Step 1: Pick an arbitrary number σ_1 from $[n]$ and set $S = \{\sigma_1\}$ and $P = \{\underline{c} \in \text{Rows}(H) : \sigma_1 \in \text{Support}(\underline{c})\}$.
3. Step i, $i \geq 2$: Choose a number $\sigma_i \in [n] - S$ such that
   $\sigma_i = \text{argmax}_{\{j \in [n] - S\}} |D_j| \times I(|D_j| \leq 2)$
   where $D_j = \{\underline{c} \in P : j \in \text{Support}(\underline{c})\}$.
   Now $S = S \cup \{\sigma_i\}$ and
   $P = P \cup \{\underline{c} \in \text{Rows}(H) - P : \sigma_i \in \text{Support}(\underline{c})\}$.
4. **Pseudocode for the Greedy Algorithm:** $P = \emptyset, S = \emptyset$, $i = 1$.
   while $|P| < m$
   do Step i
   $i = i + 1$
   end while.
5. It is clear that, $k \leq n - |S|$ at the end of the algorithm.

Backup Slides

Rawat et al. Regular Graph Construction

Any $(r+1)$-regular (bipartite) graph, with girth at least $t+1$ gives the Tanner graph of an $(n, k, r, t)_{seq}$ code.

Rawat et al. Regular Graph Construction

Any $(r+1)$-regular (bipartite) graph, with girth at least $t+1$ gives the Tanner graph of an $(n, k, r, t)_{seq}$ code.
Can show that rate is $\frac{r-1}{r+1} + \frac{1}{n}$.

Rawat et al. Regular Graph Construction

Any $(r+1)$-regular (bipartite) graph, with girth at least $t+1$ gives the Tanner graph of an $(n, k, r, t)_{seq}$ code.
Can show that rate is $\frac{r-1}{r+1} + \frac{1}{n}$.
Can show that it meets our bound only when the graph is a Moore graph, which are very rare. Hence sub-optimal for most parameters.

Analysis of Greedy Algorithm

Let $g_i \in \{1, 2\}$ be the number of new codewords added to P at step i. Let s_1^i, s_2^i, s_3^i be the number of weight $1, 2, 3$ columns in the matrix formed by codewords in P respectively.

$$\text{if } g_{i+1} = 2, g_{i+2} = 2 \text{ then}$$
$$s_1^{i+1} = s_1^i + 2r - 1, \;\; s_2^{i+1} = s_2^i + 0, \;\; s_3^{i+1} = s_3^i + 1.$$

$$\text{if } g_{i+1} = 1, g_{i+2} = 2 \text{ then}$$
$$s_1^{i+1} = s_1^i - \phi_i + r + 1, \;\; s_2^{i+1} = s_2^i - \phi_i, \;\; s_3^{i+1} = s_3^i + \phi_i,$$
$$\text{for some } 1 \leq \phi_i \leq r + 1.$$

$$\text{if } g_{i+1} = 2, g_{i+2} = 1 \text{ then}$$
$$s_1^{i+1} = s_1^i + 2r - 1 - 2l_i, \;\; s_2^{i+1} = s_2^i + l_i, \;\; s_3^{i+1} = s_3^i + 1,$$
$$\text{for some } 0 < l_i \leq 2r.$$

Analysis of Greedy Algorithm

Let $S_{uj} = \{i : g_{i+1} = u, g_{i+2} = j, |S| - 1 \geq i \geq 2\}$ and $l_{uj} = |S_{uj}|$ at the end of the algorithm. Now using the global constraints ($s_1 = s_1^{|S|} = 0$, $s_2 = s_2^{|S|} = 0$, $s_3 = s_3^{|S|} = n$ at the end of the algorithm.):

$$
\begin{aligned}
s_2^{|S|} &= \gamma_1 - \sum_{i \in S_{12}} \phi_i + \sum_{i \in S_{21}} I_i - \sum_{i \in S_{11}} J_i + \sum_{i \in S_{11}} \psi_i - (r+1) = 0, \\
m &= \frac{3n}{r+1} = 5 + g_3 + 2(l_{22} + l_{12}) + l_{21} + l_{11}.
\end{aligned}
$$

Analysis of Greedy Algorithm

$$l_{11} + l_{21} \quad \geq \quad \frac{(2r-1)m}{3(r+2)} - \frac{1}{r+2} - 1.$$

1. For $j = 1, 2$, let $L_j = |\{i : g_i = j, |S| \geq i \geq 1\}|$ at the end of the algorithm.
2. $|S| = L_1 + L_2 + 1$ and $m = L_1 + 2L_2 + 3$. Hence $L_1 \geq l_{11} + l_{21}$.

References

S. B. Balaji, G. R. Kini, and P. V. Kumar, “A tight rate bound and a matching construction for locally recoverable codes with sequential recovery from any number of multiple erasures, 2017. [Online].
Available: http://arxiv.org/abs/1611.08561

S. B. Balaji and P. Vijay Kumar, “Bounds on Codes with Locality and Availability”,2017. [Online] .
Available: https://arxiv.org/abs/1611.00159

Thank you!

Questions?