Graph-Based Error Correcting Codes for Flash Memories

Ahmed Hareedy
Duke University

Flash Memory Summit
August 5th 2019

Presentation Outline

- **Motivation and mission statement**
- **Introduction to graph-based codes**
- **High performance spatially-coupled codes**
- **Construction of multi-dimensional codes**
- **Conclusion and ongoing research**

Presentation Outline

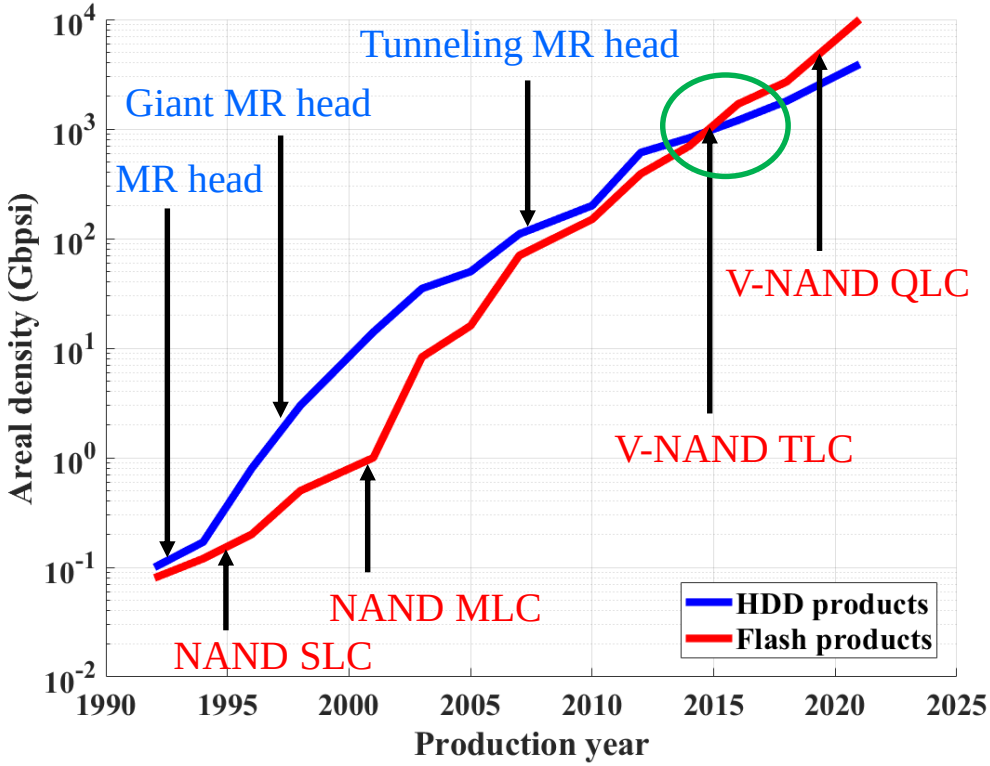
- **Motivation and mission statement**
- **Introduction to graph-based codes**
- **High performance spatially-coupled codes**
- **Construction of multi-dimensional codes**
- **Conclusion and ongoing research**

Data Storage Density Trends

- **Data storage is a story where density increases as innovations in physics lead to innovations in signal processing.**
  - Our focus here is on coding theoretic innovations.

- **The density of data storage devices evolved as follows:**
  - Areal density is in gigabits per inch^2.

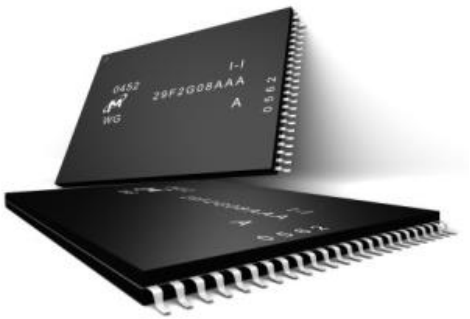
- **With the vertical NAND (3D NAND), Flash devices are already winning!**

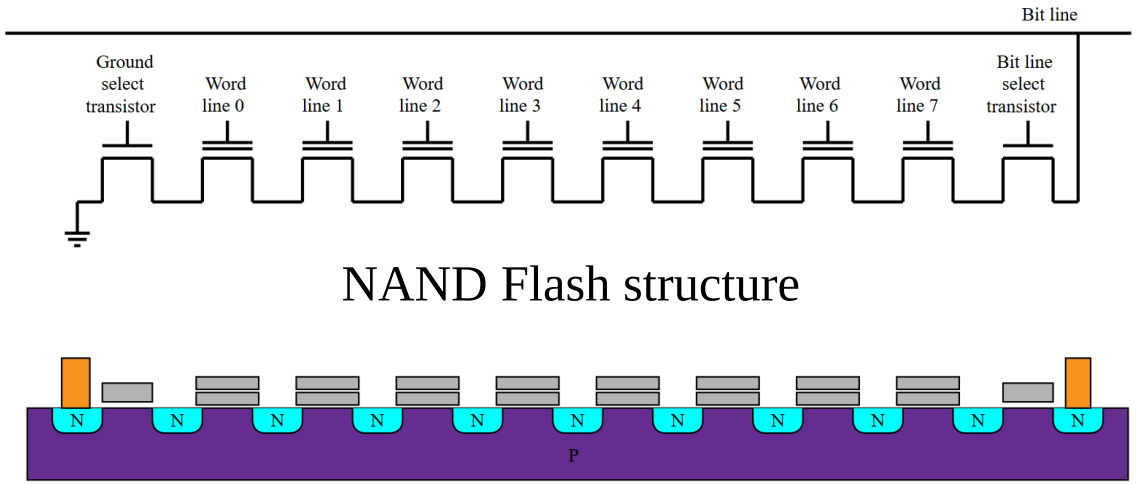


Motivation of the Work

- **We are living in the age of big data.**
  - The storage capacity of modern data centers is in the order of exabytes (10^{18} bytes) at least.
  - SSDs and HDDs are now approaching 10 terabits per inch^2!

- **These high densities result in adding/exacerbating sources of errors in modern storage devices.**
  - Flash: Inter-cell interference (ICI) and wear-out.



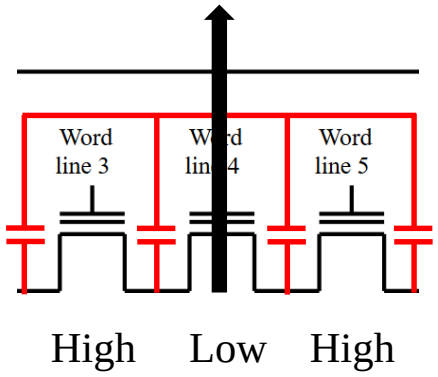


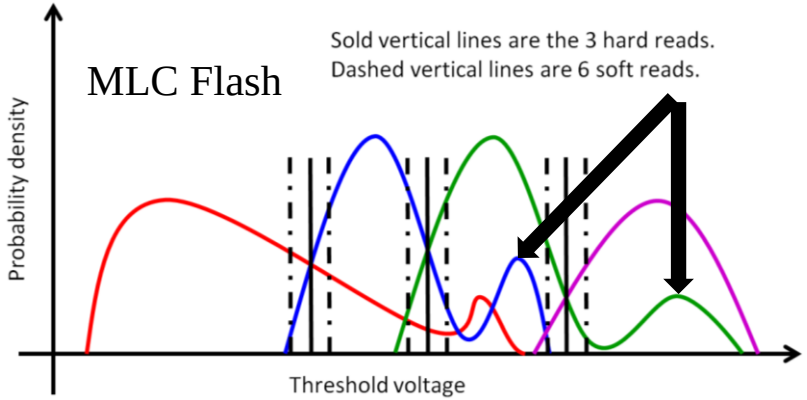
NAND Flash structure

Sources of Errors in Flash Devices

- **In Flash, data are stored in the form of charges in floating gate transistors that control their thresholds.**
- **The amount of charge can change because of many effects:**
  - ICI due to writing specific data patterns, e.g., 101 in SLC.
    - Parasitic capacitances result in unintentional increase in charge levels.
  - Device wear-out results in programming errors.

Charge increase on middle cell (more harmful within the same word)

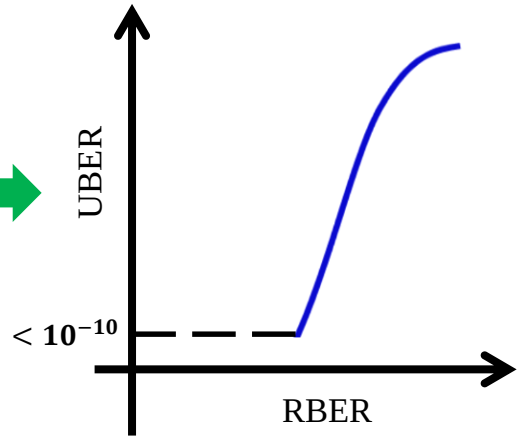




- **Advanced coding techniques are used to mitigate these effects.**
  - Constrained codes [Qin 14] [Hareedy AL-19].
  - Error-correcting codes (ECCs), which are our focus today.

Mission Statement of Coding Theorists

- **Modern storage devices (both Flash and magnetic recording devices) operate at very low error rates.**
  - Effective ECC techniques are **a must** in order to enable storage engineers to use such dense devices with confidence.
  - Graph-based codes offer great performance!



- **Our mission is to provide effective ECC techniques exploiting the characteristics of the channels underlying storage devices.**

Modern Graph-Based Codes

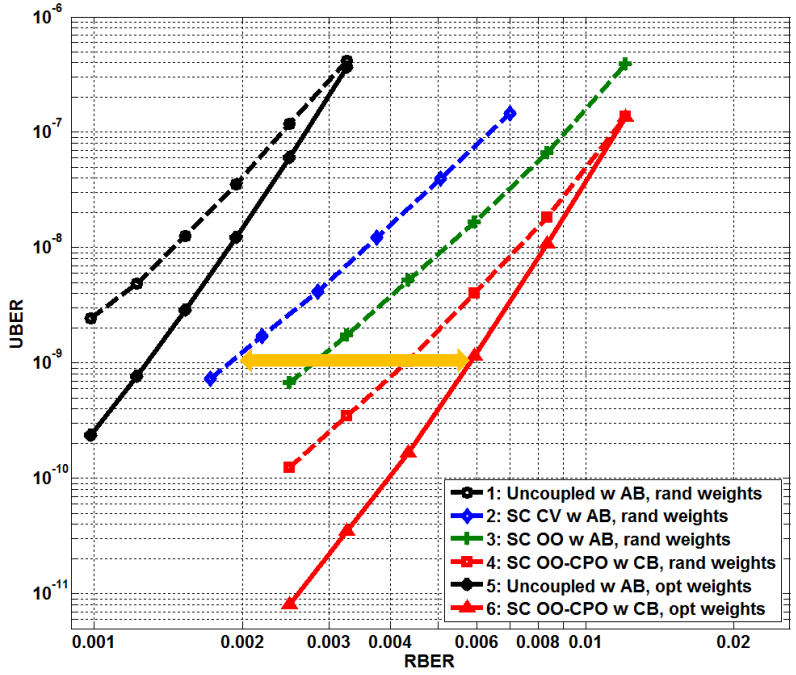
- **Spatially-coupled (SC) LDPC codes have capacity approaching performance. They also have complexity/latency advantages.**
  - We discuss a combinatorial approach to construct binary and non-binary SC codes for Flash memories.

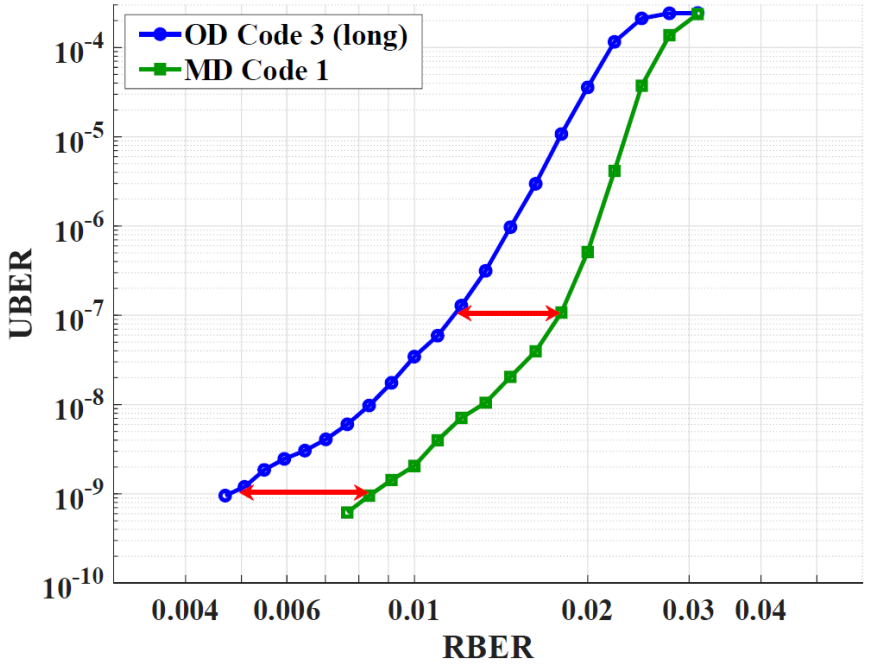
- **We then illustrate a technique to design high performance multi-dimensional (MD) graph-based codes.**
  - The technique couples multiple copies of optimized 1D (OD) graph-based codes in an informed manner.

- **Why do we need these modern codes?**
  - Optimized SC codes outperform block codes of similar parameters.
  - MD codes further increase the gains compared with longer OD codes, and they are suitable for MD devices.

Fast Forward to the Results

- **Appropriately-designed graph-based codes offer great performance gains in Flash systems.**
  - Spatially-coupled codes.
    - Gains compared with state-of-the-art SC codes.
  - Multi-dimensional codes.
    - Gains compared with one-dimensional (OD) codes.





Presentation Outline

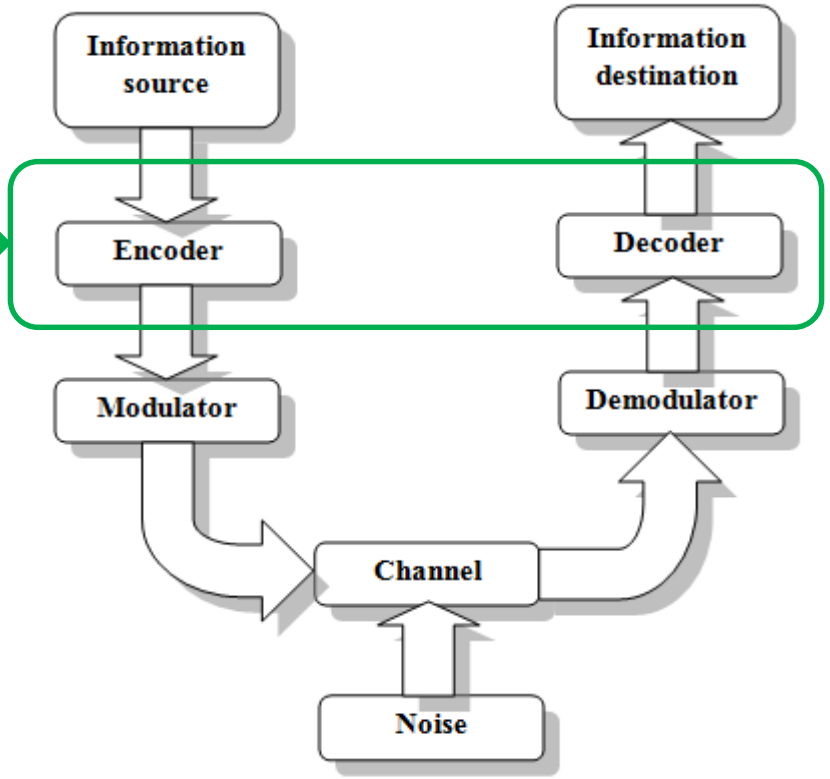
- Motivation and mission statement
- Introduction to graph-based codes
- High performance spatially-coupled codes
- Construction of multi-dimensional codes
- Conclusion and ongoing research

Importance of Coding

- **Coding is a tool to improve system performance:**
  - Through preventing errors.
  - Through correcting errors.

In data storage:

- Constrained codes are used to prevent errors.
- Error-correcting codes are used to correct what was not prevented.



Parity-Check Codes (Binary Example)

- **Parity-check codes are a class of block codes.**
  - The code is defined by a parity-check matrix **H**.
  - A codeword **c** must satisfy $\mathbf{Hc}^{\mathrm{T}} = \mathbf{0}$.

Systematic form

$$\mathbf{H}_{(n-k)\times n} = \left[\mathbf{P}^{\mathrm{T}}_{(n-k)\times k}\ \mathbf{I}_{(n-k)\times(n-k)}\right], \quad \mathbf{G}_{k\times n} = \left[\mathbf{I}_{k\times k}\ \mathbf{P}_{k\times(n-k)}\right].$$

  - Consider the following simple **H**:

$$c_1 = v_1 \oplus v_2 \oplus v_3 \oplus v_5,$$
$$c_2 = v_1 \oplus v_2 \oplus v_4 \oplus v_6,$$
$$c_3 = v_1 \oplus v_3 \oplus v_4 \oplus v_7.$$

	v_1	v_2	v_3	v_4	v_5	v_6	v_7
c_1	1	1	1	0	1	0	0
c_2	1	1	0	1	0	1	0
c_3	1	0	1	1	0	0	1

$v_1\ v_2\ v_3\ v_4\ v_5\ v_6\ v_7$

  - Suppose the vector [0 0 1 1 0 0 0] is received for hard decision.
    - $c_1 = c_2 = 1$ (unsatisfied), while $c_3 = 0$ (satisfied).
    - c_1 and c_2 send change (C) while c_3 sends stay (S) to all involved bits.
    - Flipping v_2 only from 0 to 1 makes all check equations satisfied.

Low-Density Parity-Check (LDPC) Codes

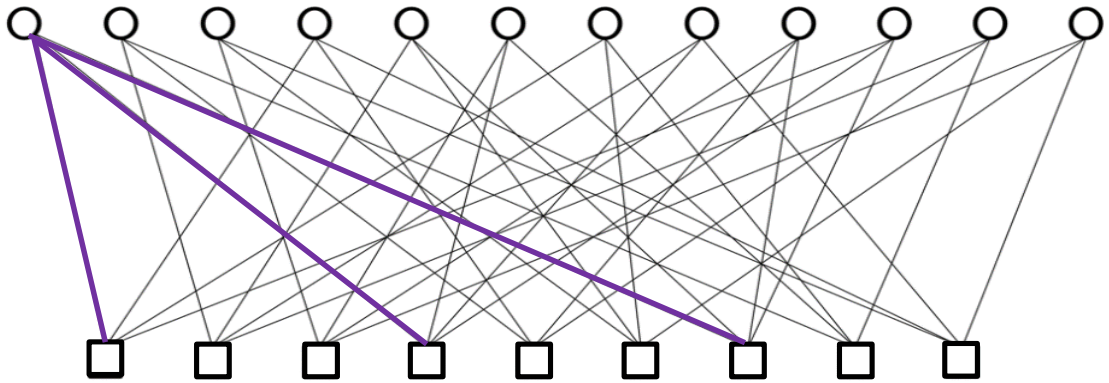
- **Sparse parity-check matrix:**

$$\mathbf{H} =$$

1	0	0	1	0	0	1	0	0	1	0	0
0	1	0	0	1	0	0	1	0	0	1	0
0	0	1	0	0	1	0	0	1	0	0	1
1	0	0	0	0	1	0	1	0	1	0	0
0	1	0	1	0	0	0	0	1	0	1	0
0	0	1	0	1	0	1	0	0	0	0	1
1	0	0	0	1	0	0	0	1	1	0	0
0	1	0	0	0	1	1	0	0	0	1	0
0	0	1	1	0	0	0	1	0	0	0	1

Columns represent variable nodes (VNs).
VNs are also called bit nodes.
Rows represent check nodes (CNs).

- **Corresponding Tanner graph (a bipartite graph):**



Circles represent VNs.
Squares represent CNs.

LDPC Codes: Some Concepts and Notation

- **Decoding is performed iteratively through messages from VNs to CNs and from CNs to VNs.**
- **Detrimental configurations are error-prone structures in the Tanner graph of the code.**

- **Useful notation:**
  - γ is the column weight (VN degree).
  - κ is the row weight (CN degree).
  - $\mathbf{H}^{\mathrm{p}}$ is the protograph matrix of a block code.
  - σ is the $z \times z$ circulant matrix; $\sigma^0 = \mathbf{I}$.

Our codes have fixed column weights.

- **Lifting:**

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

$\mathbf{H}^{\mathrm{p}}$ with $\gamma = 3$ and $\kappa = 5$

Lifting with $z = 5$

σ^3	σ^2	σ^0	σ^4	σ^1
σ^1	σ^0	σ^2	σ^3	σ^4
σ^1	σ^2	σ^4	σ^0	σ^3

$\mathbf{H}$

Simple Illustration of LDPC Decoding

- **We start from a single VN and a single CN [Gallager TH].**
  - Consider the VN v_x and the set of its CN indices $\mathcal{S}_x$ with $y \in \mathcal{S}_x$.
  - Let the set of indices of the VNs involved in c_y be $\mathcal{T}_y$.
  - Let $\mathbf{v}_y$ (with $v_{y(j)}$) be the vector of VNs having indices in $\mathcal{T}_y \setminus \{x\}$.

  - To get the probability that c_y is satisfied given v_x is 1, we marginalize over all vectors of VNs:

$$\Pr_{x,1}(c_y = 0) = \sum_{\text{all } \mathbf{v}_y \mid v_x=1 \text{ and } c_y=0} \Pr(\mathbf{v}_y)$$

This is the CN to VN message!

$$= \sum_{\text{all } \mathbf{v}_y \mid \text{VNs at } \mathcal{T}_y\setminus\{x\} \text{ have odd no. of 1's}} \prod_j \Pr(v_{y(j)}). \quad (1)$$

Similarly, given v_x is 0:

$$\Pr_{x,0}(c_y = 0) = \sum_{\text{all } \mathbf{v}_y \mid \text{VNs at } \mathcal{T}_y\setminus\{x\} \text{ have even no. of 1's}} \prod_j \Pr(v_{y(j)}). \quad (2)$$

Sum-product decoding

Continue: Belief Propagation (BP)

- **Mathematical simplification for the binary case.**
  - If the sum in (1) and (2) is replaced by max, we get the max-product or the min-sum decoding.
  - Using a famous lemma, (1) can be simplified as follows:

$$\Pr_{x,1}(c_y = 0) = \frac{1}{2}\left(1 - \prod_j \left[1 - 2\Pr(v_{y(j)} = 1)\right]\right).$$

Similarly, (2) can be simplified as follows:

$$\Pr_{x,0}(c_y = 0) = \frac{1}{2}\left(1 + \prod_j \left[1 - 2\Pr(v_{y(j)} = 1)\right]\right).$$

  - Let $\mathbf{c}_x$ (with $c_{x(i)}$) be the vector of CNs having indices in $\mathcal{S}_x \setminus \{y\}$.
  - The probability that v_x is 1 right after the channel is P_x. The probability that v_x is 0 right after the channel is $1 - P_x$.

Graphical Interpretation of BP

- **Using the remaining CNs, we get:**
  - The final BP decoding formula

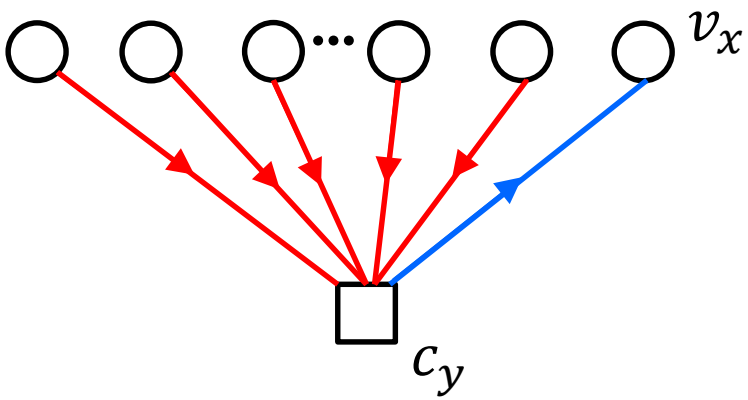
This is the VN to CN message!

$$\mathrm{LR}_{y,x} = \frac{\Pr_y(v_x = 1)}{\Pr_y(v_x = 0)} = \frac{P_x}{1 - P_x} \cdot \prod_i \frac{\Pr_{x,1}(c_{x(i)} = 0)}{\Pr_{x,0}(c_{x(i)} = 0)}. \qquad (3)$$

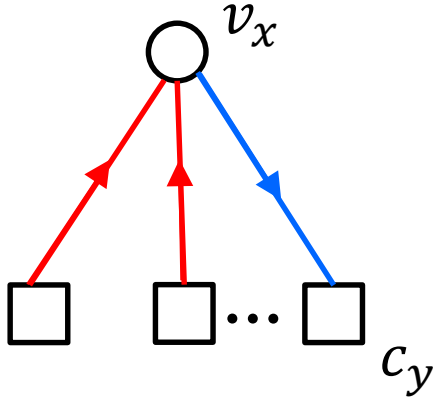
Likelihood ratio | Channel part | Coding part

  - Log domain calculations are made if $\log_e(.)$ is taken.

- **Graphical interpretation of the decoding procedure:**



Check to variable



Variable to check

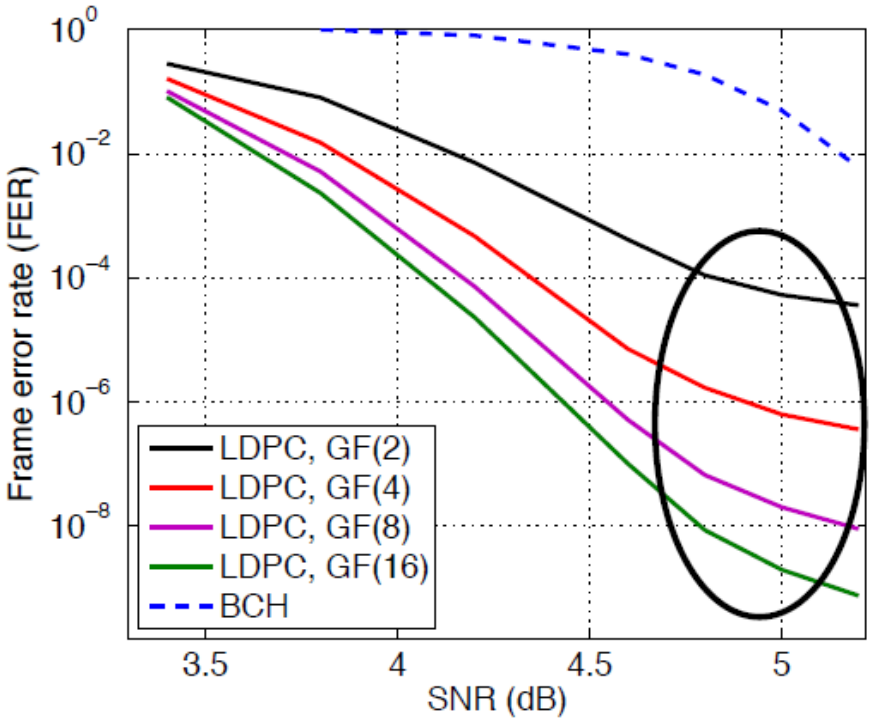
Non-Binary vs Binary LDPC Codes

- **Why non-binary?**
  - Grouping bits into symbols over Galois Field of size q (GF(q)) decreases the probability of decoding failure.
  - Increasing the Galois Field size q results in better performance.
  - Disadvantage: Decoding complexity increases.

Block length $\approx$ 1000 bits

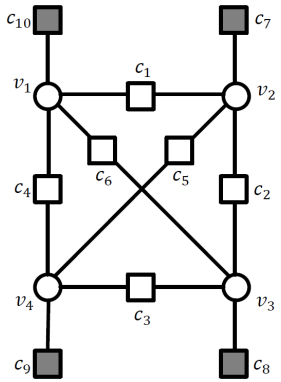
Rate $\approx$ 0.9

Column weight $\gamma = 4$

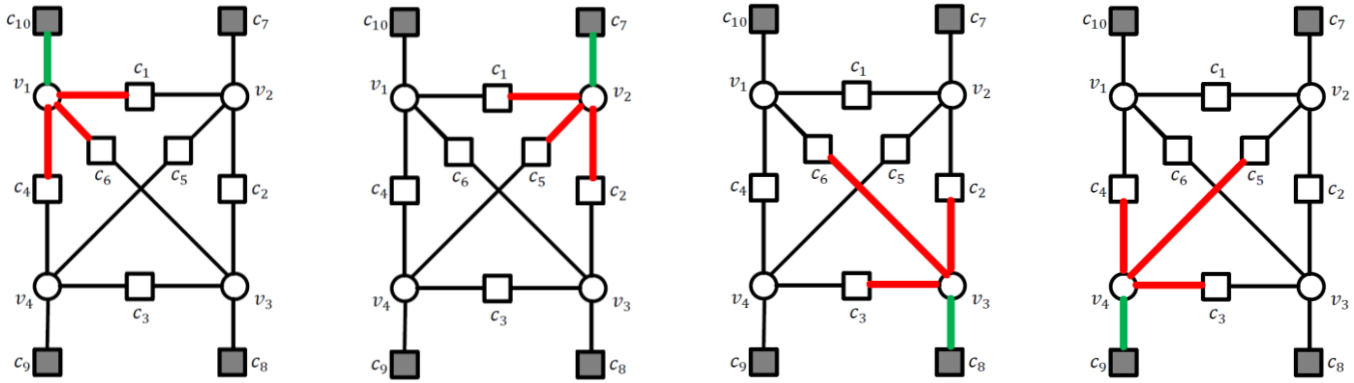


Error Floor of LDPC Codes: Binary Absorbing Sets

- **Absorbing sets [Dolecek 10] are the reason behind error floor.**
  - For an (a, b) absorbing set: a is the size of the set, b is the number of unsatisfied CNs connected to the set, and each VN is connected to more satisfied than unsatisfied neighboring CNs.
- **Binary absorbing sets are described in terms of topological conditions only.**
  - Here is a $(4, 4)$ binary absorbing set ($\gamma = 4$), which is elementary.



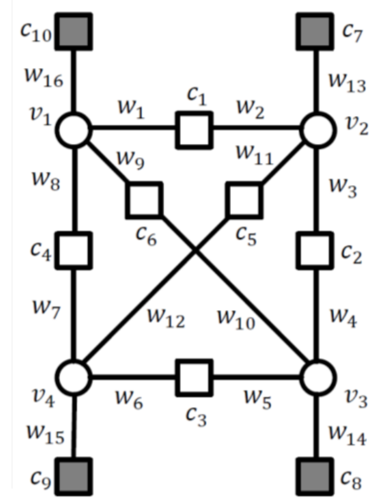
Circles represent VNs.
White squares represent satisfied CNs.
Grey squares represent unsatisfied CNs.



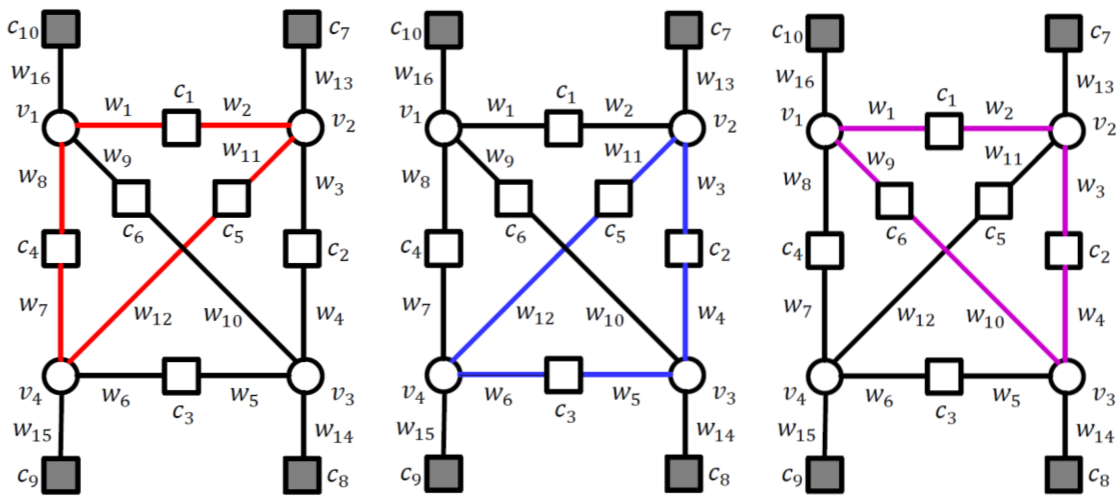
Non-Binary (NB) Absorbing Sets

- **Differences between non-binary and binary absorbing sets (ASs):**
  - For NB, the values of VNs matter.
  - Topological conditions alone are not enough; weight conditions have to be added, e.g., [Amiri 14]:

$$\prod_{i=1}^{x} w_{\theta(2i-1)} = \prod_{i=1}^{x} w_{\theta(2i)} \text{ over } \mathrm{GF}(q).$$



Each VN value is not zero, and each edge weight is not zero.



Example:
$(4, 4)$ NB absorbing set

Elementary weight conditions:

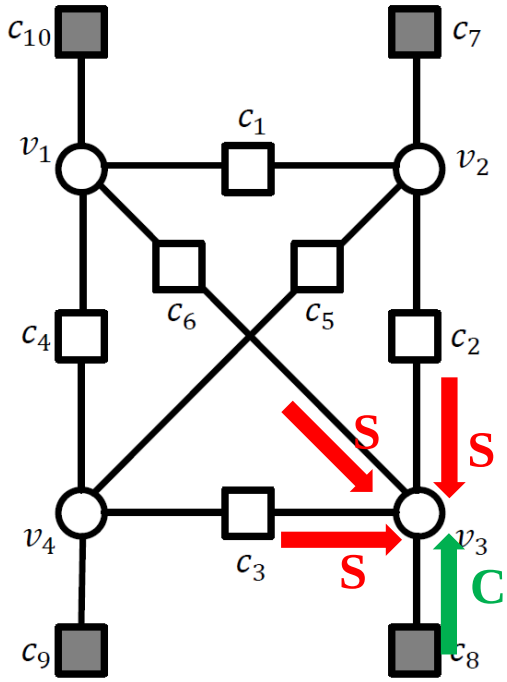
$w_1 w_{11} w_7 = w_2 w_{12} w_8$

$w_3 w_5 w_{12} = w_4 w_6 w_{11}$

$w_1 w_3 w_{10} = w_2 w_4 w_9$

Absorbing Sets Are Always Problematic!

- **How can the absorbing set cause a decoding error?**
  - Assume that an all zeros codeword is transmitted.
  - Suppose that errors occur only over all the four VNs in the shown $(4, 4)$ absorbing set.
  - Assume all the VNs are now 1's.
  - Consider hard decision decoding.
  - Each degree-2 CN now is satisfied $(1 + 1 = 0)$, while degree-1 CNs are not.
  - Each VN receives 3 stay and only 1 change messages from the connected CNs.
  - Despite being in error, all VNs stick to their wrong values.
  - Consequently, the decoder is **absorbed**!



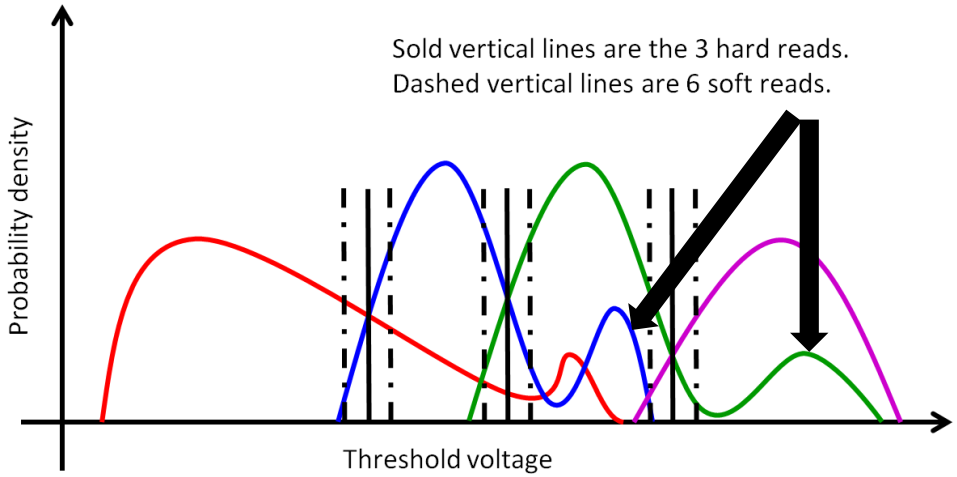
Required: More Gain for Asymmetric Channels

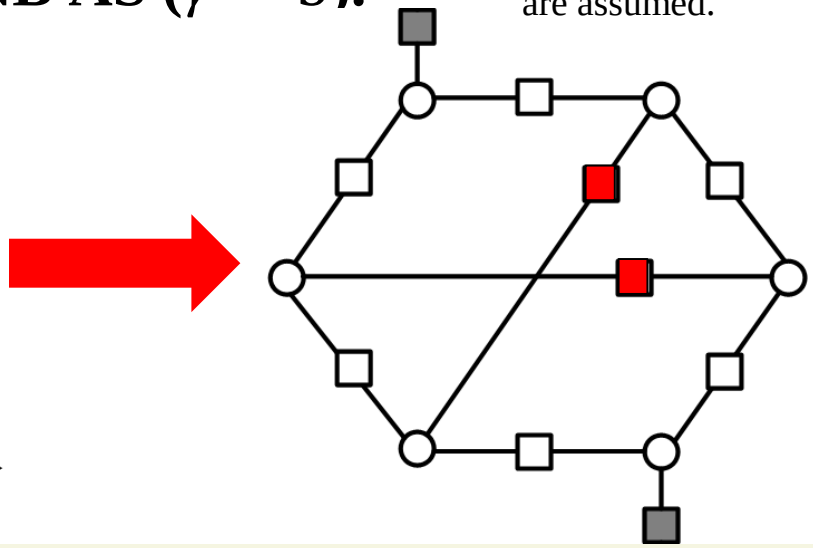
- **For symmetric channels (like AWGN) [Amiri 14]:**
  - The detrimental objects are always elementary ASs.
  - Each satisfied CN is of degree 2, and each unsatisfied CN is of degree 1.
- **Using optimization against elementary objects, only a modest gain for Flash channels (inherently asymmetric) can be achieved.**
- **Absorbing sets also affect the waterfall performance.**
  - Low-weight codewords are absorbing sets with $b = 0$.
- **Better performance gains are required!**
- **Will the detrimental objects stay the same for asymmetric channels?**

The Answer Is NO!

- **Asymmetry in the channel (e.g., in Flash) can result in:**
  - ASs with some unsatisfied CNs having degree > 1.
  - ASs with some satisfied CNs having degree > 2.
- **This is mainly because of the high VN error magnitudes.**
  - Programming errors have high magnitudes.
- **Such dominant objects are non-elementary!**

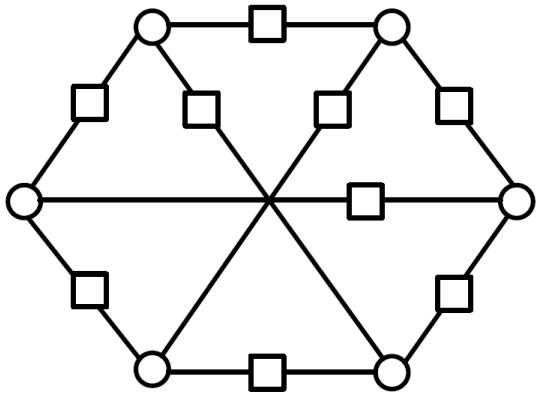
- **Example:** $(6, 4)$ **non-elementary NB AS** ($\gamma = 3$).





Detrimental Objects of Interest

- **These objects are general absorbing sets of type two (GASTs) in the case of Flash channels [Hareedy 16].**
- **Recall an (a, b, d_1, d_2, d_3) GAST over GF(q), $q \in \{2, 4, 8, \dots\}$:**
  - a is the number of VNs in the set (its size).
  - b is the number of unsatisfied CNs (degree 1 or 2).
  - d_1 (resp., d_2 and d_3) is the number of degree-1 (resp., 2 and > 2) CNs.
  - Each VN is connected to strictly more satisfied than unsatisfied CNs (for some VN values in GF(q)\ {0}).



$(6, 0, 0, 9, 0)$ GAST

- **Define also an (a, d_1) unlabeled elementary trapping (resp., absorbing) set (UTS) (resp., (UAS)).**
  - Unlabeled means all edge weights are set to 1's.

Presentation Outline

- **Motivation and mission statement**
- **Introduction to graph-based codes**
- **High performance spatially-coupled codes**
- **Construction of multi-dimensional codes**
- **Conclusion and ongoing research**

We Focus Now on Spatially-Coupled (SC) Codes

- **SC codes have capacity approaching performance [Iyengar 12].**
  - They also offer additional degrees of freedom in the code design.

- **We discuss a combinatorial approach to design high performance binary and non-binary SC codes for Flash channels.**
  - Stage 1: Optimize the partitioning parameters (OO).
  - Stage 2: Optimize the circulant powers (CPO).
  - OO: Optimal overlap. CPO: Circulant power optimizer.
  - The two stages operate on the unlabeled graph (binary matrix).

  - Edge weights (NB) are optimized via the weight consistency matrix (WCM) framework [Hareedy TI-19].
  - **Our approach exploits the characteristics of Flash channels.**

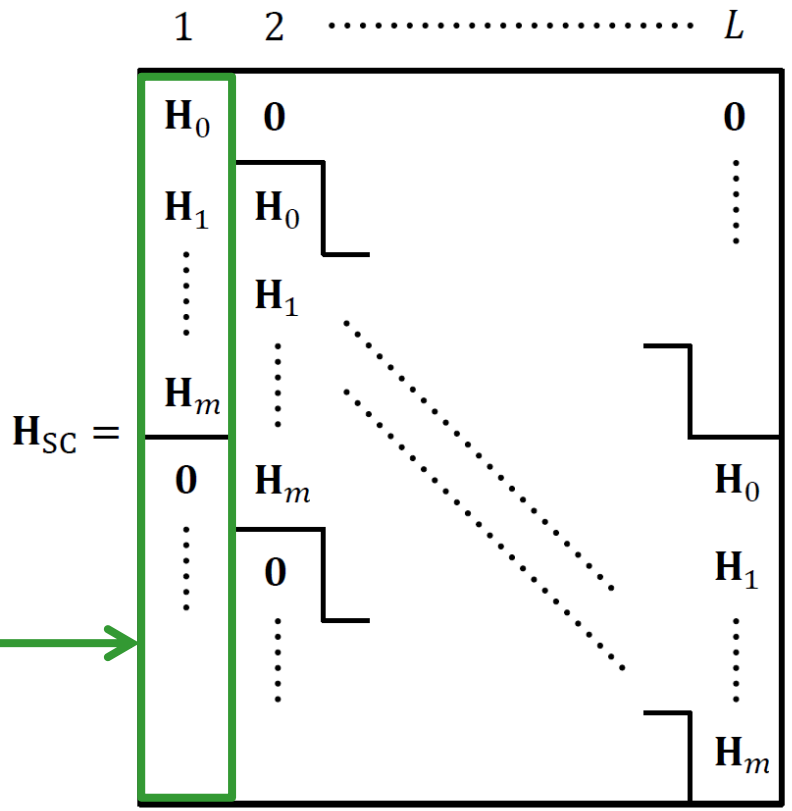
Useful Mathematical Notation

- **The following notation is useful:**
  - $\mathbf{H}$ is the binary parity check matrix of the underlying block code (we use circulant-based (CB) codes with no zero circulants).
  - $\mathbf{H}_{\text{SC}}$ is the binary parity check matrix of the SC code.
  - γ is the column weight (VN degree) of $\mathbf{H}$ and $\mathbf{H}_{\text{SC}}$.
  - κ is the row weight (CN degree) of $\mathbf{H}$.
  - Each circulant in $\mathbf{H}$ is of the form $\sigma^{f_{i,j}}$, where $0 \leq i \leq \gamma - 1$ and $0 \leq j \leq \kappa - 1$. $f_{i,j}$ are the circulant powers.
  - σ is the $z \times z$ identity matrix cyclically shifted 1 unit to the left.
  - $\mathbf{M}^{\text{p}}$ is the protograph matrix of a matrix $\mathbf{M}$ (set $z = 1$).
  - m is the memory of the SC code.
  - L is the coupling length of the SC code.
  - q is the Galois Field (GF) size in the case of NB codes.

Construction of SC Codes

- **The construction steps are:**
  - Partition $\mathbf{H}$ into $m + 1$ components: $\mathbf{H}_0$, $\mathbf{H}_1$, …, $\mathbf{H}_m$.
  - Component matrices are coupled L times to construct $\mathbf{H}_{\mathrm{SC}}$. $\mathbf{H}_{\mathrm{SC}}$ is of size $\gamma z(L + m) \times \kappa z L$.
  - If the SC code is NB, non-zero values in GF(q) are assigned to the 1's in $\mathbf{H}$.
  - Overlap parameters (for the partitioning) and circulant powers can be adjusted to enhance the properties of $\mathbf{H}_{\mathrm{SC}}$.

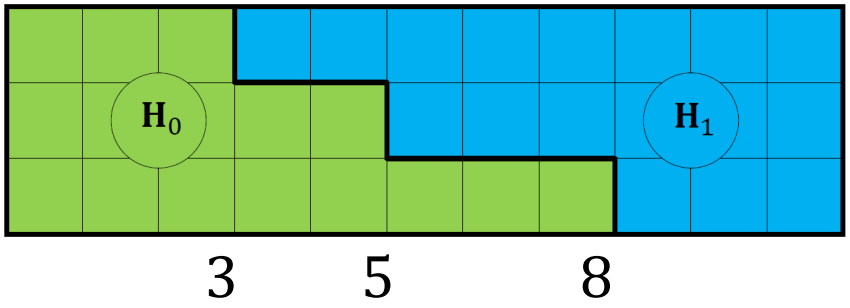
$$\mathbf{H} = \sum_{y=0}^{m} \mathbf{H}_y. \quad (4)$$



Partitioning Techniques from the Literature

- **Cutting vector (CV) partitioning [Mitchell 14]:**
  - Uses a vector of ascending integers to contiguously partition the underlying block matrix.

H =



  - In this example: $\gamma = 3$, $\kappa = 11$, $m = 1$, $L = 7$, and the CV is $[3, 5, 8]$.

$\mathbf{H}_{SC}$ =

H_0	0	0	0	0	0	0
H_1	H_0	0	0	0	0	0
0	H_1	H_0	0	0	0	0
0	0	H_1	H_0	0	0	0
0	0	0	H_1	H_0	0	0
0	0	0	0	H_1	H_0	0
0	0	0	0	0	H_1	H_0
0	0	0	0	0	0	H_1

- **Minimum overlap (MO) partitioning [Esfahanizadeh 17]:**
  - Minimizes the overlap of each pair of rows of circulants in each component matrix (non-contiguous partitioning).
- **The OO-CPO approach (also non-contiguous) outperforms both!**

What is the Goal of the Design?

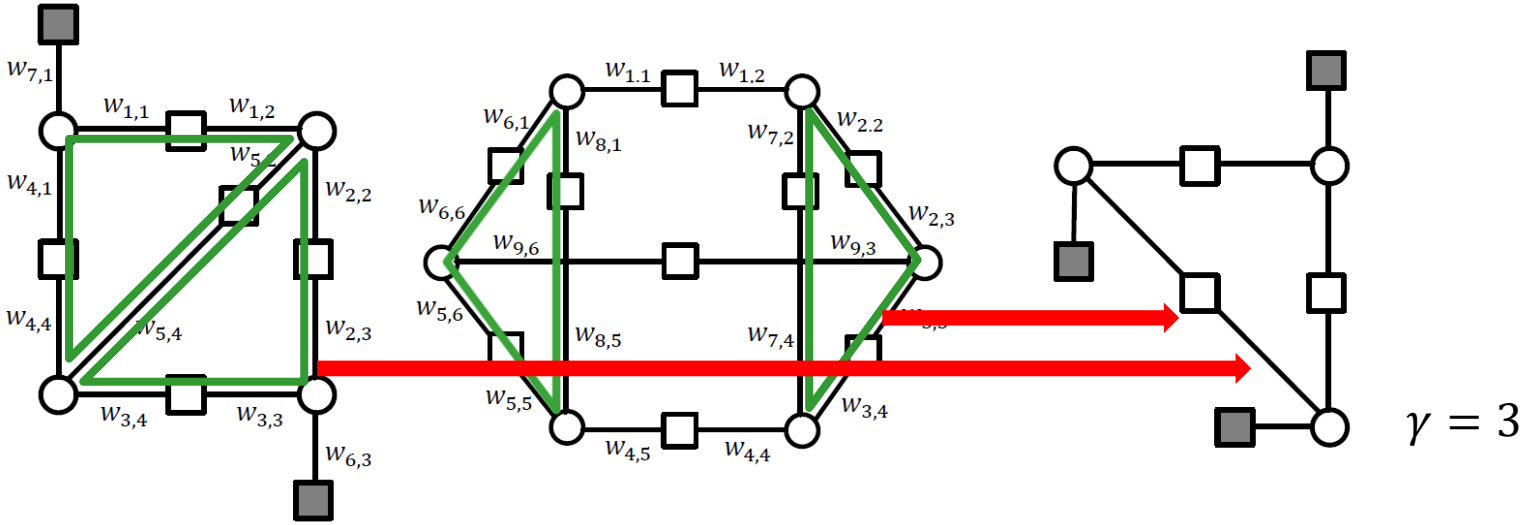
- **The goal is to remove as many detrimental GASTs as possible.**
  - The framework is general for any γ and any m.
  - From [Hareedy 16], we know the nature of detrimental GASTs for graph-based codes over Flash channels.

- **First, optimize the unlabeled graph (binary matrix).**
  - Derive the optimal partitioning (OO) corresponding to the minimum number of detrimental objects in the protograph.
  - Employ a circulant power optimizer (CPO) to further reduce the number of detrimental GASTs in the unlabeled graph.
  - Stop after this stage if the SC code is binary.

- **Then, optimize the edge weights for NB-SC codes.**
  - Apply the previously mentioned WCM framework.

The Common Denominator Substructure

- **We simplify the problem of optimizing the unlabeled graph of the SC code (the OO-CPO approach)!**
  - Search for a common denominator substructure that exists as a subgraph in multiple dominant GASTs.
  - In codes with column weight γ simulated over Flash channels, this substructure is the $(3, 3(\gamma - 2))$ UAS/UTS.
  - Minimize the number of such UASs/UTSs in the graph of $\mathbf{H}_{SC}$.

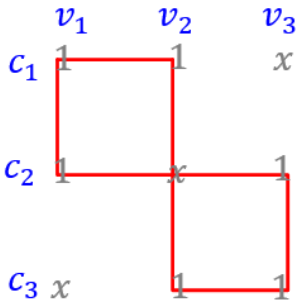


Apprropriate edge weights are assumed, with $q \geq 2$.

$(4, 2, 2, 5, 0)$ GAST $(6, 0, 0, 9, 0)$ GAST $(3, 3)$ UAS

From Protograph to Unlabeled Graph

- **The $(3, 3(\gamma - 2))$ UAS/UTS is a cycle of length 6.**
  - There is only one protograph configuration that can generate it in the unlabeled graph.
  - This configuration is also a cycle of length 6.



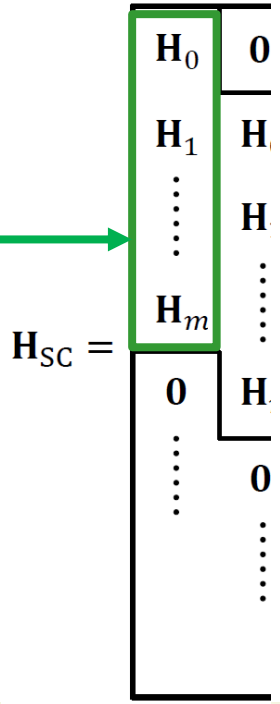
- **Note that the $(4, 4(\gamma - 2))$ UAS/UTS is a cycle of length 8 with no internal connections.**
  - The $(4, 4(\gamma - 2))$ UAS/UTS is important for magnetic recording.
  - There are multiple protograph configurations, which we call patterns, that can generate it in the unlabeled graph.

- **The OO-CPO approach is applicable to a variety of UASs/UTSs.**

OO: The First Step Towards Our Goal

- **We aim at establishing a discrete optimization problem:**
  - The total number of cycles of length 6 in the protograph, F, is expressed in terms of the overlap (partitioning) parameters.

- **The problem needs further simplification!**
- **Exploiting the repetitive nature of SC codes:**
  - F is computed only over submatrices of $\mathbf{H}_{\mathrm{SC}}^{\mathrm{p}}$.
  - Overlap parameters are only defined over $\mathbf{\Pi}_1^{1,\mathrm{p}}$, which is the protograph matrix of $\mathbf{\Pi}_1^1$.
- **The next step is to find the expression of F.**

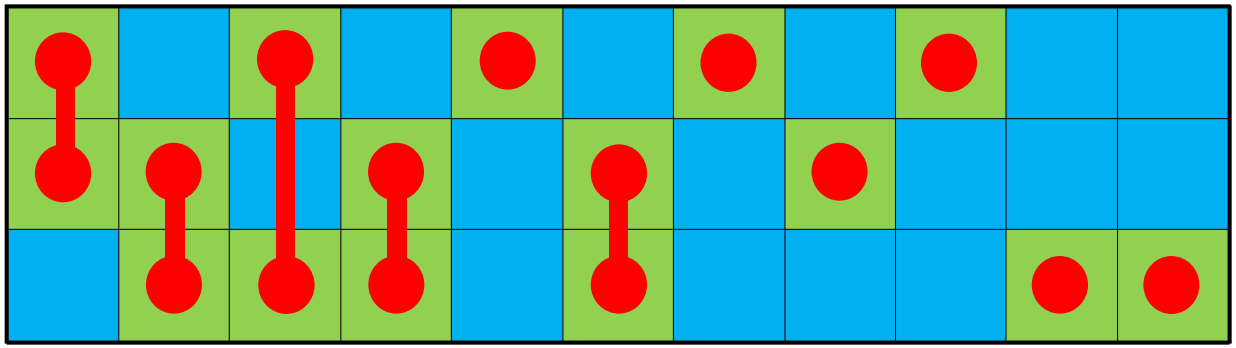


What Are the Overlap Parameters?

- **The set of independent non-zero overlap parameters is $\mathcal{O}_{\text{ind}}$.**
- **Example: For $\gamma = 3$ and $m = 1$, we have:**

  $\mathcal{O}_{\text{ind}} = \{t_{\{0\}}, t_{\{1\}}, t_{\{2\}}, t_{\{0,1\}}, t_{\{0,2\}}, t_{\{1,2\}}, t_{\{0,1,2\}}\}$ (at most degree γ).
  - Other overlap parameters are functions of the ones in $\mathcal{O}_{\text{ind}}$.
- **We illustrate their definitions via an example:**
  - Consider the case of $\kappa = 11$:
  - $t_{\{0\}} = 5.$
  - $t_{\{1\}} = 5.$
  - $t_{\{2\}} = 6.$
  - $t_{\{0,1\}} = 1.$
  - $t_{\{0,2\}} = 1.$
  - $t_{\{1,2\}} = 3.$
  - $t_{\{0,1,2\}} = 0.$

$\mathbf{H}^{\text{p}} =$



$\mathbf{H}_0^{\text{p}}$

Define a degree-μ overlap and a degree-μ overlap parameter.

$\mathbf{H}_1^{\text{p}}$

Useful Definitions and Facts

- **Definitions:**
  - $\mathcal{O}$ is the set of non-zero overlap parameters. Moreover,

$$\mathcal{O}_{\text{ind}} = \{t_{\{i_1,\ldots,i_\mu\}} \mid 1 \leq \mu \leq \gamma,\ 0 \leq i_1,\ldots,i_\mu \leq m\gamma - 1,$$
$$\forall\{i_{\tau_1}, i_{\tau_2}\} \subseteq \{i_1,\ldots,i_\mu\}\ i_{\tau_1} \not\equiv i_{\tau_2} \pmod{\gamma}\}.$$

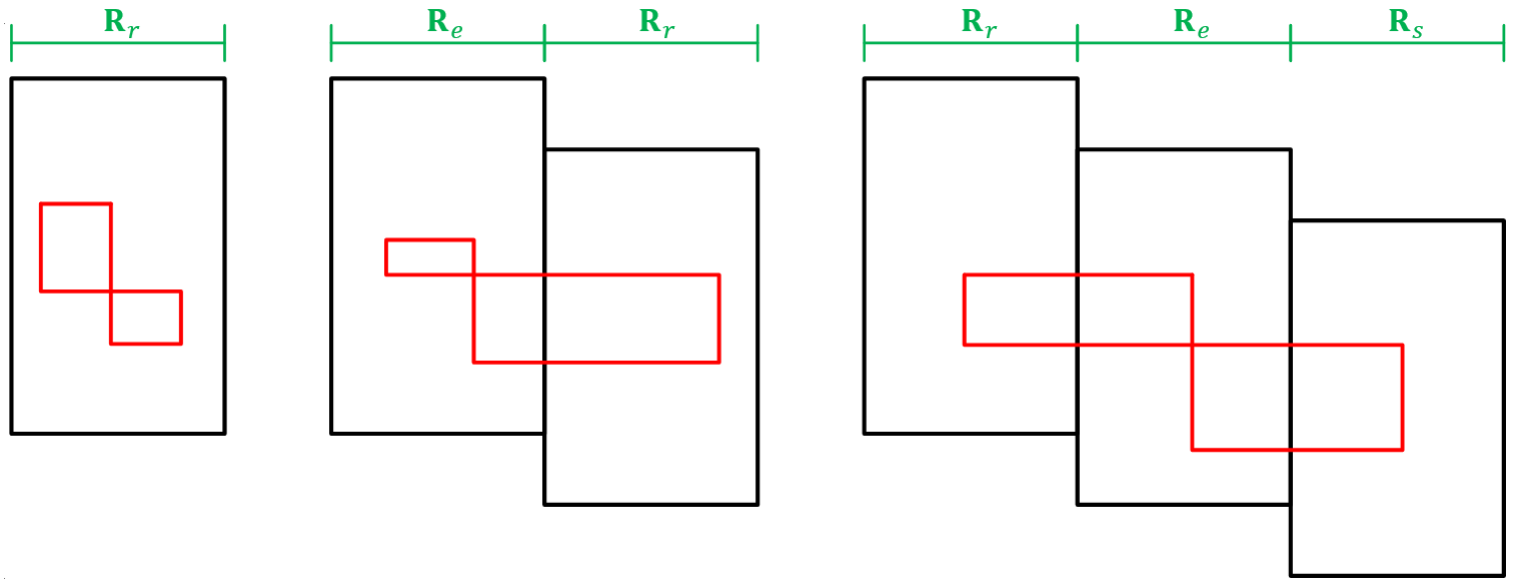
  - $\mathbf{R}_r$ is the reference replica.
  - The CNs of the cycle are of the form $c_x = (r-1)\gamma + i_x$.
  - $(y)^+ = \max\{y, 0\}$.
  - F_1^k is the number of cycle instances that start at replica $\mathbf{R_1}$ and span k consecutive replicas. Start and span are w.r.t. VNs.
  - Each VN of a cycle corresponds to an overlap.

- **A cycle of length 6 spans at most $\chi = m + 1$ consecutive replicas.**

$$F = \sum_{k=1}^{\chi} (L - k + 1) F_1^k$$

Building the Discrete Optimization Problem

- **The cycle has three cases depending on how its VNs are distributed across replicas.**
  - Those three cases are shown below.

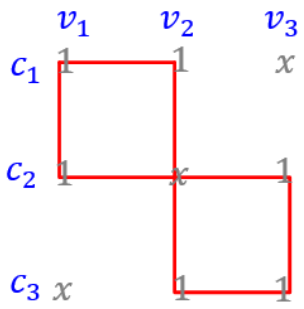


- **We provide more details about the first and third cases.**

Analysis of Protograph Cycle (1 of 2)

- **<u>Lemma 1:</u> The cycle has three cases.**
  - Case 1: The number of instances with all overlaps in $\mathbf{R}_r$ is:

$$\mathcal{A}\left(t_{\{i_1,i_2\}}, t_{\{i_1,i_3\}}, t_{\{i_2,i_3\}}, t_{\{i_1,i_2,i_3\}}\right)$$
$$= t_{\{i_1,i_2,i_3\}}\left(t_{\{i_1,i_2,i_3\}}-1\right)^{+}\left(t_{\{i_2,i_3\}}-2\right)^{+}$$
$$+ t_{\{i_1,i_2,i_3\}}\left(t_{\{i_1,i_3\}}-t_{\{i_1,i_2,i_3\}}\right)\left(t_{\{i_2,i_3\}}-1\right)^{+}$$
$$+ \left(t_{\{i_1,i_2\}}-t_{\{i_1,i_2,i_3\}}\right)t_{\{i_1,i_2,i_3\}}\left(t_{\{i_2,i_3\}}-1\right)^{+}$$
$$+ \left(t_{\{i_1,i_2\}}-t_{\{i_1,i_2,i_3\}}\right)\left(t_{\{i_1,i_3\}}-t_{\{i_1,i_2,i_3\}}\right)t_{\{i_2,i_3\}}.$$



Overlaps are $c_1 - c_2$, $c_1 - c_3$, and $c_2 - c_3$.

  - Case 3: The number of instances with overlaps in three replicas, $\mathbf{R}_r$, $\mathbf{R}_e$, and $\mathbf{R}_s$, $r < e < s$, is:

$$\mathcal{C}\left(t_{\{i_1,i_2\}}, t_{\{i_1+(r-e)\gamma,i_3+(r-e)\gamma\}}, t_{\{i_2+(r-s)\gamma,i_3+(r-s)\gamma\}}\right)$$
$$= t_{\{i_1,i_2\}}t_{\{i_1+(r-e)\gamma,i_3+(r-e)\gamma\}}t_{\{i_2+(r-s)\gamma,i_3+(r-s)\gamma\}}.$$

Analysis of Protograph Cycle (2 of 2)

- **Theorem 1:** **The total number of instances in the binary protograph of an SC Code with $\gamma \geq 3$, κ, m, $L \geq m+1$, and $\mathcal{O}$, is:**

$$\chi = m+1 \qquad F = \sum_{k=1}^{\chi} (L-k+1) F_1^k, \tag{5}$$

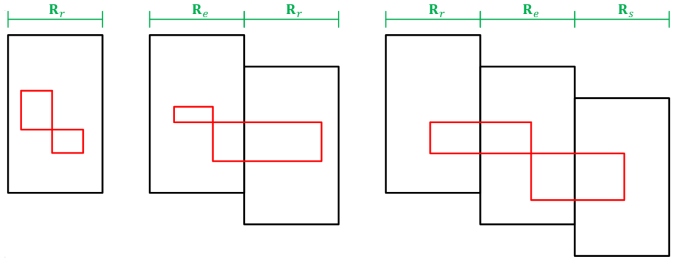
where F_1^k, $k \in \{1, 2, \ldots, m+1\}$ are given by:

$$F_1^1 = \sum_{\{i_1,i_2,i_3\} \subset \{0,\ldots,(m+1)\gamma-1\}} \mathcal{A}\left(t_{\{i_1,i_2\}}, t_{\{i_1,i_3\}}, t_{\{i_2,i_3\}}, t_{\{i_1,i_2,i_3\}}\right),$$

$$F_1^2 = \sum_{i_1 \in \{0,\ldots,(m+1)\gamma-1\}, \{i_2,i_3\} \subset \{\gamma,\ldots,(m+1)\gamma-1\}} \mathcal{B}\left(t_{\{i_1,i_2\}}, t_{\{i_1,i_3\}}, t_{\{i_1,i_2,i_3\}}, t_{\{i_2-\gamma,i_3-\gamma\}}\right)$$
$$+ \sum_{i_1 \in \{0,\ldots,(m+1)\gamma-1\}, \{i_2,i_3\} \subset \{0,\ldots,m\gamma-1\}} \mathcal{B}\left(t_{\{i_1,i_2\}}, t_{\{i_1,i_3\}}, t_{\{i_1,i_2,i_3\}}, t_{\{i_2+\gamma,i_3+\gamma\}}\right),$$

$$F_1^{k\geq 3} = \sum_{i_1 \in \{0,\ldots,(m+1)\gamma-1\}, \{i_2,i_3\} \subset \{(k-1)\gamma,\ldots,(m+1)\gamma-1\}} \mathcal{B}\left(t_{\{i_1,i_2\}}, t_{\{i_1,i_3\}}, t_{\{i_1,i_2,i_3\}}, t_{\{i_2+(1-k)\gamma,i_3+(1-k)\gamma\}}\right)$$
$$+ \sum_{i_1 \in \{0,\ldots,(m+1)\gamma-1\}, \{i_2,i_3\} \subset \{0,\ldots,(m-k+2)\gamma-1\}} \mathcal{B}\left(t_{\{i_1,i_2\}}, t_{\{i_1,i_3\}}, t_{\{i_1,i_2,i_3\}}, t_{\{i_2+(k-1)\gamma,i_3+(k-1)\gamma\}}\right)$$
$$+ \sum_{h=2}^{k-1} \sum_{i_1 \in \{(h-1)\gamma,\ldots,(m+1)\gamma-1\}, i_2 \in \{(k-1)\gamma,\ldots,(m+1)\gamma-1\}, i_3 \in \{(k-1)\gamma,\ldots,(m+h)\gamma\}} \mathcal{C}\left(t_{\{i_1,i_2\}}, t_{\{i_1+(1-h)\gamma,i_3+(1-h)\gamma\}}, t_{\{i_2+(1-k)\gamma,i_3+(1-k)\gamma\}}\right). \tag{6}$$

with $\overline{i_1} \neq \overline{i_2}$, $\overline{i_1} \neq \overline{i_3}$, $\overline{i_2} \neq \overline{i_3}$ and $\overline{i_x} = (i_x \bmod \gamma)$.



Then, We Compute the OO Parameters

- **The discrete optimization problem is described as follows.**
  - Mathematical formulation:

$$F^* = \min_{\mathcal{O}_{\text{ind}}} F. \tag{7}$$

  - Optimization constraints:

    Linear constraints on the parameters in $\mathcal{O}_{\text{ind}}$ capturing interval constraints and the balanced partitioning constraint.

  - As with the set $\mathcal{O}_{\text{ind}}$, the optimization constraints depend only on code parameters, and not on the common substructure of interest (which depends on the channel).
  - A solution to (7) is $\mathbf{t}^*$. We call $\mathbf{t}^*$ an **optimal vector**. All optimal vectors perform the same way.

Now, We Focus on Circulant Powers

- **After $\mathbf{H}_{\mathrm{SC}}^{\mathrm{p}}$ is designed via $\mathbf{t}^{*}$, the CPO is applied to further reduce the number of $(3, 3(\gamma-2))$ UASs/UTSs in the graph of $\mathbf{H}_{\mathrm{SC}}$.**
  - The $(3, 3(\gamma-2))$ UAS/UTS is a **cycle of length** 6.
  - We only need to operate on $\chi = m + 1$ replicas.
  - $\mathbf{\Pi}_{1}^{\chi,\mathrm{p}}$ is the non-zero part of the first χ replicas in $\mathbf{H}_{\mathrm{SC}}^{\mathrm{p}}$.
  - A cycle-6 defined as $c_1 - v_1 - c_2 - v_2 - c_3 - v_3$ in the graph of $\mathbf{\Pi}_{1}^{\chi,\mathrm{p}}$ results in z cycles of length 6 after lifting iff [Fossorier 04]:

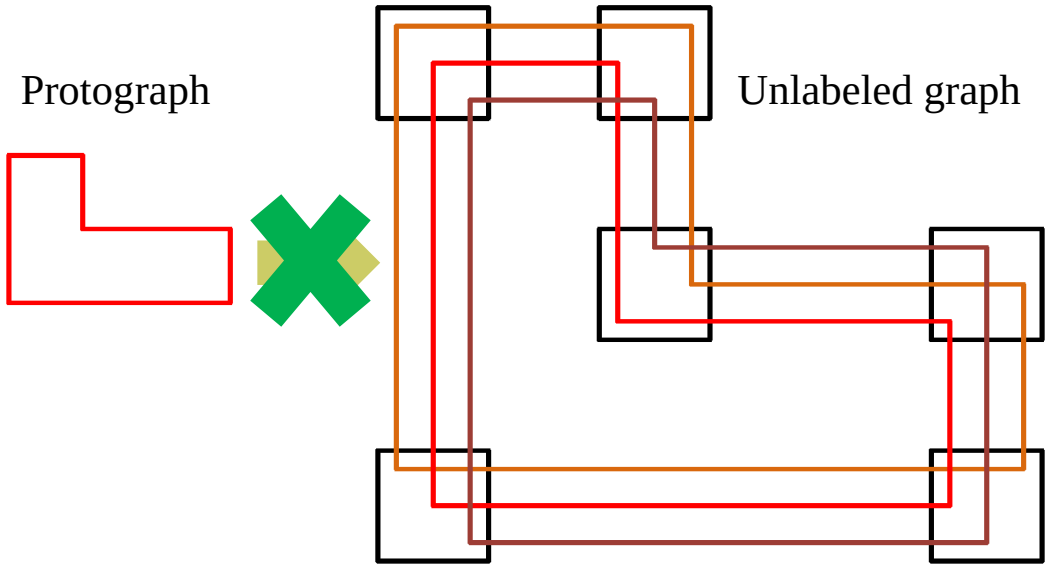
$$f'_{c_1,v_1} + f'_{c_2,v_2} + f'_{c_3,v_3} \equiv f'_{c_1,v_2} + f'_{c_2,v_3} + f'_{c_3,v_1} \pmod{z}, \tag{8}$$

    where $f'_{i',j'}$ are the circulant powers associated with the 1's in $\mathbf{\Pi}_{1}^{\chi,\mathrm{p}}$ (obtained from $f_{i,j}$, which are the powers of $\mathbf{H}^{\mathrm{p}}$).

The Idea of the CPO

- **Cycles in the optimal protograph are not reflected in the unlabeled graph after lifting.**
  - Apply the CPO to break the condition in (8) for as many cycles in the optimized graph of $\mathbf{\Pi}_1^{\chi,\mathrm{p}}$ as possible.

Example: $z = 3$.



The Steps of the CPO (1 of 2)

- **The steps of the CPO are:**
  1. Assign initial circulant powers to the $\gamma\kappa$ 1's in $\mathbf{H}^{\mathrm{p}}$.
  2. Construct $\mathbf{\Pi}_1^{\chi,\mathrm{p}}$ using $\mathbf{H}^{\mathrm{p}}$ and $\mathbf{t}^*$.
  3. Define a variable $\psi_{i,j}$ (resp., $\psi'_{i',j'}$) for each 1 in $\mathbf{H}^{\mathrm{p}}$ (resp., $\mathbf{\Pi}_1^{\chi,\mathrm{p}}$).
  4. Locate all cycles of lengths 6 and 4 in $\mathbf{\Pi}_1^{\chi,\mathrm{p}}$.
  5. For each cycle of length 6, check whether (8) is satisfied or not.
  6. If (8) is satisfied, mark the cycle as active cycle.
  7. The number of active cycles spanning k consecutive replicas in $\mathbf{\Pi}_1^{\chi,\mathrm{p}}$ is $(\chi - k + 1)F_1^{k,\mathrm{a}}$.
  8. Compute the number of $(3, 3(\gamma - 2))$ UASs/UTSs in $\mathbf{H}_{\mathrm{SC}}$ via:

$$F_{\mathrm{SC}} = \sum_{k=1}^{\chi} \left((L - k + 1) F_1^{k,\mathrm{a}} \right) z, \tag{9}$$

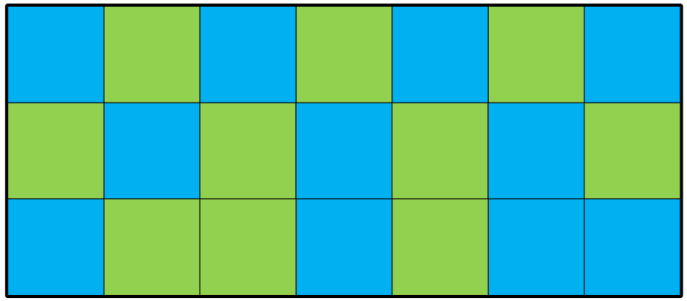
The Steps of the CPO (2 of 2)

9. Count the number of active cycles each 1 in $\mathbf{\Pi}_1^{\chi,\mathrm{p}}$ is involved in. Assign appropriate weights.
10. Store the weighted counts in $\psi'_{i',j'}$ and calculate the variables $\psi_{i,j}$.
11. Sort the $\gamma\kappa$ 1's in $\mathbf{H}^{\mathrm{p}}$ descendingly according to the counts in variables $\psi_{i,j}$.

12. Heuristically, pick a subset of 1's from the top of this list, and change the circulant powers associated with them.
13. Using these interim new powers, do Steps 5, 6, 7, and 8.
14. If F_{SC} is reduced while maintaining no cycles of length 4, update F_{SC} and the circulant powers, then go to Step 9.
15. Otherwise, return to Step 12.
16. Iterate until the target F_{SC} is achieved, or no more reduction in F_{SC}.

Example on the OO-CPO Approach

- **The objective is to design an SC code with $\gamma = 3$, $\kappa = 7$, $z = 7$, $m = 1$, and $L = 30$ using the OO-CPO approach.**
  - OO: Solving (7) yields an optimal vector $\boldsymbol{t}^* = [3\ 4\ 3\ 0\ 1\ 2\ 0]^{\mathrm{T}}$, which gives $F^* = 1170$ cycles of length 6 in the graph of $\mathbf{H}_{\mathrm{SC}}^{\mathrm{p}}$.
  - CPO: Applying the CPO afterwards results in only 203 $(3, 3)$ UASs in the graph of $\mathbf{H}_{\mathrm{SC}}$. (P.S. $203 = 1 \times 29 \times 7$.)



$\mathbf{H}_0^{\mathrm{p}}$ $\mathbf{H}_1^{\mathrm{p}}$

0	1	3	5	2	4	1
0	1	2	3	4	5	6
0	5	0	2	4	6	2

Circulant powers in **H**

- **OO-CPO is not only better, but also faster than other techniques!**
  - We can pick any optimal vector as they all perform the same way.

Number of $(3, 3)$ UASs in Different SC Codes

- **All the codes here have $\gamma = 3$, $m = 1$, and $L = 30$.**

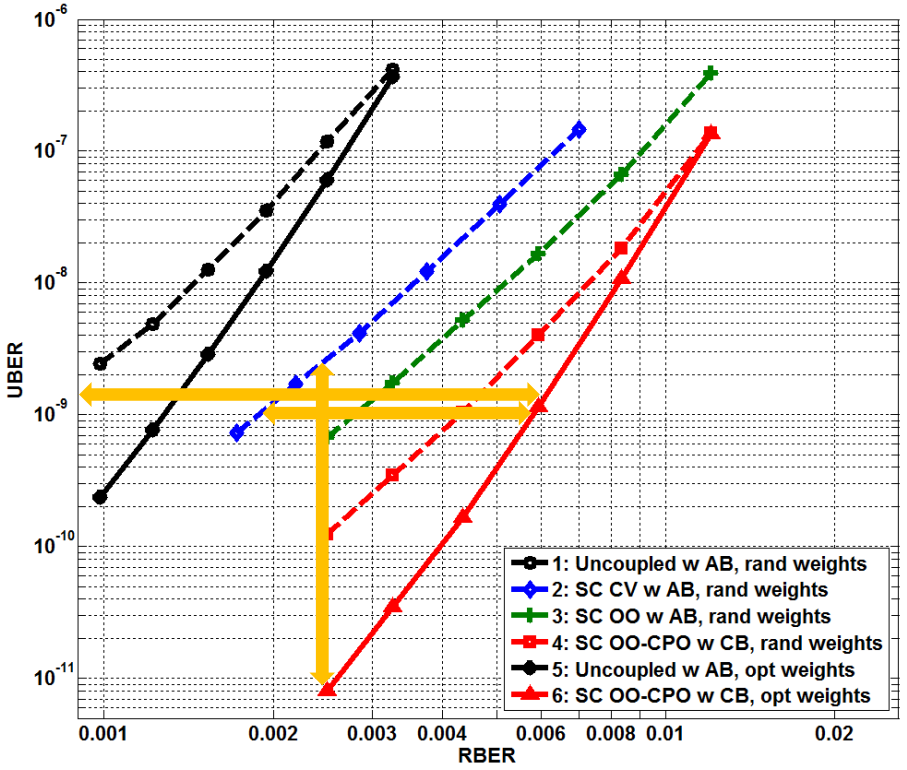
Design Technique	Number of $(3, 3)$ UASs			
	$\kappa = z = 7$	$\kappa = z = 11$	$\kappa = z = 13$	$\kappa = z = 17$
Uncoupled with AB	8820	36300	60840	138720
SC CV with AB	3290	14872	25233	59024
SC MO with AB	609	3850	6851	15997
SC best with AB	609	3520		
SC OO-CPO with CB	203	2596	5356	14960

- **The OO-CPO achieves:**
  - Between 6.5% and 66.7% reduction compared with the MO.
  - Between 74.7% and 93.8% reduction compared with the CV.

- **The OO-CPO also defeats that best (reached exhaustively) that can be achieved with array-based (AB) circulants!**

Significant Performance Gains on Flash!

- **Channel: normal-Laplace mixture (NLM) Flash [Parnell 14].**
  - IBM MLC channel, with 3 reads and sector size 512 bytes.
  - RBER is raw BER. UBER is uncorrectable BER (FER/512/8).
- **All the codes have $\gamma = 3$, $\kappa = z = 19$, $m = 1$, $L = 20$, and $q = 4$. (14440 bits and rate 0.834)**
- **The OO-CPO-WCM approach outperforms existing methods:**
  - Code 6 outperforms Code 2 by 2.5 orders of magnitude.
  - Code 6 achieves 200% RBER gain compared with Code 2.

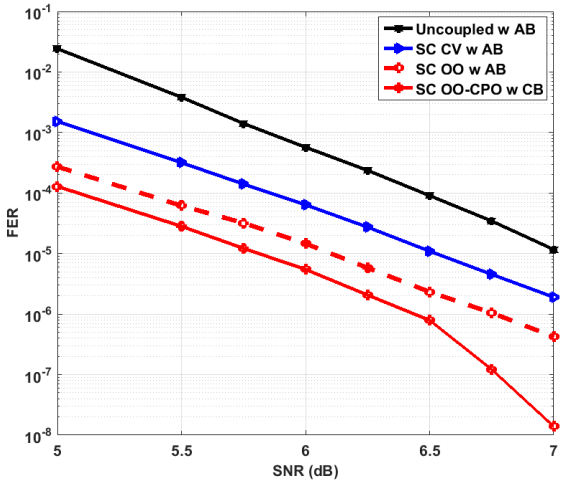


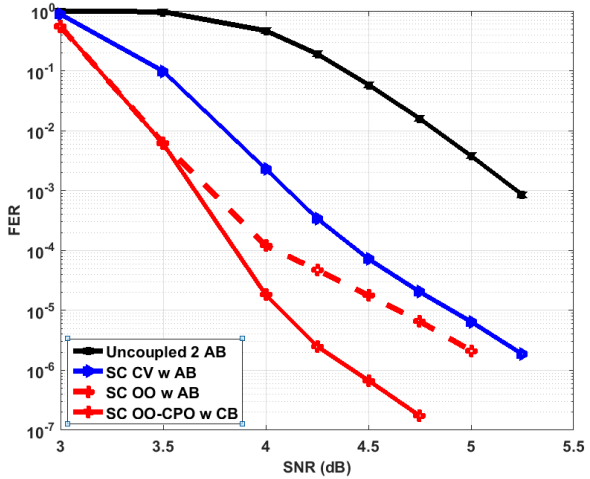
Significant Performance Gains on AWGN!

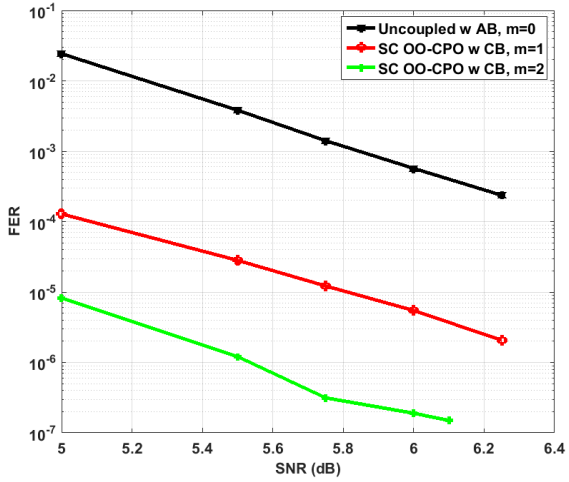
- **There are OO-CPO results for $\gamma \geq 3$ and $m \geq 1$ for asymmetric Flash [Hareedy 17] and AWGN [Esfahanizadeh 19] channels.**
  - We managed to achieve **zero $(3,3)$ UASs** in SC codes having $\gamma = 3$ and $m = 2$ via the OO-CPO approach.

- **All the codes are binary with $\kappa = z = 17$ and $L = 30$ (8670 bits).**

$\gamma = 3$ **and** $m = 1$ $\gamma = 4$ **and** $m = 1$ $\gamma = 3$ **and** $m \in \{1, 2\}$







Notice the waterfall gains!

Presentation Outline

- Motivation and mission statement
- Introduction to graph-based codes
- High performance spatially-coupled codes
- Construction of multi-dimensional codes
- Conclusion and ongoing research

The Goal of MD Construction

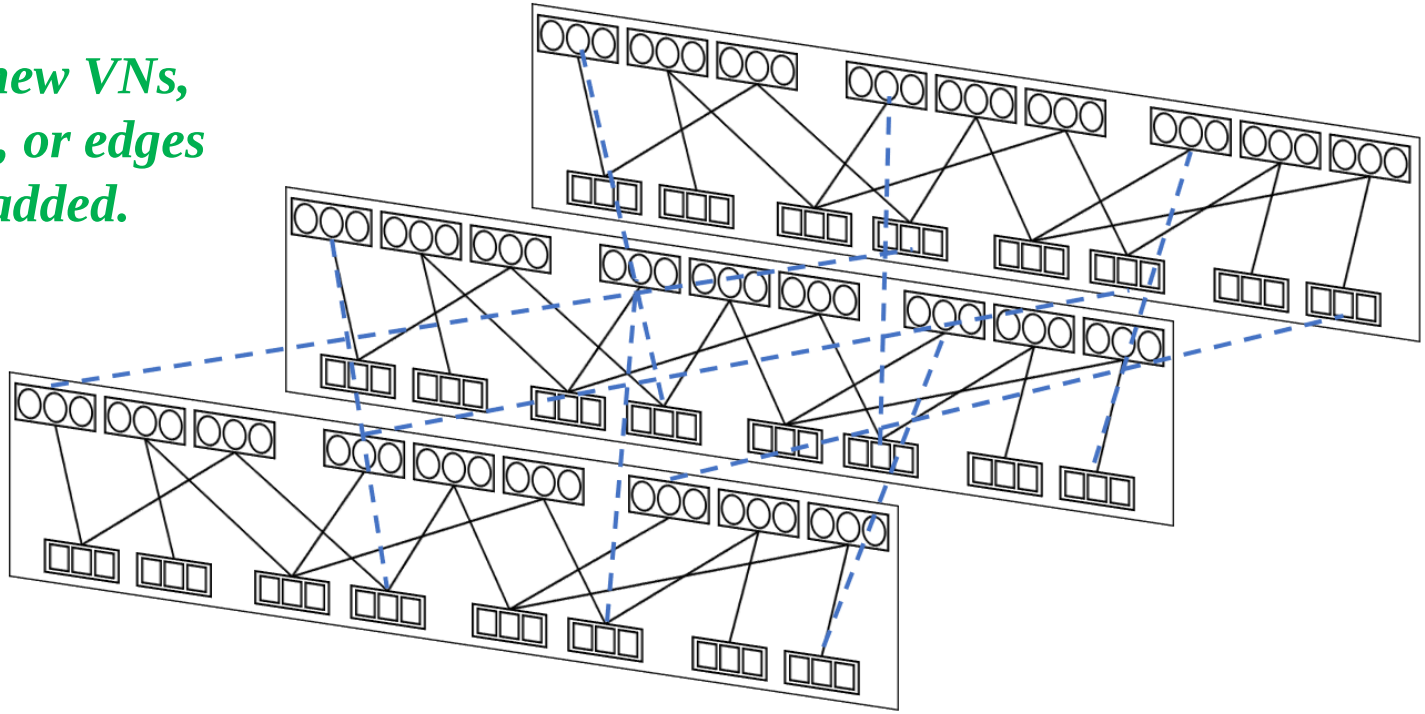
- **We aim at optimally coupling multiple copies of a high performance OD code to suit MD storage devices.**
  - We relocate the most problematic non-zero (NZ) entries to minimize the number of detrimental objects in the MD code.
  - Consider a prime number $M > 2$ of copies of an OD code.
  - The parity-check matrix of the OD (MD) code is $\mathbf{H}_{\text{OD}}$ ($\mathbf{H}_{\text{MD}}$).
  - The matrices used for relocations are $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_{M-1}$, where: $\mathbf{H}_{\text{OD}} = \mathbf{H}'_{\text{OD}} + \sum_{\ell=1}^{M-1} \mathbf{X}_\ell$.

$$\mathbf{H}_{\text{MD}} = \begin{bmatrix} \mathbf{H}'_{\text{OD}} & \mathbf{X}_{M-1} & \mathbf{X}_{M-2} & \cdots & \mathbf{X}_2 & \mathbf{X}_1 \\ \mathbf{X}_1 & \mathbf{H}'_{\text{OD}} & \mathbf{X}_{M-1} & \cdots & \mathbf{X}_3 & \mathbf{X}_2 \\ \mathbf{X}_2 & \mathbf{X}_1 & \mathbf{H}'_{\text{OD}} & \cdots & \mathbf{X}_4 & \mathbf{X}_3 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{X}_{M-2} & \mathbf{X}_{M-3} & \mathbf{X}_{M-4} & \cdots & \mathbf{H}'_{\text{OD}} & \mathbf{X}_{M-1} \\ \mathbf{X}_{M-1} & \mathbf{X}_{M-2} & \mathbf{X}_{M-3} & \cdots & \mathbf{X}_1 & \mathbf{H}'_{\text{OD}} \end{bmatrix}. \quad (10)$$

Example Illustrating Relocations

- **Let $M = 3$. Consider an OD code that is SC ($\mathbf{H}_{\mathrm{OD}} = \mathbf{H}_{\mathrm{SC}}$) with:**
  - $\gamma = 2$, $\kappa = 3$, $z = 3$, $m = 1$, and $L = 3$.
- **One circulant is relocated here from the component matrix $\mathbf{H}_1$.**
  - This relocation is applied for all 3 replicas.

No new VNs, CNs, or edges are added.



- **How can we optimally perform these relocations?**

Representing an Object via Its Basic Cycles

- **Consider again an (a, d_1) UAS (elementary).**
  - The number of degree-2 CNs is:

$$d_2 = \frac{1}{2}(a\gamma - d_1).$$

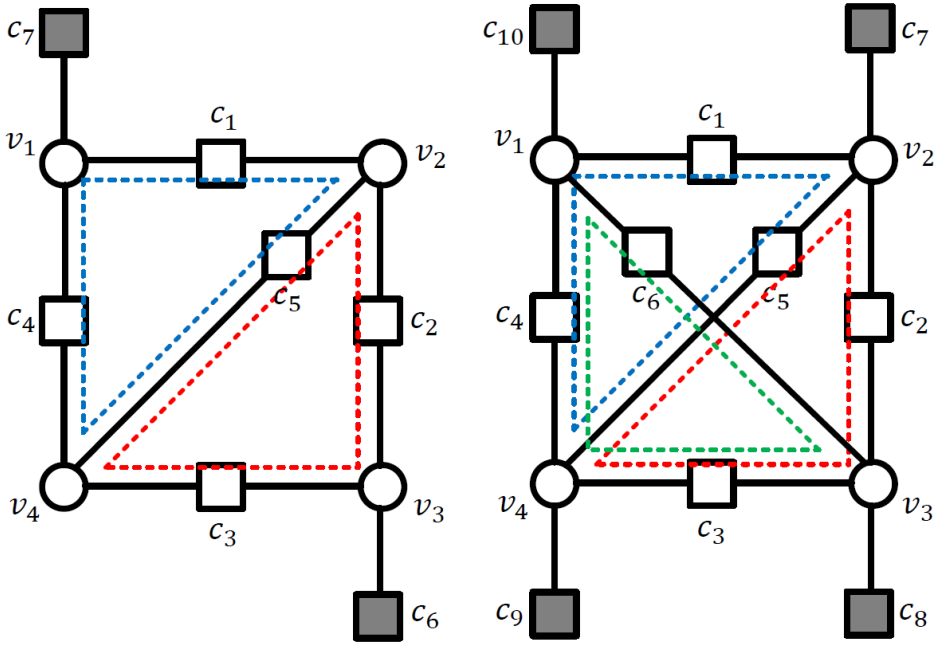
Recall that a is the number of VNs, and d_1 is the number of degree-1 CNs.

  - The number of basic cycles in the cycle basis of the UAS is:

$$n_{\mathrm{f}} = d_2 - a + 1 = \frac{1}{2}(a(\gamma - 2) - d_1 + 2).$$

  - Examples:
    - A $(4, 2)$ UAS, $\gamma = 3$. Here $n_{\mathrm{f}} = 2$.
    - A $(4, 4)$ UAS, $\gamma = 4$. Here $n_{\mathrm{f}} = 3$.

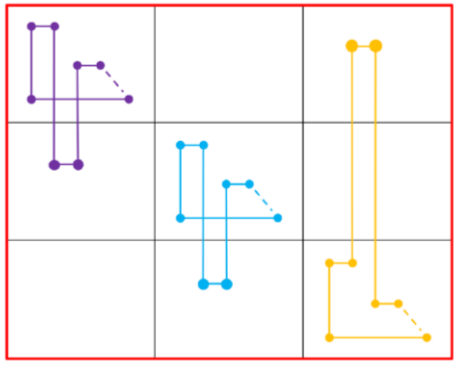
Basic cycles are shown in dotted lines.



What Happens for a Single Cycle?

- **The NZ entries of a cycle of length $2k$ are $\mathcal{E}_{i_w,j_w}$, $w \in \{1, 2, \dots, 2k\}$.**
  - **The problem:** Finding the condition on a set of relocations of some of these entries to be unsuccessful.
  - Unsuccessful relocations: M cycles of length $2k$ remain.

- **The MD mapping R for all entries is defined as follows:**
  - R: $\{\mathcal{E}_{i,j}, \forall i, j\} \rightarrow \{0, 1, \dots, M-1\}$.
  - $R(\mathcal{E}_{i,j}) = 0$ if $\mathcal{E}_{i,j}$ is kept in $\mathbf{H}'_{\mathrm{OD}}$.
  - $R(\mathcal{E}_{i,j}) = \ell > 0$ if $\mathcal{E}_{i,j}$ is relocated to $\mathbf{X}_\ell$.



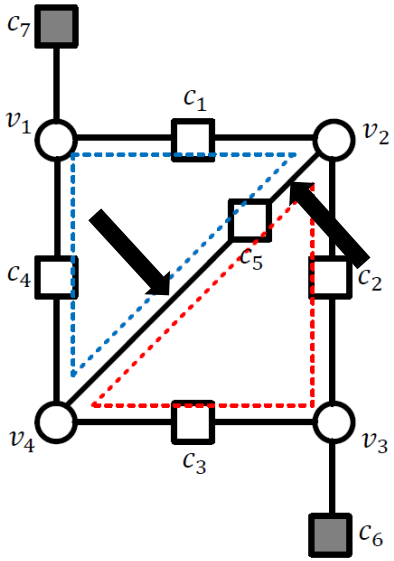
- **The condition of unsuccessful relocations [Esfahanizadeh 18]:**

$$\sum_{w=1}^{2k} (-1)^w R(\mathcal{E}_{i_w,j_w}) \equiv 0 \pmod{M}. \quad (11)$$

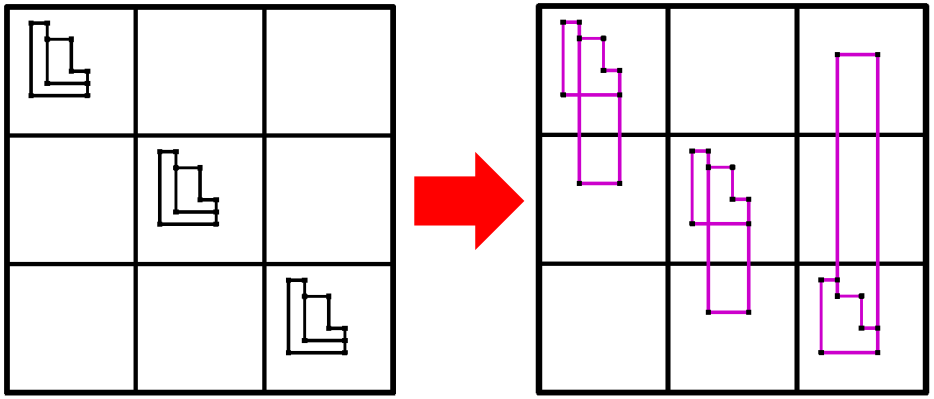
$R(\mathcal{E}_{i_x,j_x}) = R(\mathcal{E}_{i_{x+1},j_{x+1}}) = 1.$
Thus, (11) is satisfied.

Then, We Focus on Objects

- **<u>Theorem 2:</u> The necessary and sufficient condition for a set of relocations of an (a, d_1) UAS to be unsuccessful is:**
  - Equation (11) is satisfied for all basic cycles.
  - Unsuccessful relocations: M (a, d_1) UASs remain.
  - More details in [Hareedy TW-19].

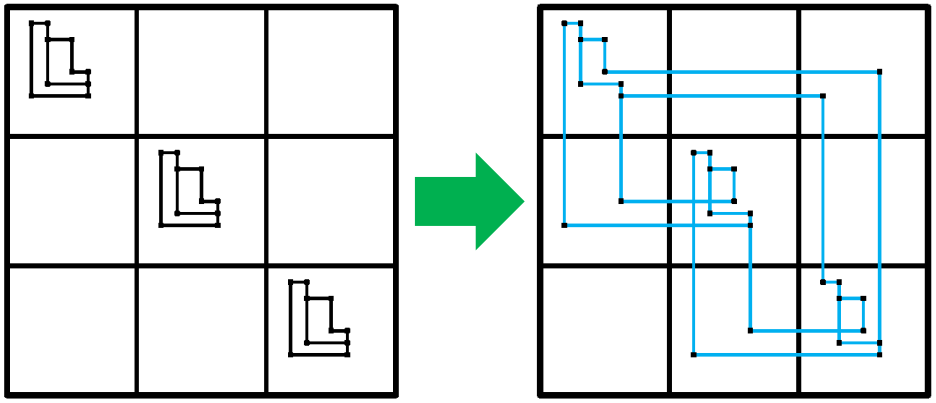


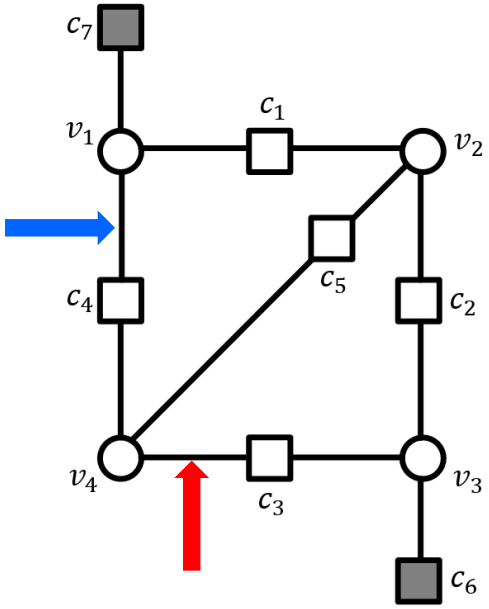
- **Example with a $(4, 2)$ UAS and $M = 3$:**
  - $R(\mathcal{E}_{c_5,v_2}) = R(\mathcal{E}_{c_5,v_4}) = 1$.
  - Three $(4, 2)$ UASs remain.

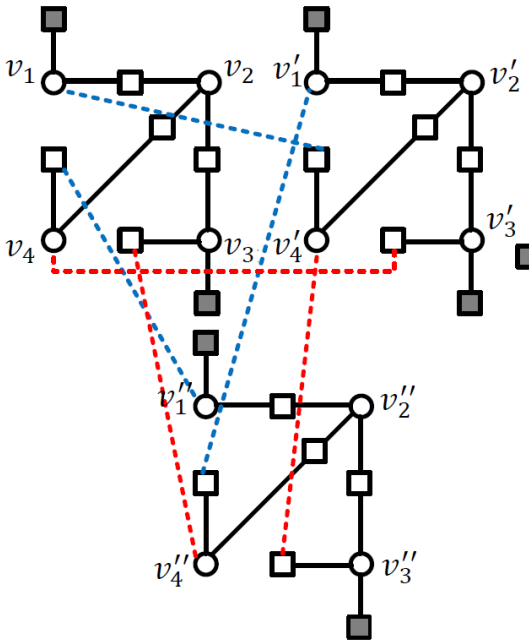


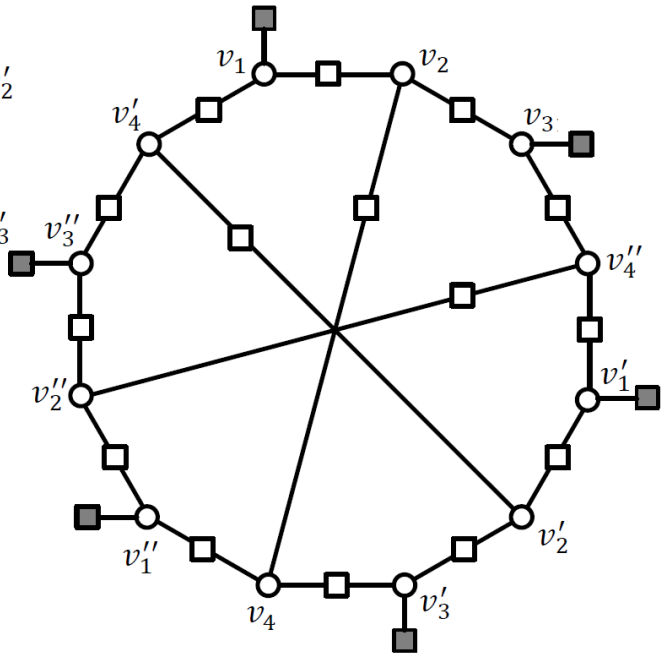
Successful Relocations Graphically

- **Example with a $(4, 2)$ UAS and $M = 3$:**
  - $R(\mathcal{E}_{c_3,v_4}) = R(\mathcal{E}_{c_4,v_1}) = 1.$
  - One $(12, 6)$ UAS appears.
    Three $(4, 2)$ UASs removed!









Parameters of the Codes to Be Compared

- **Our MD code construction algorithm takes relocation decisions based on the majority of the votes of UASs.**

- **The parameters of the codes ($M = 3$):**
  - OD Codes 1 and 3 are SC codes designed via OO-CPO approach.
  - OD Codes 1 and 3 have $\gamma = 3$, $\kappa = z = 19$, $m = 1$, and GF(4).
  - OD Code 2 is a block code having $\gamma = 4$ and GF(2).
  - OD Code 1 has length $= 5054$ bits and rate ≈ 0.82.
  - OD Code 3 has length $= 15162$ bits and rate ≈ 0.83.
  - OD Code 2 has length $= 4240$ bits and rate ≈ 0.90.

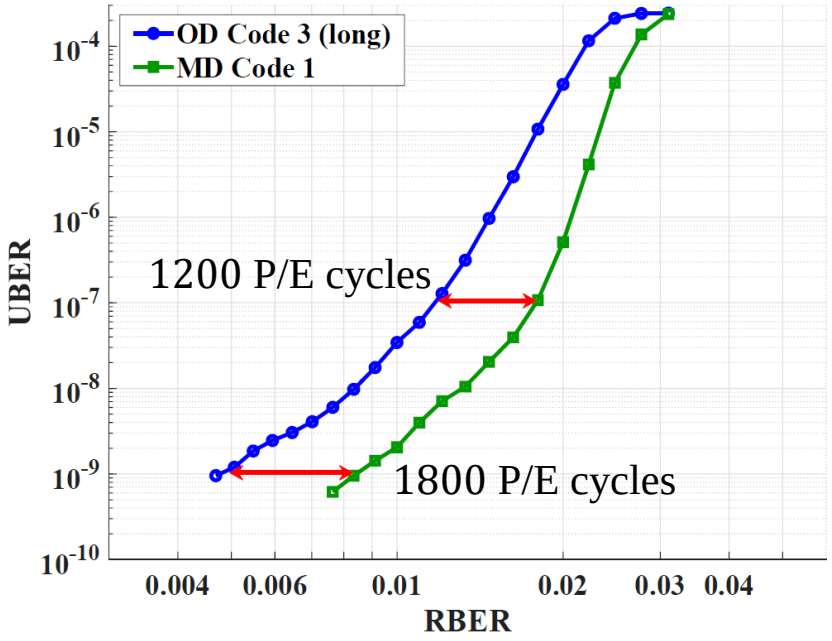
  - OD Code 1 (resp., 2) is the OD code of MD Code 1 (resp., 2).
  - MD Code 1 has length $= 15162$ bits and rate ≈ 0.82 (for Flash).
  - MD Code 2 has length $= 12720$ bits and rate ≈ 0.90 (for AWGN).

Comparison Between OD and MD Codes

- **Number of UASs of interest in OD and MD codes:**

Code	Three OD Code 1 copies	MD Code 1
Number of $(4, 2)$ UASs	4218	0
Code	Three OD Code 2 copies	MD Code 2
Number of $(4, 4)$ UASs	3392	0

- **Performance of OD and MD codes over NLM Flash channel:**



- **Informed MD coupling offers:**
  - Significant reduction in the number of detrimental objects!
  - Significant lifetime gain in Flash (despite similar code parameters)!

Presentation Outline

- **Motivation and mission statement**
- **Introduction to graph-based codes**
- **High performance spatially-coupled codes**
- **Construction of multi-dimensional codes**
- **Conclusion and ongoing research**

Conclusion

- **Modern storage systems require effective ECC techniques to reach the very low error rates they operate at.**
  - Graph-based codes offer excellent performance.
- **High performance SC codes for Flash memories are designed via optimizing partitioning and lifting parameters.**
  - Optimized SC codes outperform block codes of similar parameters.
- **Coupling multiple copies of OD codes to construct MD codes remarkably improves the performance.**
  - Significant Flash lifetime gains are achievable via MD codes.
- **These frameworks open the door for more reliable usage of ultra dense, including multi-dimensional, storage devices.**

Ongoing Research

- **Designing high performance time-variant SC codes (replicas are different) for Flash memories.**
- **Combining novel MD constrained and graph-based codes to maximize the performance gains in MD storage devices.**
- **Developing effective error floor prediction techniques for asymmetric Flash and for MD channels.**
- **Constructing codes offering local and global error correction capability for warm storage devices, e.g., Intel Optane.**

References (1 of 3)

- **[Qin 14] M. Qin *et al.*, "Constrained codes that mitigate inter-cell interference in read/write cycles for flash memories," *IEEE J. Sel. Areas Commun.*, 2014.**
- **[Hareedy AL-19] A. Hareedy and R. Calderbank, "Asymmetric LOCO codes: constrained codes for Flash memories," submitted to *Allerton*, 2019.**
- **[Gallager TH] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA, USA: MIT Press, 1963.**
- **[Dolecek 10] L. Dolecek *et al.*, "Analysis of absorbing sets and fully absorbing sets of array-based LDPC codes," *IEEE Trans. Inf. Theory*, 2010.**
- **[Amiri 14] B. Amiri *et al.*, "Analysis and enumeration of absorbing sets for non-binary graph-based codes," *IEEE Trans. Commun.*, 2014.**
- **[Hareedy 16] A. Hareedy *et al.*, "A general non-binary LDPC code optimization framework suitable for dense Flash memory and magnetic storage," *IEEE J. Sel. Areas Commun.*, 2016.**

References (2 of 3)

- **[Iyengar 12] A. R. Iyengar *et al.*, "Windowed decoding of protograph-based LDPC convolutional codes over erasure channels," *IEEE Trans. Inf. Theory*, 2012.**
- **[Hareedy TI-19] A. Hareedy *et al.*, "A combinatorial methodology for optimizing non-binary graph-based codes: theoretical analysis and applications in data storage," *IEEE Trans. Inf. Theory*, 2019.**
- **[Mitchell 14] D. G. Mitchell *et al.*, "Absorbing set characterization of array-based spatially coupled LDPC codes," in *Proc. IEEE ISIT*, 2014.**
- **[Esfahanizadeh 17] H. Esfahanizadeh *et al.*, "A novel combinatorial framework to construct spatially-coupled codes: minimum overlap partitioning," in *Proc. IEEE ISIT*, 2017.**
- **[Fossorier 04] M. P. C. Fossorier, "Quasi-cyclic low-density parity-check codes from circulant permutation matrices," *IEEE Trans. Inf. Theory*, 2004.**
- **[Parnell 14] T. Parnell *et al.*, "Modelling of the threshold voltage distributions of sub-20nm NAND flash memory," in *Proc. IEEE GLOBECOM*, 2014.**

References (3 of 3)

- **[Hareedy 17] A. Hareedy *et al.*, "High performance non-binary spatially-coupled codes for Flash memories," in *Proc. IEEE ITW*, 2017.**

- **[Esfahanizadeh 19] H. Esfahanizadeh *et al.*, "Finite-length construction of high performance spatially-coupled codes via optimized partitioning and lifting," *IEEE Trans. Commun.*, 2019.**

- **[Esfahanizadeh 18] H. Esfahanizadeh *et al.*, "Multi-dimensional spatially-coupled code design through informed relocation of circulants," in *Proc. Allerton*, 2018.**

- **[Hareedy TW-19] A. Hareedy *et al.*, "Minimizing the number of detrimental objects in multi-dimensional graph-based codes," submitted to *ITW*, 2019.**

- ***Contact:***

  - *Ahmed Hareedy, ahmed.hareedy@duke.edu.*
  - *Robert Calderbank, robert.calderbank@duke.edu.*
  - *Lara Dolecek, dolecek@ee.ucla.edu.*

Thank You