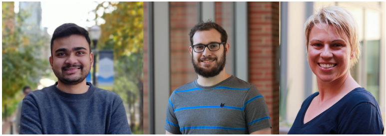
Concentrated Stopping Set Design for Coded Merkle Tree: Improving Security Against Data Availability Attacks in Blockchain Systems

Debarnab Mitra, Lev Tauz, and Lara Dolecek

Electrical and Computer Engineering
University of California, Los Angeles

ITW 2020

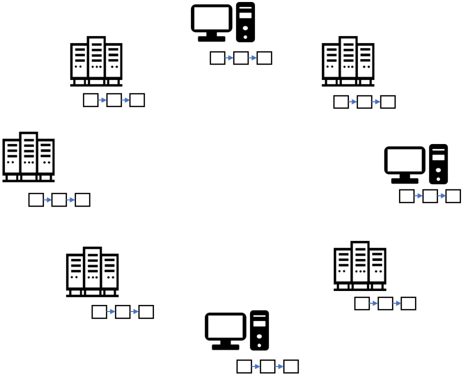


UCLA Samueli School of Engineering

Blockchain

- Distributed Ledger
- Decentralized trust platforms
- Application:
  - Finance and currency
  - Healthcare services
  - Supply chain management
  - Industrial IoT
  - e-voting

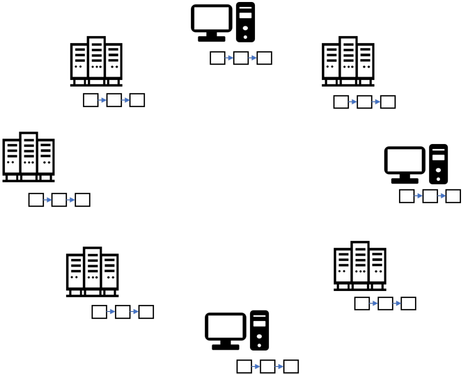
Central Problem: Prohibitive Storage Overhead



- Ledger maintained by a network of nodes

[1] As of 3/12/2021, https://bitinfocharts.com/

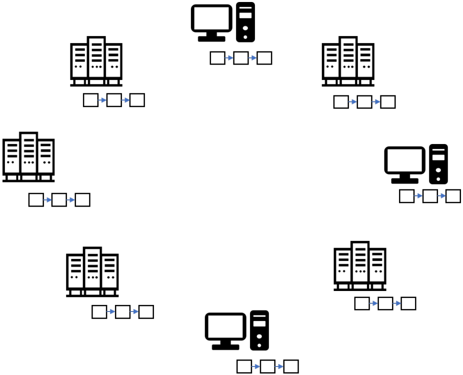
Central Problem: Prohibitive Storage Overhead



- Ledger maintained by a network of nodes
- Each node maintains a local copy of the ledger

[1] As of 3/12/2021, https://bitinfocharts.com/

Central Problem: Prohibitive Storage Overhead

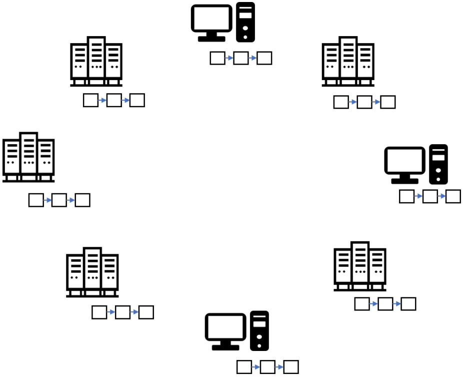


- Ledger maintained by a network of nodes
- Each node maintains a local copy of the ledger

Significant storage overhead

[1] As of 3/12/2021, https://bitinfocharts.com/

Central Problem: Prohibitive Storage Overhead



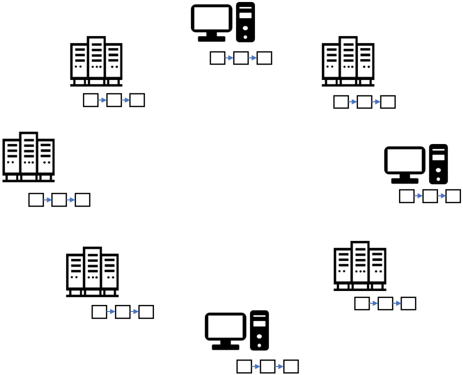
Significant storage overhead

- Ledger maintained by a network of nodes
- Each node maintains a local copy of the ledger

- Bitcoin ledger size $\sim$ 350GB[1]
- Ethereum ledger size $\sim$ 600GB[1]

[1] As of 3/12/2021, https://bitinfocharts.com/

Central Problem: Prohibitive Storage Overhead



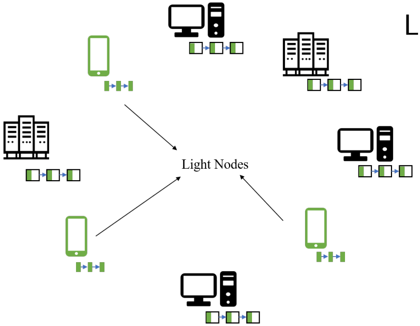
Significant storage overhead

- Ledger maintained by a network of nodes
- Each node maintains a local copy of the ledger
- Prohibitive for resource limited nodes

- Bitcoin ledger size ∼ 350GB[1]
- Ethereum ledger size ∼ 600GB[1]

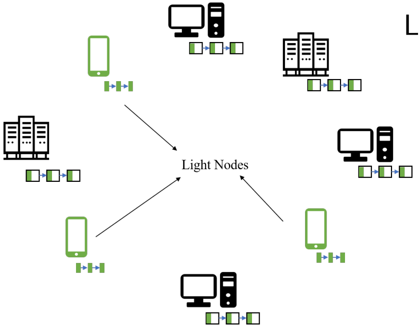
[1] As of 3/12/2021, https://bitinfocharts.com/

Allowing Light Nodes



Light Nodes:

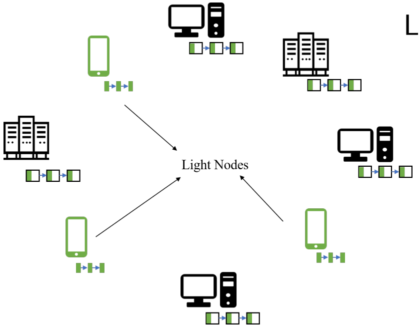
Allowing Light Nodes



Light Nodes:

- Only store block headers
  (total size $\sim$ 1GB for Ethereum)

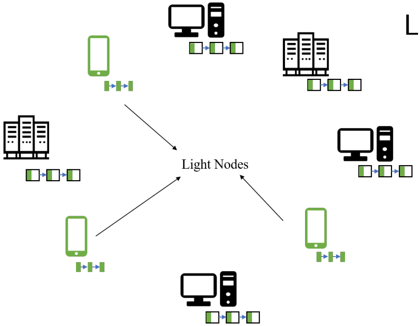
Allowing Light Nodes



Light Nodes:

- Only store block headers
  (total size $\sim$ 1GB for Ethereum)
- Can verify transaction inclusion in a block

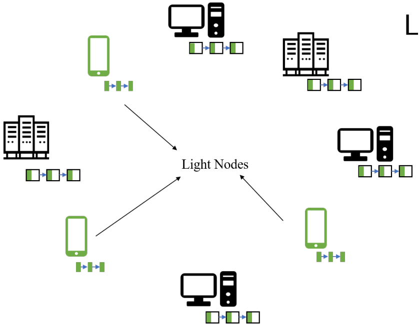
Allowing Light Nodes



Light Nodes:

- Only store block headers
  (total size $\sim$ 1GB for Ethereum)
- Can verify transaction inclusion in a block
- Cannot verify transaction correctness

Allowing Light Nodes



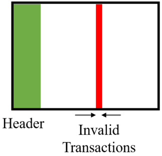
Light Nodes:

- Only store block headers (total size $\sim$ 1GB for Ethereum)
- Can verify transaction inclusion in a block
- Cannot verify transaction correctness $\rightarrow$ Rely on honest Full nodes for fraud notification

Data Availability(DA) Attack

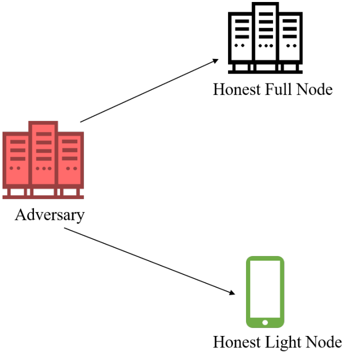
Systems with light nodes and a dishonest majority of full nodes are vulnerable to DA attacks [Al-Bassam '18], [Yu '19]

Adversary creates an invalid block

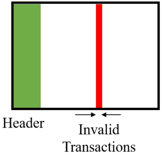


Data Availability(DA) Attack

Systems with light nodes and a dishonest majority of full nodes are vulnerable to DA attacks [Al-Bassam '18], [Yu '19]

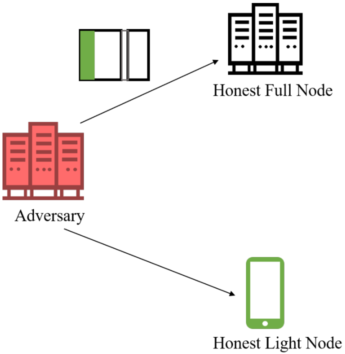


Adversary creates an invalid block

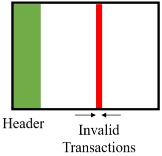


Data Availability(DA) Attack

Systems with light nodes and a dishonest majority of full nodes are vulnerable to DA attacks [Al-Bassam '18], [Yu '19]



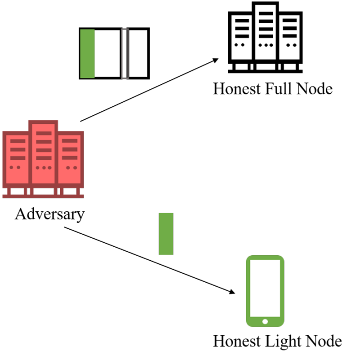
Adversary creates an invalid block



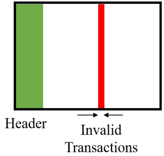
- Adversary: Provides block to Full node but hides invalid portion

Data Availability(DA) Attack

Systems with light nodes and a dishonest majority of full nodes are vulnerable to DA attacks [Al-Bassam '18], [Yu '19]



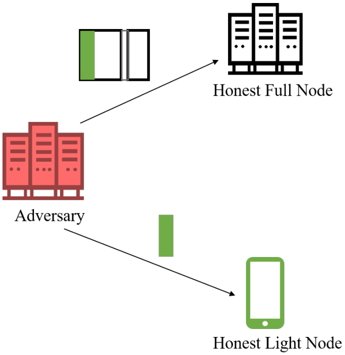
Adversary creates an invalid block



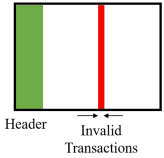
- Adversary: Provides block to Full node but hides invalid portion
  Provides header to Light node

Data Availability(DA) Attack

Systems with light nodes and a dishonest majority of full nodes are vulnerable to DA attacks [Al-Bassam '18], [Yu '19]



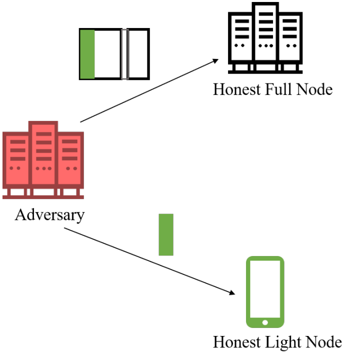
Adversary creates an invalid block



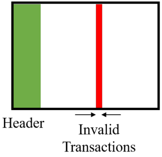
- Adversary: Provides block to Full node but hides invalid portion
  Provides header to Light node
- Honest Nodes: Cannot verify missing transactions

Data Availability(DA) Attack

Systems with light nodes and a dishonest majority of full nodes are vulnerable to DA attacks [Al-Bassam '18], [Yu '19]



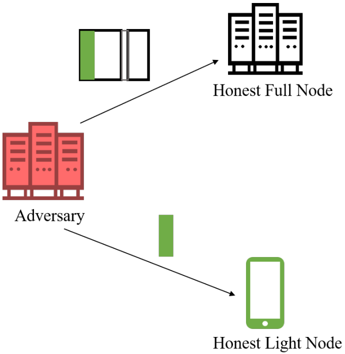
Adversary creates an invalid block



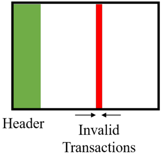
- Adversary: Provides block to Full node but hides invalid portion
  Provides header to Light node
- Honest Nodes: Cannot verify missing transactions $\rightarrow$ No fraud proof

Data Availability(DA) Attack

Systems with light nodes and a dishonest majority of full nodes are vulnerable to DA attacks [Al-Bassam '18], [Yu '19]



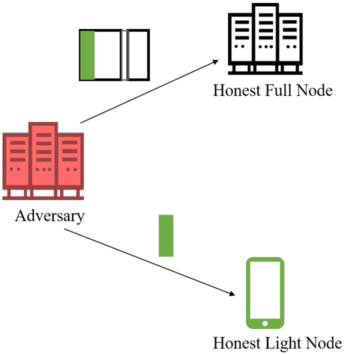
Adversary creates an invalid block



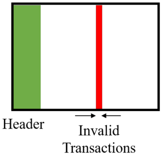
- Adversary: Provides block to Full node but hides invalid portion
  Provides header to Light node
- Honest Nodes: Cannot verify missing transactions $\rightarrow$ No fraud proof
- Light Nodes: No fraud proof

Data Availability(DA) Attack

Systems with light nodes and a dishonest majority of full nodes are vulnerable to DA attacks [Al-Bassam '18], [Yu '19]

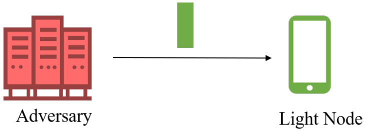


Adversary creates an invalid block

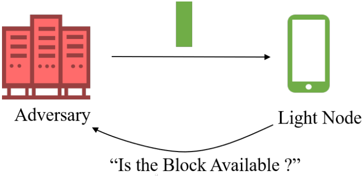


- Adversary: Provides block to Full node but hides invalid portion
  Provides header to Light node
- Honest Nodes: Cannot verify missing transactions $\rightarrow$ No fraud proof
- Light Nodes: No fraud proof $\rightarrow$ accept the header.

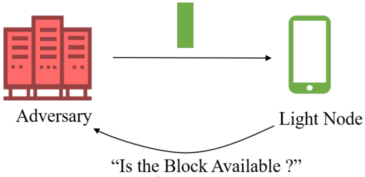
Ensuring Data Availability



Ensuring Data Availability

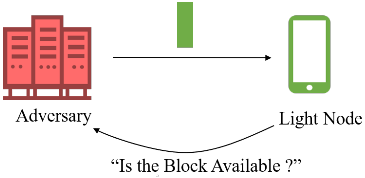


Ensuring Data Availability



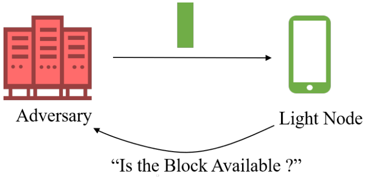
- Anonymously request/sample few random chunks of the block

Ensuring Data Availability

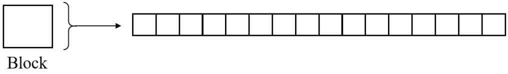


- Anonymously request/sample few random chunks of the block
- Adversary can hide a small portion

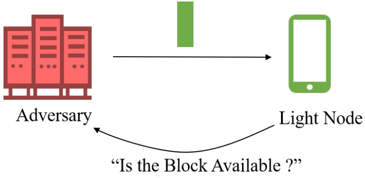
Ensuring Data Availability



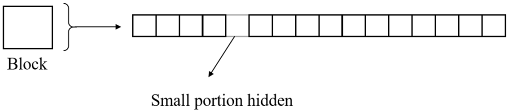
- Anonymously request/sample few random chunks of the block
- Adversary can hide a small portion



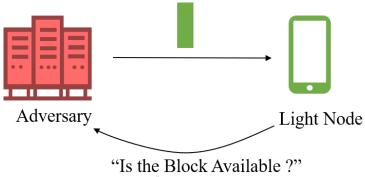
Ensuring Data Availability

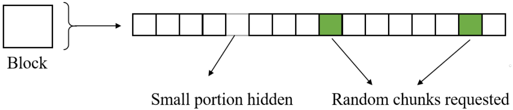


- Anonymously request/sample few random chunks of the block
- Adversary can hide a small portion



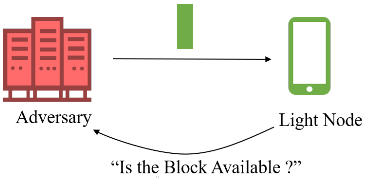
Ensuring Data Availability



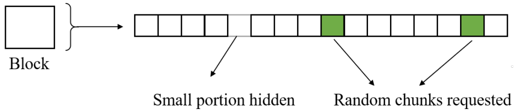


- Anonymously request/sample few random chunks of the block
- Adversary can hide a small portion

Ensuring Data Availability

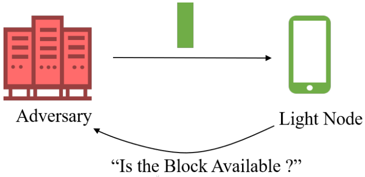


- Anonymously request/sample few random chunks of the block
- Adversary can hide a small portion

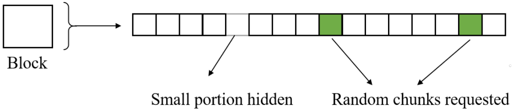


Probability of failure using 2 random samples:

Ensuring Data Availability



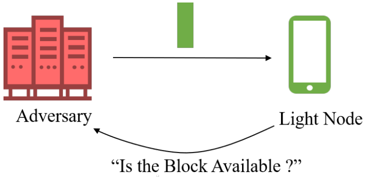
- Anonymously request/sample few random chunks of the block
- Adversary can hide a small portion



Probability of failure using 2 random samples:

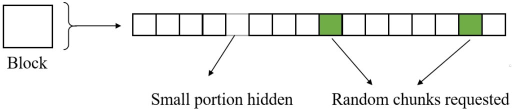
$$\left(1-\frac{1}{16}\right)\left(1-\frac{1}{15}\right)=0.87$$

Ensuring Data Availability



- Anonymously request/sample few random chunks of the block
- Adversary can hide a small portion

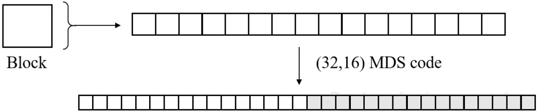
No coding:



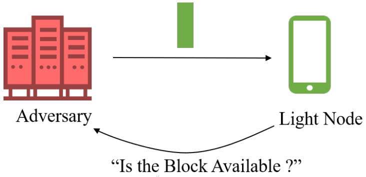
Probability of failure using 2 random samples:

$$\left(1 - \frac{1}{16}\right)\left(1 - \frac{1}{15}\right) = 0.87$$

Erasure coding:

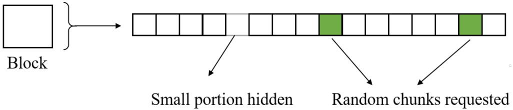


Ensuring Data Availability



- Anonymously request/sample few random chunks of the block
- Adversary can hide a small portion

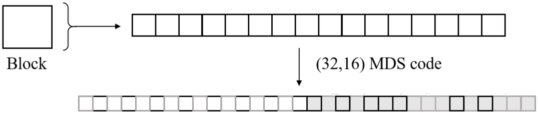
No coding:



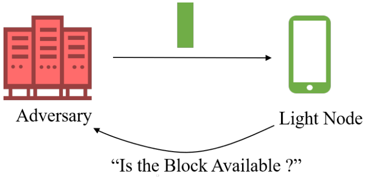
Probability of failure using 2 random samples:

$$\left(1 - \frac{1}{16}\right)\left(1 - \frac{1}{15}\right) = \color{red}{0.87}$$

Erasure coding:

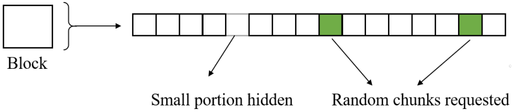


Ensuring Data Availability



- Anonymously request/sample few random chunks of the block
- Adversary can hide a small portion

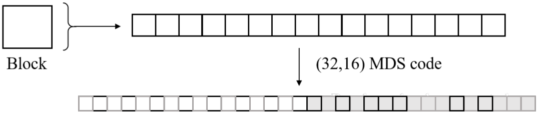
No coding:



Probability of failure using 2 random samples:

$$\left(1 - \frac{1}{16}\right)\left(1 - \frac{1}{15}\right) = 0.87$$

Erasure coding:



Probability of failure using 2 random samples:

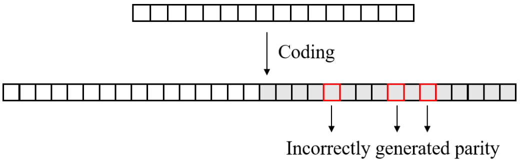
$$\left(1 - \frac{17}{32}\right)\left(1 - \frac{17}{31}\right) = 0.21$$

Choice of Code Matters

Choice of Code Matters

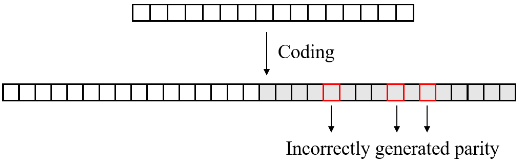
- Incorrect coding attack:

Choice of Code Matters



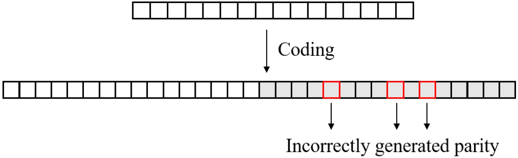
- Incorrect coding attack:
  - Adversary sends incorrectly coded block to Full Nodes

Choice of Code Matters



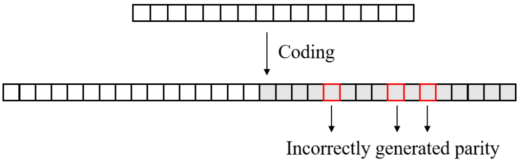
- Incorrect coding attack:
  - Adversary sends incorrectly coded block to Full Nodes
  - Honest Full nodes can detect and send incorrect coding proof
  - Incorrect coding proof size: $\mathcal{O}$(sparsity of parity check equations)

Choice of Code Matters



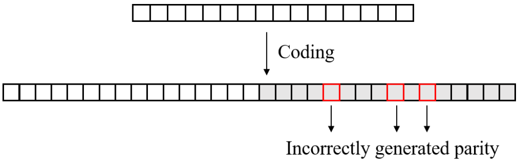
- Incorrect coding attack:
  - Adversary sends incorrectly coded block to Full Nodes
  - Honest Full nodes can detect and send incorrect coding proof
  - Incorrect coding proof size: $\mathcal{O}$(sparsity of parity check equations)
  - MDS codes: proof size $= \mathcal{O}$(block size)

Choice of Code Matters



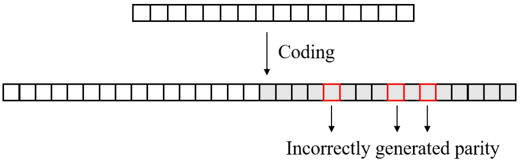
- Incorrect coding attack:
  - Adversary sends incorrectly coded block to Full Nodes
  - Honest Full nodes can detect and send incorrect coding proof
  - Incorrect coding proof size: $\mathcal{O}$(sparsity of parity check equations)
  - MDS codes: proof size $= \mathcal{O}$(block size)
- Decoding complexity

Choice of Code Matters



- Incorrect coding attack:
  - Adversary sends incorrectly coded block to Full Nodes
  - Honest Full nodes can detect and send incorrect coding proof
  - Incorrect coding proof size: $\mathcal{O}$(sparsity of parity check equations)
  - MDS codes: proof size $= \mathcal{O}$(block size)
- Decoding complexity
- Undecodable ratio α

Choice of Code Matters



- Incorrect coding attack:
  - Adversary sends incorrectly coded block to Full Nodes
  - Honest Full nodes can detect and send incorrect coding proof
  - Incorrect coding proof size: $\mathcal{O}$(sparsity of parity check equations)
  - MDS codes: proof size $= \mathcal{O}$(block size)
- Decoding complexity
- Undecodable ratio α
  - Probability of Light node failure using s random samples $= (1-\alpha)^s$

LDPC Codes: A Strong Contender

LPDC codes:

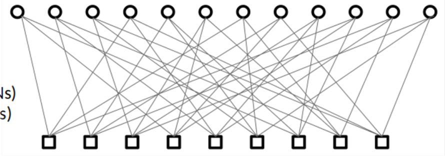
- Characterized by a sparse parity check matrix

LDPC Codes: A Strong Contender

LPDC codes:

- Characterized by a sparse parity check matrix
- Tanner Graph

circles: variable nodes (VNs)
squares: check nodes (CNs)

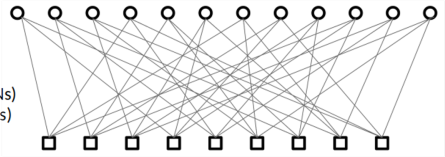


LDPC Codes: A Strong Contender

LPDC codes:

- Characterized by a sparse parity check matrix
- Tanner Graph

circles: variable nodes (VNs)
squares: check nodes (CNs)

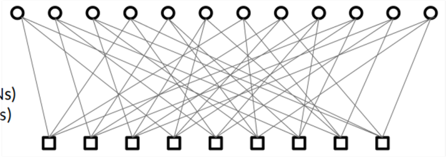


LDPC codes have been shown to be suitable for this application [Yu' 19]

LDPC Codes: A Strong Contender

LPDC codes:

- Characterized by a sparse parity check matrix
- Tanner Graph



LDPC codes have been shown to be suitable for this application [Yu' 19]

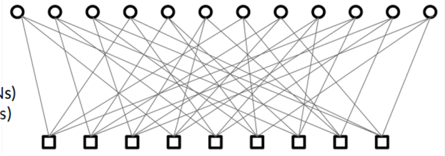
- Small incorrect coding proof size due to small check node degree

LDPC Codes: A Strong Contender

LPDC codes:

- Characterized by a sparse parity check matrix
- Tanner Graph

circles: variable nodes (VNs)
squares: check nodes (CNs)



LDPC codes have been shown to be suitable for this application [Yu' 19]

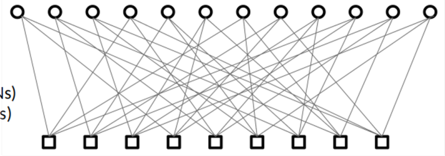
- Small incorrect coding proof size due to small check node degree
- Linear decoding in terms of the block size using peeling decoder

LDPC Codes: A Strong Contender

LPDC codes:

- Characterized by a sparse parity check matrix
- Tanner Graph

circles: variable nodes (VNs)
squares: check nodes (CNs)

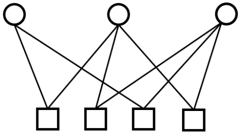


LDPC codes have been shown to be suitable for this application [Yu' 19]

- Small incorrect coding proof size due to small check node degree
- Linear decoding in terms of the block size using peeling decoder
- What about the undecodable ratio?

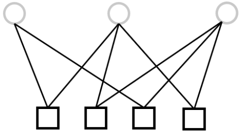
Challenge with LDPC Codes: Small Stopping Sets

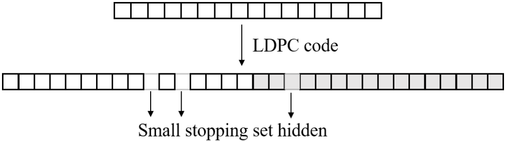
- Substructure in the Tanner Graph



Challenge with LDPC Codes: Small Stopping Sets

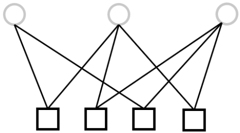
- Substructure in the Tanner Graph
- If hidden, prevents peeling decoder from decoding the block $\rightarrow$ No fraud proof

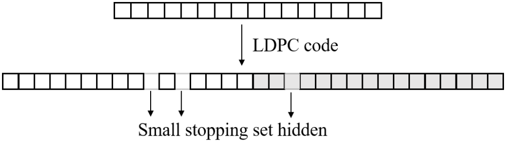




Challenge with LDPC Codes: Small Stopping Sets

- Substructure in the Tanner Graph
- If hidden, prevents peeling decoder from decoding the block $\rightarrow$ No fraud proof



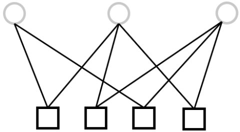


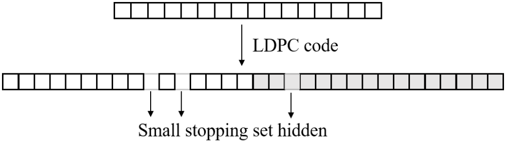
Probability of failure using 2 random samples:

$$\left(1-\frac{3}{32}\right)\left(1-\frac{3}{31}\right) = 0.81$$

Challenge with LDPC Codes: Small Stopping Sets

- Substructure in the Tanner Graph
- If hidden, prevents peeling decoder from decoding the block $\rightarrow$ No fraud proof





Probability of failure using 2 random samples:

$$\left(1 - \frac{3}{32}\right)\left(1 - \frac{3}{31}\right) = 0.81$$

> Our work: Design of specialized LDPC codes with a coupled sampling strategy to achieve a significantly lower probability of failure.

Motivation: Not all VNs are equal

In this work, we considered an adversary which randomly hides a stopping set of a particular size.

Motivation: Not all VNs are equal

In this work, we considered an adversary which randomly hides a stopping set of a particular size.

Lemma

Of all stopping sets (SSs) of size μ, when an adversary randomly hides one of them, and light nodes sample all VNs in the set $\mathcal{L}$, then

Motivation: Not all VNs are equal

In this work, we considered an adversary which randomly hides a stopping set of a particular size.

Lemma

Of all stopping sets (SSs) of size μ, when an adversary randomly hides one of them, and light nodes sample all VNs in the set $\mathcal{L}$, then

$$\text{Probability of failure} = 1 - \begin{array}{c}\textit{fraction of SSs} \\ \textit{of size } \mu \textit{ touched by } \mathcal{L}\end{array}$$

Motivation: Not all VNs are equal

In this work, we considered an adversary which randomly hides a stopping set of a particular size.

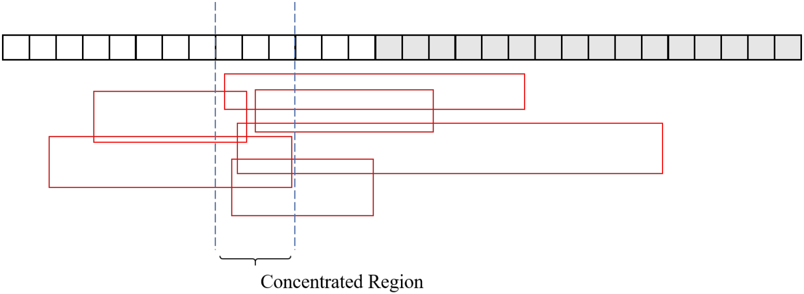
Lemma

Of all stopping sets (SSs) of size μ, when an adversary randomly hides one of them, and light nodes sample all VNs in the set $\mathcal{L}$, then

$$\text{Probability of failure} = 1 - \textit{fraction of SSs of size } \mu \textit{ touched by } \mathcal{L}$$

- Selecting a set $\mathcal{L}$ of VNs which touches large no. of SSs $\rightarrow$ Prob. of failure $\downarrow$

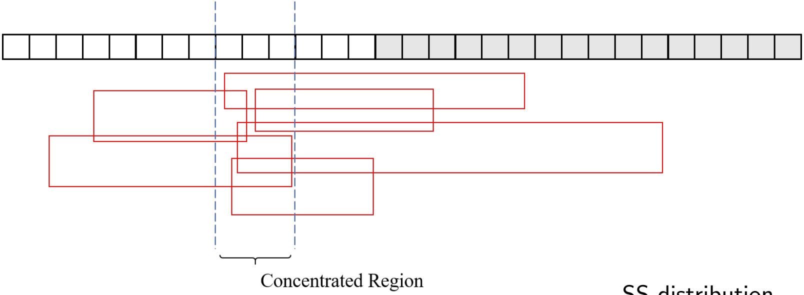
Concentrated Stopping Set Design



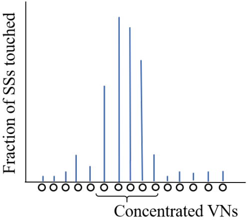
Code Design Idea:

- Concentrate stopping sets to a small section of VNs

Concentrated Stopping Set Design



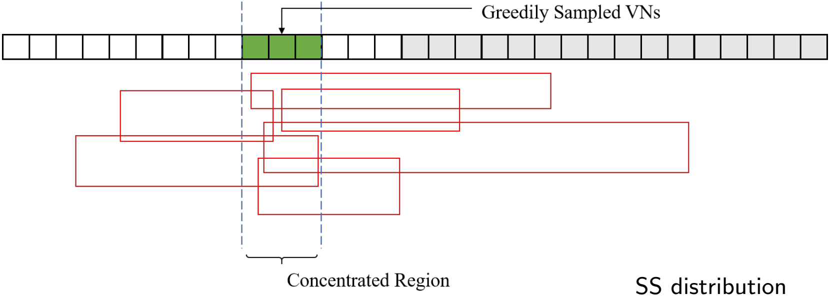
SS distribution



Code Design Idea:

- Concentrate stopping sets to a small section of VNs

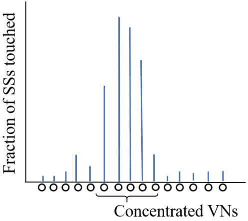
Concentrated Stopping Set Design



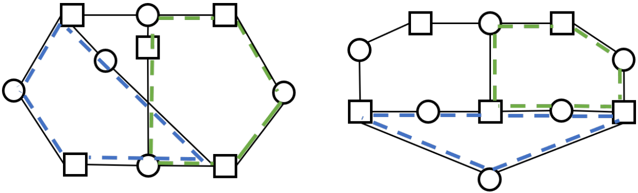
Code Design Idea:

- Concentrate stopping sets to a small section of VNs
- Greedily Sample this small section of VNs

SS distribution

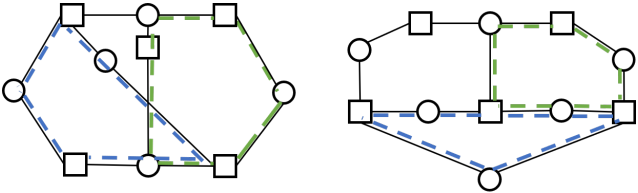


How to Concentrate Stopping Sets?



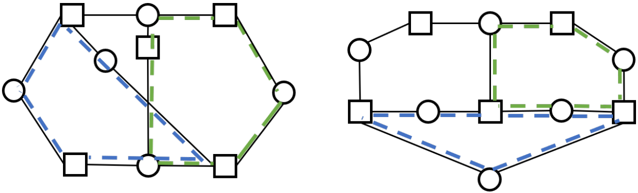
- When there are no degree 1 VNs, stopping sets are either cycles or interconnection of cycles [Tian '03]

How to Concentrate Stopping Sets?



- When there are no degree 1 VNs, stopping sets are either cycles or interconnection of cycles [Tian '03]
- Concentrating cycles $\implies$ Concentrating stopping sets

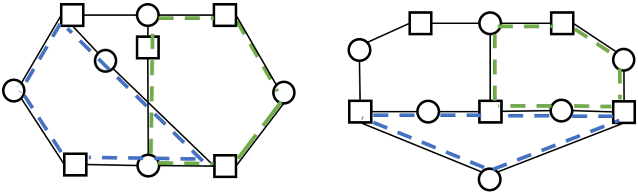
How to Concentrate Stopping Sets?



- When there are no degree 1 VNs, stopping sets are either cycles or interconnection of cycles [Tian '03]
- Concentrating cycles $\implies$ Concentrating stopping sets

- How to design codes with concentrated cycles?

How to Concentrate Stopping Sets?



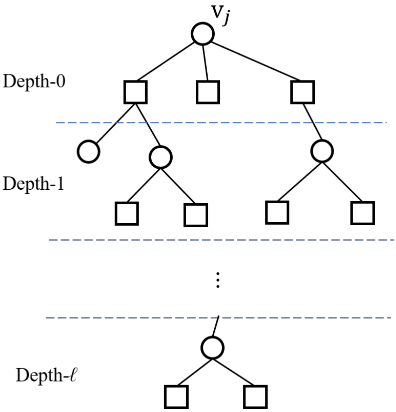
- When there are no degree 1 VNs, stopping sets are either cycles or interconnection of cycles [Tian '03]
- Concentrating cycles $\implies$ Concentrating stopping sets

- How to design codes with concentrated cycles?
  We do so by modifying the well-known Progressive Edge Growth (PEG) algorithm

PEG Algorithm

- Constructs a Tanner Graph in an edge by edge manner [Xiao '05]

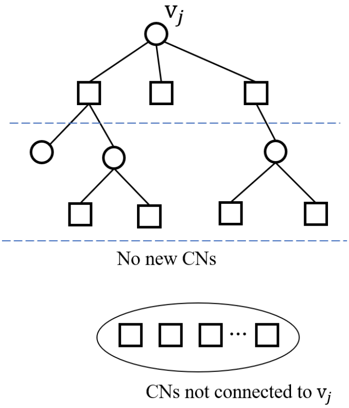
PEG Algorithm



- Constructs a Tanner Graph in an edge by edge manner [Xiao '05]

For each VN v_j
  Expand Tanner Graph in a BFS fashion

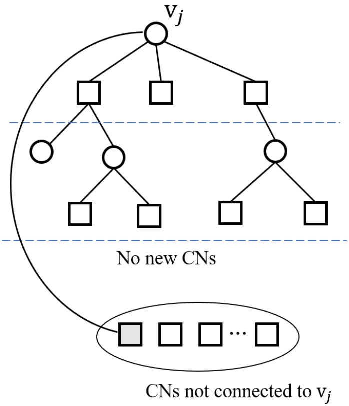
PEG Algorithm



- Constructs a Tanner Graph in an edge by edge manner [Xiao '05]

For each VN v_j
Expand Tanner Graph in a BFS fashion
If $\exists$ CNs not connected to v_j

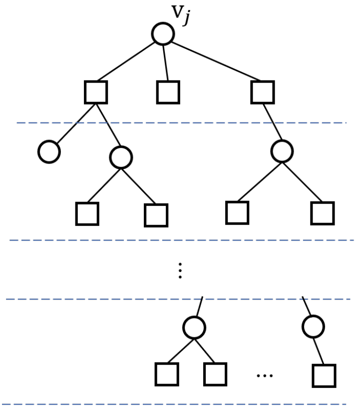
PEG Algorithm



- Constructs a Tanner Graph in an edge by edge manner [Xiao '05]

For each VN v_j
  Expand Tanner Graph in a BFS fashion
  If $\exists$ CNs not connected to v_j
  - Select a CN with min degree not connected to v_j

PEG Algorithm

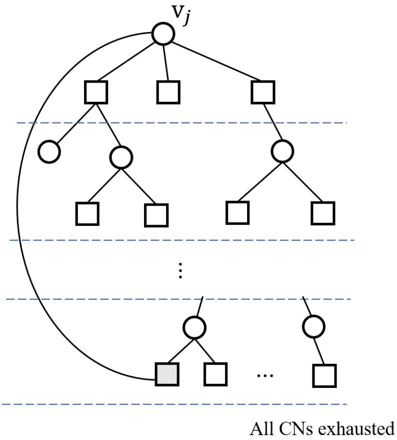


- Constructs a Tanner Graph in an edge by edge manner [Xiao '05]

For each VN v_j
Expand Tanner Graph in a BFS fashion
If $\exists$ CNs not connected to v_j
- Select a CN with min degree not connected to v_j

Else

PEG Algorithm



- Constructs a Tanner Graph in an edge by edge manner [Xiao '05]

For each VN v_j

Expand Tanner Graph in a BFS fashion

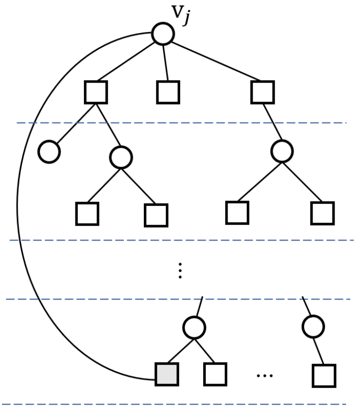
If $\exists$ CNs not connected to v_j

- Select a CN with min degree not connected to v_j

Else

- Find CNs most distant to v_j
- Select one with minimum degree

PEG Algorithm



All CNs exhausted

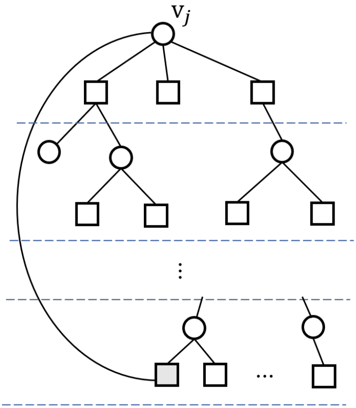
- Constructs a Tanner Graph in an edge by edge manner [Xiao '05]

For each VN v_j
Expand Tanner Graph in a BFS fashion
If $\exists$ CNs not connected to v_j
- Select a CN with min degree not connected to v_j

Else
- Find CNs most distant to v_j
- Select one with minimum degree

New cycles created

PEG Algorithm



- Constructs a Tanner Graph in an edge by edge manner [Xiao '05]

For each VN v_j
Expand Tanner Graph in a BFS fashion
If $\exists$ CNs not connected to v_j
- Select a CN with min degree not connected to v_j

Else
- Find CNs most distant to v_j
- Select one with minimum degree

New cycles created

We modify the CN selection criteria in green to concentrate cycles

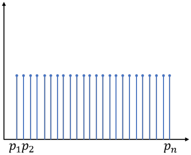
Using Entropy to Concentrate Cycles

For distribution $p = (p_1, p_2, \ldots, p_n)$, Entropy $\mathcal{H}(p) = \sum_{i=1}^{n} p_i \log \frac{1}{p_i}$

Using Entropy to Concentrate Cycles

For distribution $p = (p_1, p_2, \ldots, p_n)$, Entropy $\mathcal{H}(p) = \sum_{i=1}^{n} p_i \log \frac{1}{p_i}$

- Uniform distributions have high entropy

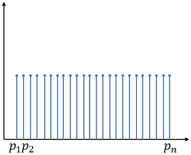


High Entropy

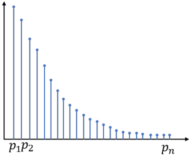
Using Entropy to Concentrate Cycles

For distribution $p = (p_1, p_2, \ldots, p_n)$, Entropy $\mathcal{H}(p) = \sum_{i=1}^{n} p_i \log \frac{1}{p_i}$

- Uniform distributions have high entropy
- Concentrated distributions have low entropy



High Entropy

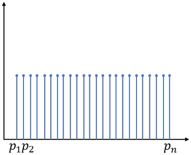


Low Entropy

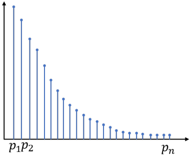
Using Entropy to Concentrate Cycles

For distribution $p = (p_1, p_2, \ldots, p_n)$, Entropy $\mathcal{H}(p) = \sum_{i=1}^{n} p_i \log \frac{1}{p_i}$

- Uniform distributions have high entropy
- Concentrated distributions have low entropy



High Entropy



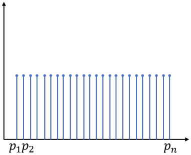
Low Entropy

We want the cycle distributions to be concentrated

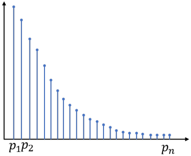
Using Entropy to Concentrate Cycles

For distribution $p = (p_1, p_2, \ldots, p_n)$, Entropy $\mathcal{H}(p) = \sum_{i=1}^{n} p_i \log \frac{1}{p_i}$

- Uniform distributions have high entropy
- Concentrated distributions have low entropy



High Entropy



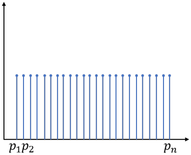
Low Entropy

We want the cycle distributions to be concentrated
$\rightarrow$ Select CNs such that the entropy of the cycle distribution is minimized

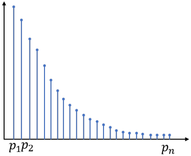
Using Entropy to Concentrate Cycles

For distribution $p = (p_1, p_2, \ldots, p_n)$, Entropy $\mathcal{H}(p) = \sum_{i=1}^{n} p_i \log \frac{1}{p_i}$

- Uniform distributions have high entropy
- Concentrated distributions have low entropy



High Entropy



Low Entropy

EC (Entropy Constrained)-PEG Algorithm
For each VN v_j
Expand Tanner Graph in a BFS fashion
If $\exists$ CNs not connected to v_j
- select a CN with min degree not connected to v_j

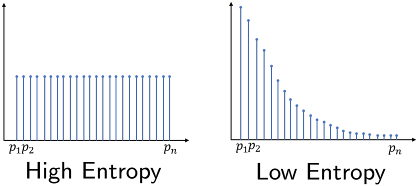
Else *New cycles created*

We want the cycle distributions to be concentrated
$\rightarrow$ Select CNs such that the entropy of the cycle distribution is minimized

Using Entropy to Concentrate Cycles

For distribution $p = (p_1, p_2, \ldots, p_n)$, Entropy $\mathcal{H}(p) = \sum_{i=1}^{n} p_i \log \frac{1}{p_i}$

- Uniform distributions have high entropy
- Concentrated distributions have low entropy



EC (Entropy Constrained)-PEG Algorithm

For each VN v_j

Expand Tanner Graph in a BFS fashion

If $\exists$ CNs not connected to v_j

- select a CN with min degree not connected to v_j

Else *New cycles created*

- Find CNs most distant to v_j

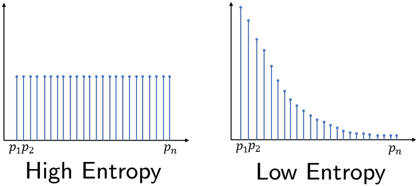
We want the cycle distributions to be concentrated

$\rightarrow$ Select CNs such that the entropy of the cycle distribution is minimized

Using Entropy to Concentrate Cycles

For distribution $p = (p_1, p_2, \ldots, p_n)$, Entropy $\mathcal{H}(p) = \sum_{i=1}^{n} p_i \log \frac{1}{p_i}$

- Uniform distributions have high entropy
- Concentrated distributions have low entropy



EC (Entropy Constrained)-PEG Algorithm

For each VN v_j

Expand Tanner Graph in a BFS fashion

If $\exists$ CNs not connected to v_j

- select a CN with min degree not connected to v_j

Else *New cycles created*

- Find CNs most distant to v_j
- Select CN that results in minimum entropy of resultant cycle distribution

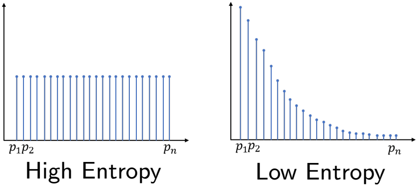
We want the cycle distributions to be concentrated

$\rightarrow$ Select CNs such that the entropy of the cycle distribution is minimized

Using Entropy to Concentrate Cycles

For distribution $p = (p_1, p_2, \ldots, p_n)$, Entropy $\mathcal{H}(p) = \sum_{i=1}^{n} p_i \log \frac{1}{p_i}$

- Uniform distributions have high entropy
- Concentrated distributions have low entropy



EC (Entropy Constrained)-PEG Algorithm

For each VN v_j

Expand Tanner Graph in a BFS fashion

If $\exists$ CNs not connected to v_j

- select a CN with min degree not connected to v_j

Else *New cycles created*

- Find CNs most distant to v_j
- Select CN that results in minimum entropy of resultant cycle distribution
- Update cycle distribution

We want the cycle distributions to be concentrated

$\rightarrow$ Select CNs such that the entropy of the cycle distribution is minimized

EC-PEG Algorithm

- Whenever a new edge, that creates cycles, is added to the Tanner Graph, we update the cycle counts of each VN

EC-PEG Algorithm

- Whenever a new edge, that creates cycles, is added to the Tanner Graph, we update the cycle counts of each VN

VNs $(v_1, v_2, \ldots, v_n)$

EC-PEG Algorithm

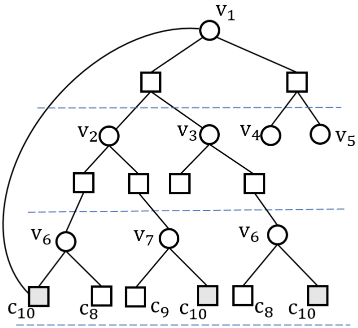
- Whenever a new edge, that creates cycles, is added to the Tanner Graph, we update the cycle counts of each VN

VNs $(v_1, v_2, \ldots, v_n)$

- $\lambda_i^g :=$ No. of cycles of length g that v_i is a part of, $g = 4, 6, 8$

EC-PEG Algorithm

- Whenever a new edge, that creates cycles, is added to the Tanner Graph, we update the cycle counts of each VN

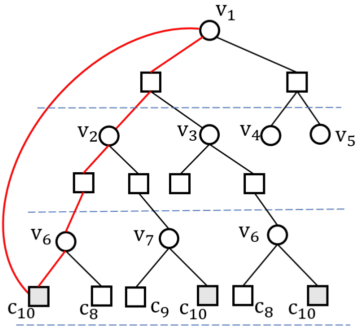


VNs $(v_1, v_2, \ldots, v_n)$

- $\lambda_i^g :=$ No. of cycles of length g that v_i is a part of, $g = 4, 6, 8$

EC-PEG Algorithm

- Whenever a new edge, that creates cycles, is added to the Tanner Graph, we update the cycle counts of each VN

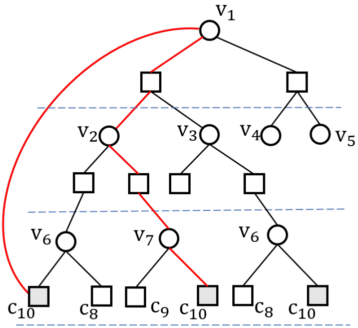


VNs $(v_1, v_2, \ldots, v_n)$

- $\lambda_i^g :=$ No. of cycles of length g that v_i is a part of, $g = 4, 6, 8$

EC-PEG Algorithm

- Whenever a new edge, that creates cycles, is added to the Tanner Graph, we update the cycle counts of each VN

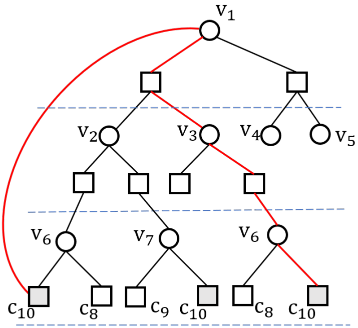


VNs $(v_1, v_2, \dots, v_n)$

- $\lambda_i^g :=$ No. of cycles of length g that v_i is a part of, $g = 4, 6, 8$

EC-PEG Algorithm

- Whenever a new edge, that creates cycles, is added to the Tanner Graph, we update the cycle counts of each VN

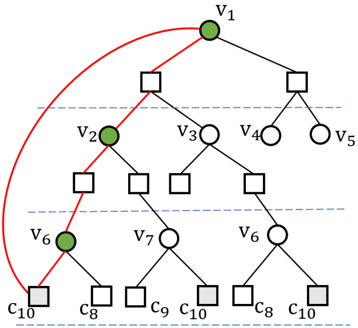


VNs $(v_1, v_2, \ldots, v_n)$

- $\lambda_i^g :=$ No. of cycles of length g that v_i is a part of, $g = 4, 6, 8$

EC-PEG Algorithm

- Whenever a new edge, that creates cycles, is added to the Tanner Graph, we update the cycle counts of each VN

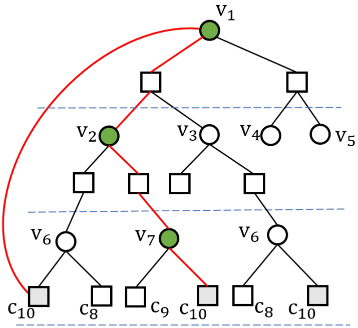


VNs $(v_1, v_2, \ldots, v_n)$

- λ_i^g := No. of cycles of length g that v_i is a part of, $g = 4, 6, 8$
- $$\lambda_1^6 = \lambda_1^6 + 1$$
  $$\lambda_2^6 = \lambda_2^6 + 1$$
  $$\lambda_6^6 = \lambda_6^6 + 1$$

EC-PEG Algorithm

- Whenever a new edge, that creates cycles, is added to the Tanner Graph, we update the cycle counts of each VN



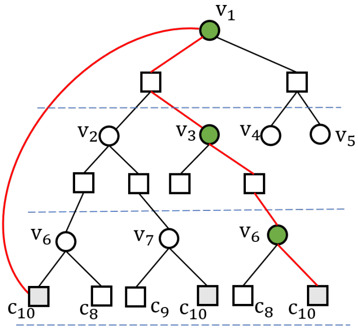
VNs $(v_1, v_2, \ldots, v_n)$

- $\lambda_i^g :=$ No. of cycles of length g that v_i is a part of, $g = 4, 6, 8$
-

$$\lambda_1^6 = \lambda_1^6 + 1$$
$$\lambda_2^6 = \lambda_2^6 + 1$$
$$\lambda_7^6 = \lambda_7^6 + 1$$

EC-PEG Algorithm

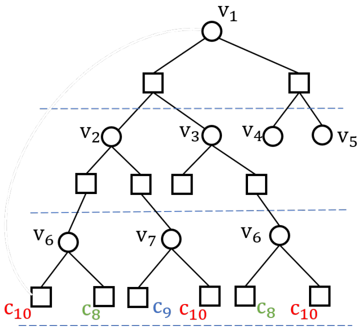
- Whenever a new edge, that creates cycles, is added to the Tanner Graph, we update the cycle counts of each VN



VNs $(v_1, v_2, \ldots, v_n)$

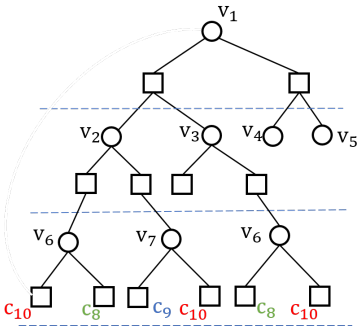
- λ_i^g := No. of cycles of length g that v_i is a part of, $g = 4, 6, 8$
- $$\lambda_1^6 = \lambda_1^6 + 1$$
  $$\lambda_3^6 = \lambda_3^6 + 1$$
  $$\lambda_6^6 = \lambda_6^6 + 1$$

EC-PEG Algorithm: CN Selection Procedure



Candidate CNs : c_8, c_9, c_{10}

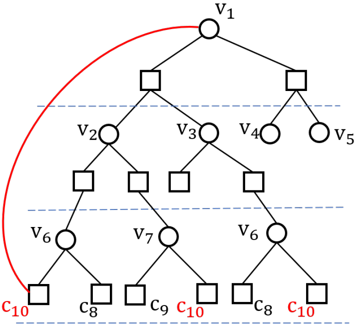
EC-PEG Algorithm: CN Selection Procedure



Candidate CNs : c_8, c_9, c_{10}

- For each CN candidate, calculate the resultant VN cycle counts

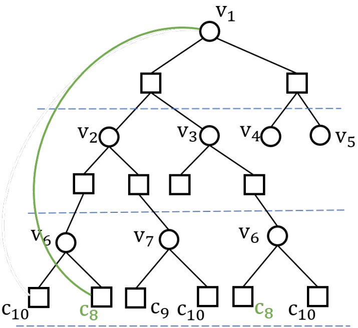
EC-PEG Algorithm: CN Selection Procedure



Candidate CNs : c_8, c_9, c_{10}

- For each CN candidate, calculate the resultant VN cycle counts
- $(\lambda_1^4, \ldots, \lambda_n^4), (\lambda_1^6, \ldots, \lambda_n^6), (\lambda_1^8, \ldots, \lambda_n^8)$

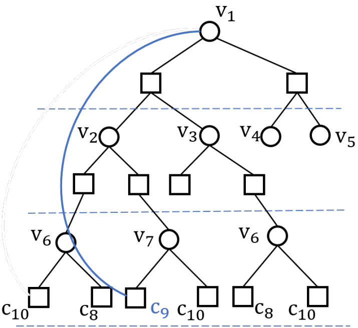
EC-PEG Algorithm: CN Selection Procedure



Candidate CNs : c_8, c_9, c_{10}

- For each CN candidate, calculate the resultant VN cycle counts
- $(\lambda_1^4, \ldots, \lambda_n^4), (\lambda_1^6, \ldots, \lambda_n^6), (\lambda_1^8, \ldots, \lambda_n^8)$
- $(\lambda_1^4, \ldots, \lambda_n^4), (\lambda_1^6, \ldots, \lambda_n^6), (\lambda_1^8, \ldots, \lambda_n^8)$

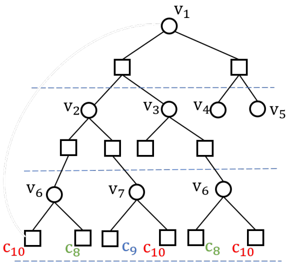
EC-PEG Algorithm: CN Selection Procedure



Candidate CNs : c_8, c_9, c_{10}

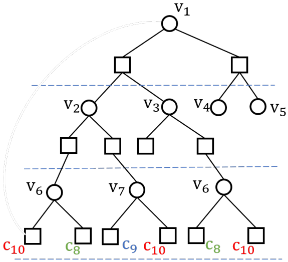
- For each CN candidate, calculate the resultant VN cycle counts
- $(\lambda_1^4, \ldots, \lambda_n^4), (\lambda_1^6, \ldots, \lambda_n^6), (\lambda_1^8, \ldots, \lambda_n^8)$
- $(\lambda_1^4, \ldots, \lambda_n^4), (\lambda_1^6, \ldots, \lambda_n^6), (\lambda_1^8, \ldots, \lambda_n^8)$
- $(\lambda_1^4, \ldots, \lambda_n^4), (\lambda_1^6, \ldots, \lambda_n^6), (\lambda_1^8, \ldots, \lambda_n^8)$

EC-PEG algorithm: CN selection Procedure



- $(\lambda_1^4, \ldots, \lambda_n^4), (\lambda_1^6, \ldots, \lambda_n^6), (\lambda_1^8, \ldots, \lambda_n^8)$
- $(\lambda_1^4, \ldots, \lambda_n^4), (\lambda_1^6, \ldots, \lambda_n^6), (\lambda_1^8, \ldots, \lambda_n^8)$
- $(\lambda_1^4, \ldots, \lambda_n^4), (\lambda_1^6, \ldots, \lambda_n^6), (\lambda_1^8, \ldots, \lambda_n^8)$

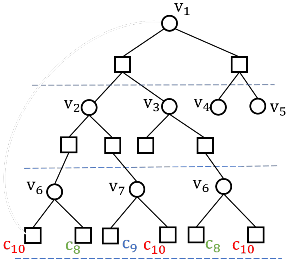
EC-PEG algorithm: CN selection Procedure



- $(\lambda_1^4, \ldots, \lambda_n^4), (\lambda_1^6, \ldots, \lambda_n^6), (\lambda_1^8, \ldots, \lambda_n^8)$
- $(\lambda_1^4, \ldots, \lambda_n^4), (\lambda_1^6, \ldots, \lambda_n^6), (\lambda_1^8, \ldots, \lambda_n^8)$
- $(\lambda_1^4, \ldots, \lambda_n^4), (\lambda_1^6, \ldots, \lambda_n^6), (\lambda_1^8, \ldots, \lambda_n^8)$

$$\underbrace{(\lambda_1^g, \ldots, \lambda_n^g)}_{\text{cycle counts}}$$

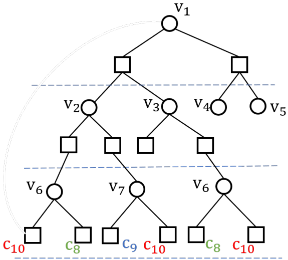
EC-PEG algorithm: CN selection Procedure



- $(\lambda_1^4, \dots, \lambda_n^4), (\lambda_1^6, \dots, \lambda_n^6), (\lambda_1^8, \dots, \lambda_n^8)$
- $(\lambda_1^4, \dots, \lambda_n^4), (\lambda_1^6, \dots, \lambda_n^6), (\lambda_1^8, \dots, \lambda_n^8)$
- $(\lambda_1^4, \dots, \lambda_n^4), (\lambda_1^6, \dots, \lambda_n^6), (\lambda_1^8, \dots, \lambda_n^8)$

$$\underbrace{(\lambda_1^g, \dots, \lambda_n^g)}_{\text{cycle counts}} \rightarrow \underbrace{\left(\frac{\lambda_1^g}{\sum_{i=1}^n \lambda_i^g}, \dots, \frac{\lambda_n^g}{\sum_{i=1}^n \lambda_i^g}\right) := \alpha^g}_{\text{normalized counts}}$$

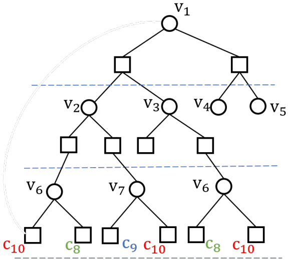
EC-PEG algorithm: CN selection Procedure



- $(\lambda_1^4, \ldots, \lambda_n^4), (\lambda_1^6, \ldots, \lambda_n^6), (\lambda_1^8, \ldots, \lambda_n^8)$
- $(\lambda_1^4, \ldots, \lambda_n^4), (\lambda_1^6, \ldots, \lambda_n^6), (\lambda_1^8, \ldots, \lambda_n^8)$
- $(\lambda_1^4, \ldots, \lambda_n^4), (\lambda_1^6, \ldots, \lambda_n^6), (\lambda_1^8, \ldots, \lambda_n^8)$

$$\underbrace{(\lambda_1^g, \ldots, \lambda_n^g)}_{\text{cycle counts}} \rightarrow \underbrace{\left(\frac{\lambda_1^g}{\sum_{i=1}^n \lambda_i^g}, \ldots, \frac{\lambda_n^g}{\sum_{i=1}^n \lambda_i^g}\right) := \alpha^g}_{\text{normalized counts}} \rightarrow \underbrace{\mathcal{H}\left(\frac{\alpha^4 + \alpha^6 + \alpha^8}{3}\right)}_{\text{entropy of combined counts}}$$

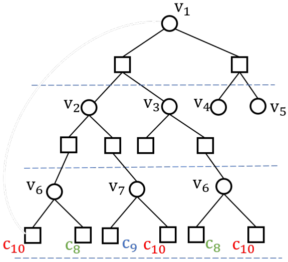
EC-PEG algorithm: CN selection Procedure



- $(\lambda_1^4, \ldots, \lambda_n^4), (\lambda_1^6, \ldots, \lambda_n^6), (\lambda_1^8, \ldots, \lambda_n^8) \rightarrow \mathcal{H}(\frac{\alpha^4+\alpha^6+\alpha^8}{3})$
- $(\lambda_1^4, \ldots, \lambda_n^4), (\lambda_1^6, \ldots, \lambda_n^6), (\lambda_1^8, \ldots, \lambda_n^8)$
- $(\lambda_1^4, \ldots, \lambda_n^4), (\lambda_1^6, \ldots, \lambda_n^6), (\lambda_1^8, \ldots, \lambda_n^8)$

$$\underbrace{(\lambda_1^g, \ldots, \lambda_n^g)}_{\text{cycle counts}} \rightarrow \underbrace{(\frac{\lambda_1^g}{\sum_{i=1}^n \lambda_i^g}, \ldots, \frac{\lambda_n^g}{\sum_{i=1}^n \lambda_i^g}) := \alpha^g}_{\text{normalized counts}} \rightarrow \underbrace{\mathcal{H}(\frac{\alpha^4+\alpha^6+\alpha^8}{3})}_{\text{entropy of combined counts}}$$

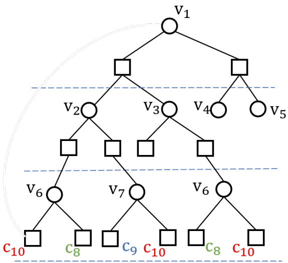
EC-PEG algorithm: CN selection Procedure



- $(\lambda_1^4, \ldots, \lambda_n^4), (\lambda_1^6, \ldots, \lambda_n^6), (\lambda_1^8, \ldots, \lambda_n^8) \rightarrow \mathcal{H}(\frac{\alpha^4+\alpha^6+\alpha^8}{3})$
- $(\lambda_1^4, \ldots, \lambda_n^4), (\lambda_1^6, \ldots, \lambda_n^6), (\lambda_1^8, \ldots, \lambda_n^8) \rightarrow \mathcal{H}(\frac{\alpha^4+\alpha^6+\alpha^8}{3})$
- $(\lambda_1^4, \ldots, \lambda_n^4), (\lambda_1^6, \ldots, \lambda_n^6), (\lambda_1^8, \ldots, \lambda_n^8)$

$$\underbrace{(\lambda_1^g, \ldots, \lambda_n^g)}_{\text{cycle counts}} \rightarrow \underbrace{\left(\frac{\lambda_1^g}{\sum_{i=1}^n \lambda_i^g}, \ldots, \frac{\lambda_n^g}{\sum_{i=1}^n \lambda_i^g}\right) := \alpha^g}_{\text{normalized counts}} \rightarrow \underbrace{\mathcal{H}\left(\frac{\alpha^4+\alpha^6+\alpha^8}{3}\right)}_{\text{entropy of combined counts}}$$

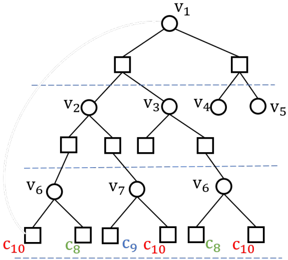
EC-PEG algorithm: CN selection Procedure



- $(\lambda_1^4, \dots, \lambda_n^4), (\lambda_1^6, \dots, \lambda_n^6), (\lambda_1^8, \dots, \lambda_n^8) \rightarrow \mathcal{H}(\frac{\alpha^4 + \alpha^6 + \alpha^8}{3})$
- $(\lambda_1^4, \dots, \lambda_n^4), (\lambda_1^6, \dots, \lambda_n^6), (\lambda_1^8, \dots, \lambda_n^8) \rightarrow \mathcal{H}(\frac{\alpha^4 + \alpha^6 + \alpha^8}{3})$
- $(\lambda_1^4, \dots, \lambda_n^4), (\lambda_1^6, \dots, \lambda_n^6), (\lambda_1^8, \dots, \lambda_n^8) \rightarrow \mathcal{H}(\frac{\alpha^4 + \alpha^6 + \alpha^8}{3})$

$$\underbrace{(\lambda_1^g, \dots, \lambda_n^g)}_{\text{cycle counts}} \rightarrow \underbrace{\left(\frac{\lambda_1^g}{\sum_{i=1}^n \lambda_i^g}, \dots, \frac{\lambda_n^g}{\sum_{i=1}^n \lambda_i^g}\right) := \alpha^g}_{\text{normalized counts}} \rightarrow \underbrace{\mathcal{H}\left(\frac{\alpha^4 + \alpha^6 + \alpha^8}{3}\right)}_{\text{entropy of combined counts}}$$

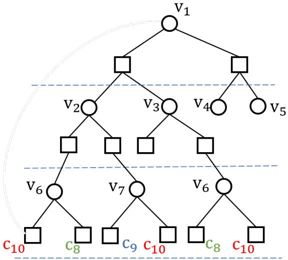
EC-PEG algorithm: CN selection Procedure



- $(\lambda_1^4, \ldots, \lambda_n^4), (\lambda_1^6, \ldots, \lambda_n^6), (\lambda_1^8, \ldots, \lambda_n^8) \rightarrow \mathcal{H}(\frac{\alpha^4+\alpha^6+\alpha^8}{3})$
- $(\lambda_1^4, \ldots, \lambda_n^4), (\lambda_1^6, \ldots, \lambda_n^6), (\lambda_1^8, \ldots, \lambda_n^8) \rightarrow \mathcal{H}(\frac{\alpha^4+\alpha^6+\alpha^8}{3})$
- $(\lambda_1^4, \ldots, \lambda_n^4), (\lambda_1^6, \ldots, \lambda_n^6), (\lambda_1^8, \ldots, \lambda_n^8) \rightarrow \mathcal{H}(\frac{\alpha^4+\alpha^6+\alpha^8}{3})$

CN selection procedure:

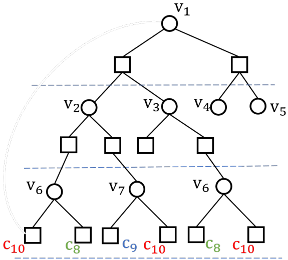
EC-PEG algorithm: CN selection Procedure



- $(\lambda_1^4, \ldots, \lambda_n^4), (\lambda_1^6, \ldots, \lambda_n^6), (\lambda_1^8, \ldots, \lambda_n^8) \rightarrow \mathcal{H}(\frac{\alpha^4+\alpha^6+\alpha^8}{3})$
- $(\lambda_1^4, \ldots, \lambda_n^4), (\lambda_1^6, \ldots, \lambda_n^6), (\lambda_1^8, \ldots, \lambda_n^8) \rightarrow \mathcal{H}(\frac{\alpha^4+\alpha^6+\alpha^8}{3})$
- $(\lambda_1^4, \ldots, \lambda_n^4), (\lambda_1^6, \ldots, \lambda_n^6), (\lambda_1^8, \ldots, \lambda_n^8) \rightarrow \mathcal{H}(\frac{\alpha^4+\alpha^6+\alpha^8}{3})$

CN selection procedure:
Select CN that results in minimum $\mathcal{H}(\frac{\alpha^4+\alpha^6+\alpha^8}{3})$

EC-PEG algorithm: CN selection Procedure



- $(\lambda_1^4, \ldots, \lambda_n^4), (\lambda_1^6, \ldots, \lambda_n^6), (\lambda_1^8, \ldots, \lambda_n^8) \rightarrow \mathcal{H}(\frac{\alpha^4+\alpha^6+\alpha^8}{3})$
- $(\lambda_1^4, \ldots, \lambda_n^4), (\lambda_1^6, \ldots, \lambda_n^6), (\lambda_1^8, \ldots, \lambda_n^8) \rightarrow \mathcal{H}(\frac{\alpha^4+\alpha^6+\alpha^8}{3})$
- $(\lambda_1^4, \ldots, \lambda_n^4), (\lambda_1^6, \ldots, \lambda_n^6), (\lambda_1^8, \ldots, \lambda_n^8) \rightarrow \mathcal{H}(\frac{\alpha^4+\alpha^6+\alpha^8}{3})$

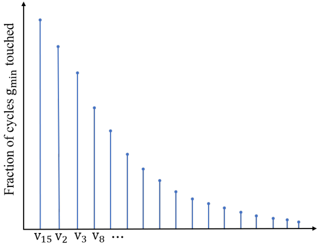
CN selection procedure:
Select CN that results in minimum $\mathcal{H}(\frac{\alpha^4+\alpha^6+\alpha^8}{3})$

Note:

- Minimizing the entropy of joint cycle counts ensures that all cycle distributions are concentrated towards the same set of VNs

Sampling Strategy

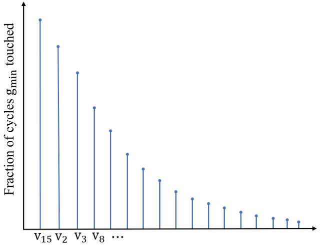
- Our sampling strategy greedily samples VNs that are part of a large number of cycles



$g =$ smallest cycle length in Tanner Graph $\mathcal{G}$
While sample set size $< s$
- $v =$ VN that is part of largest no. of cycles of length g in $\mathcal{G}$
- sample set $=$ sample set $\cup$ v
- remove v and all incident edges from $\mathcal{G}$

Sampling Strategy

- Our sampling strategy greedily samples VNs that are part of a large number of cycles



$g =$ smallest cycle length in Tanner Graph $\mathcal{G}$
While sample set size $< s$
- $v =$ VN that is part of largest no. of cycles of length g in $\mathcal{G}$
- sample set $=$ sample set $\cup$ v
- remove v and all incident edges from $\mathcal{G}$

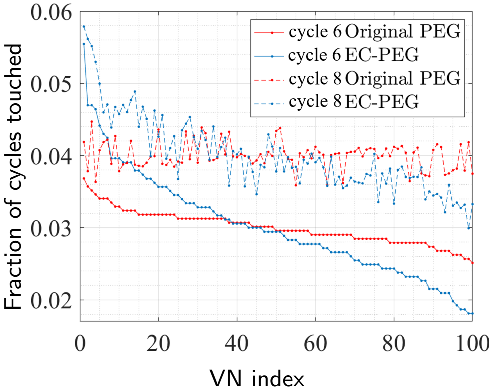
If $\nexists$ cycles of length g in $\mathcal{G}$
- $g = g + 2$

Simulation Results

- Code parameters: Code length $= 100$, VN degree $= 4$, Rate $= \frac{1}{2}$, girth $= 6$.

Simulation Results

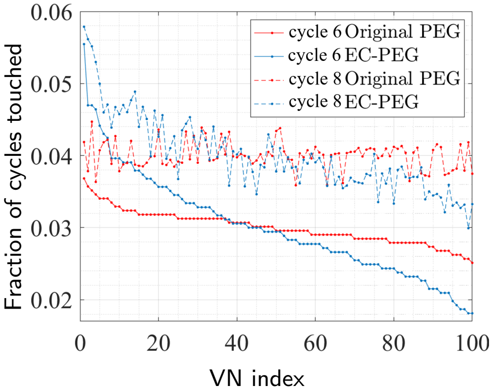
- Code parameters: Code length $= 100$, VN degree $= 4$, Rate $= \frac{1}{2}$, girth $= 6$.



- VN indices arranged in decreasing order of cycle 6 fractions

Simulation Results

- Code parameters: Code length $= 100$, VN degree $= 4$, Rate $= \frac{1}{2}$, girth $= 6$.



- VN indices arranged in decreasing order of cycle 6 fractions
- Cycle 6 and cycle 8 concentrated towards same set of VNs

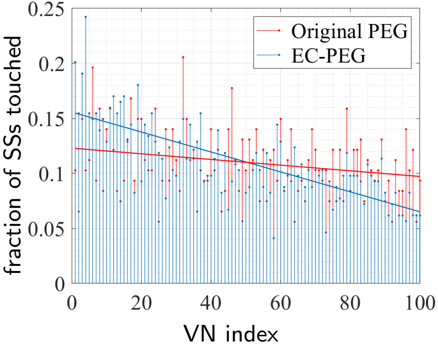
Simulation Results

Fraction of SSs of size 11, 12 touched by different VNs

Simulation Results

Fraction of SSs of size 11, 12 touched by different VNs

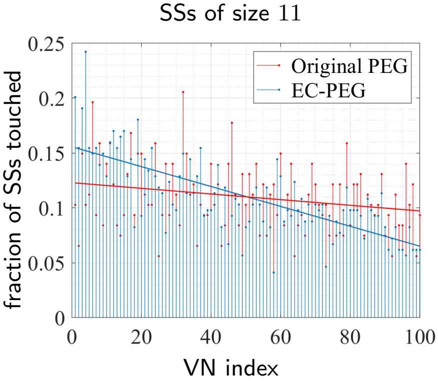
SSs of size 11

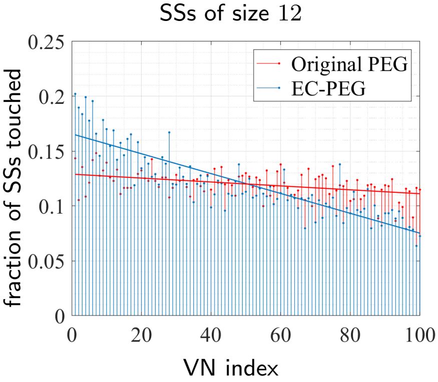


- VN indices arranged in decreasing order of cycle 6 fractions

Simulation Results

Fraction of SSs of size 11, 12 touched by different VNs

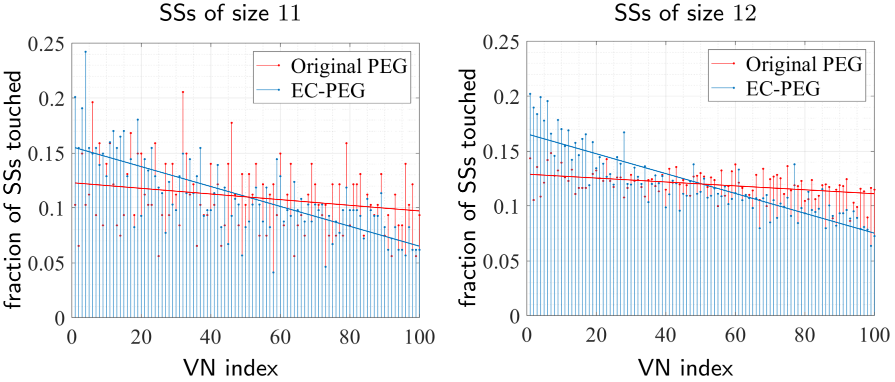




- VN indices arranged in decreasing order of cycle 6 fractions

Simulation Results

Fraction of SSs of size 11, 12 touched by different VNs



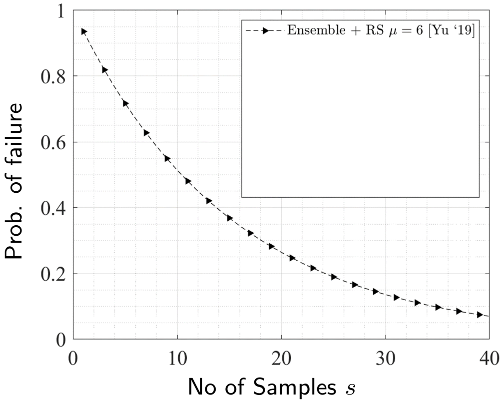
- VN indices arranged in decreasing order of cycle 6 fractions
- SSs are concentrated towards the same set of VNs as the cycles

Simulation Results

Probability of failure for a stopping set of size μ

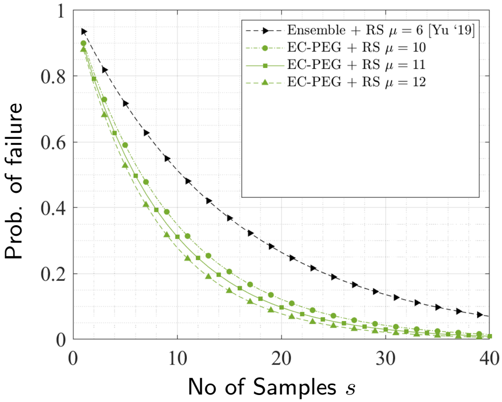
Simulation Results

Probability of failure for a stopping set of size μ RS: Random Sampling



Simulation Results

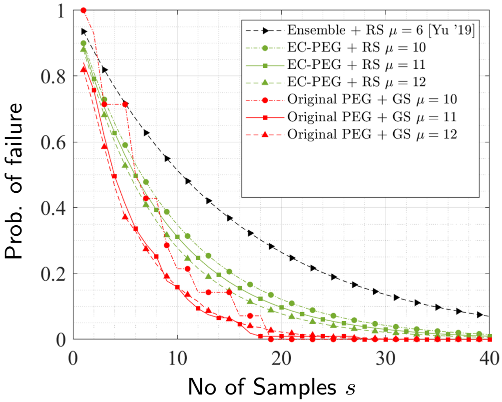
Probability of failure for a stopping set of size μ RS: Random Sampling



Simulation Results

Probability of failure for a stopping set of size μ

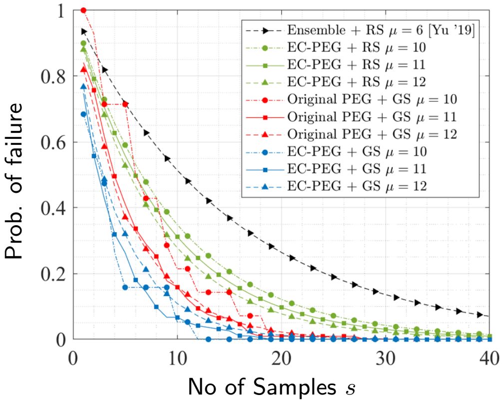
RS: Random Sampling
GS: Greedy Sampling



Simulation Results

Probability of failure for a stopping set of size μ

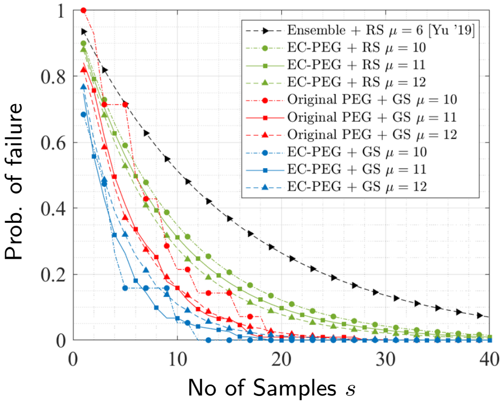
RS: Random Sampling
GS: Greedy Sampling



Simulation Results

Probability of failure for a stopping set of size μ

RS: Random Sampling
GS: Greedy Sampling



- Concentrated LDPC codes with Greedy sampling improve the probability of failure

Incorrect Coding Proof Size

- Depends on the maximum check node degree

Rate	Code length	VN degree	Ensemble [Yu '19]	PEG	EC-PEG
$\frac{1}{2}$	100	4	16	9	11
	200	4	16	9	15
$\frac{1}{4}$	100	4	8	7	10
	200	4	8	6	9

Table: Maximum CN degree for different codes.

Incorrect Coding Proof Size

- Depends on the maximum check node degree

Rate	Code length	VN degree	Ensemble [Yu '19]	PEG	EC-PEG
$\frac{1}{2}$	100	4	16	9	11
	200	4	16	9	15
$\frac{1}{4}$	100	4	8	7	10
	200	4	8	6	9

Table: Maximum CN degree for different codes.

- Concentrated LDPC codes do not sacrifice on the incorrect coding proof size

Conclusion and Ongoing Work

- Summary:
  - We provided a specialized code construction technique to concentrate stopping sets in LDPC codes

Conclusion and Ongoing Work

- Summary:
  - We provided a specialized code construction technique to concentrate stopping sets in LDPC codes
  - Coupled with a greedy sampling strategy, concentrated LDPC codes reduce the probability of light node failure compared to earlier approaches

Conclusion and Ongoing Work

- Summary:
  - We provided a specialized code construction technique to concentrate stopping sets in LDPC codes
  - Coupled with a greedy sampling strategy, concentrated LDPC codes reduce the probability of light node failure compared to earlier approaches
- Ongoing work:
  - Improving security against stronger adversaries that can selectively pick a stopping set that has a lower probability of being sampled to hide

References

- (Al-Bassam '18) M. Al-Bassam, et al., *"Fraud and Data Availability Proofs: Maximising Light Client Security and Scaling Blockchains with Dishonest Majorities,"* arXiv preprint arXiv:1809.09044, 2018.
- (Yu '19) M. Yu, et al., *"Coded Merkle Tree: Solving Data Availability Attacks in Blockchains,"* International Conference on Financial Cryptography and Data Security, Springer, Cham, 2020.
- (Xiao '05) X.Y. Hu, et al., *"Regular and irregular progressive edge-growth tanner graphs,"* IEEE Transactions of Information Theory, vol. 51, no. 1, 2005.
- (Tian '03) T. Tian, et al., *"Construction of irregular LDPC codes with low error floors,"* IEEE International Conference on Communications, May 2003.