From Stopping sets to Trapping sets

The Exhaustive Search Algorithm & The Suppressing Effect

Chih-Chun Wang

School of Electrical & Computer Engineering
Purdue University

Content

- Good exhaustive trapping set search algorithm for arbitrary codes.
- The suppressing effect for cyclically lifted code ensembles.

Content

- Good exhaustive trapping set search algorithm for arbitrary codes.
  - New results on the hardness of the problem

- The suppressing effect for cyclically lifted code ensembles.

Content

- Good exhaustive trapping set search algorithm for arbitrary codes.
  - New results on the hardness of the problem
  - Existing work on exhaustive search for stopping sets
- The suppressing effect for cyclically lifted code ensembles.

Content

- Good exhaustive trapping set search algorithm for arbitrary codes.
  - New results on the hardness of the problem
  - Existing work on exhaustive search for stopping sets
  - The exhaustive search for trapping sets based on exhaustive search for stopping sets.
- The suppressing effect for cyclically lifted code ensembles.

Content

- Good exhaustive trapping set search algorithm for arbitrary codes.
  - New results on the hardness of the problem
  - Existing work on exhaustive search for stopping sets
  - The exhaustive search for trapping sets based on exhaustive search for stopping sets.
  - Lessons from the results of exhaustive search algorithms
- The suppressing effect for cyclically lifted code ensembles.

Content

- Good exhaustive trapping set search algorithm for arbitrary codes.
  - New results on the hardness of the problem
  - Existing work on exhaustive search for stopping sets
  - The exhaustive search for trapping sets based on exhaustive search for stopping sets.
  - Lessons from the results of exhaustive search algorithms
- The suppressing effect for cyclically lifted code ensembles.
  - Definition: $\text{Prob}(\text{the bad structure remains after lifting})$

Content

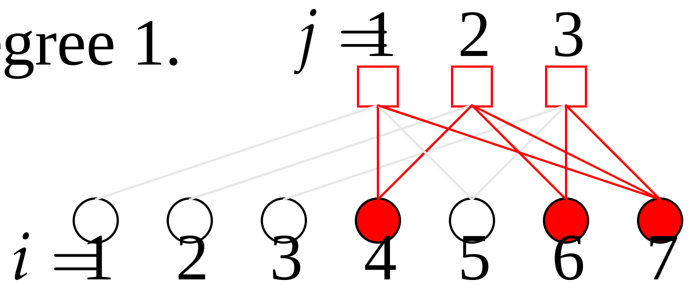
- Good exhaustive trapping set search algorithm for arbitrary codes.
  - New results on the hardness of the problem
  - Existing work on exhaustive search for stopping sets
  - The exhaustive search for trapping sets based on exhaustive search for stopping sets.
  - Lessons from the results of exhaustive search algorithms
- The suppressing effect for cyclically lifted code ensembles.
  - Definition: $\text{Prob}(\text{the bad structure remains after lifting})$
  - Quantifying the suppressing effect.

Content

- Good exhaustive trapping set search algorithm for arbitrary codes.
  - New results on the hardness of the problem
  - Existing work on exhaustive search for stopping sets
  - The exhaustive search for trapping sets based on exhaustive search for stopping sets.
  - Lessons from the results of exhaustive search algorithms
- The suppressing effect for cyclically lifted code ensembles.
  - Definition: $\text{Prob}(\text{the bad structure remains after lifting})$
  - Quantifying the suppressing effect.
  - A design criteria for base code optimization.

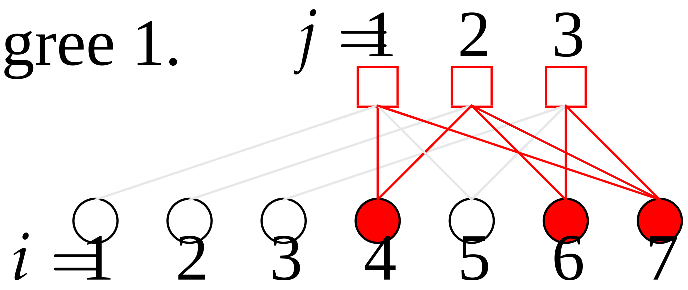
Stopping Sets

- Definition: a set of variable nodes ⇒ the induced graph contains no check node of degree 1.



Stopping Sets

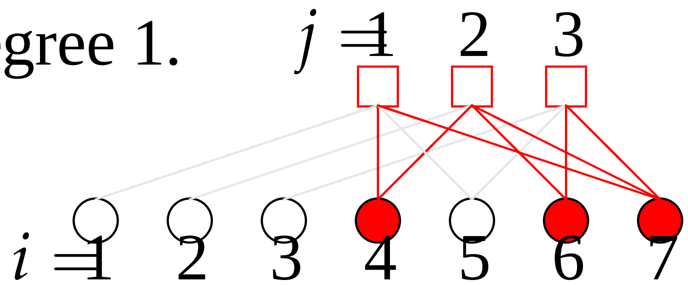
- Definition: a set of variable nodes ⇒ the induced graph contains no check node of degree 1.



- Why exhaustive search algorithms (for small stopping sets)?

Stopping Sets

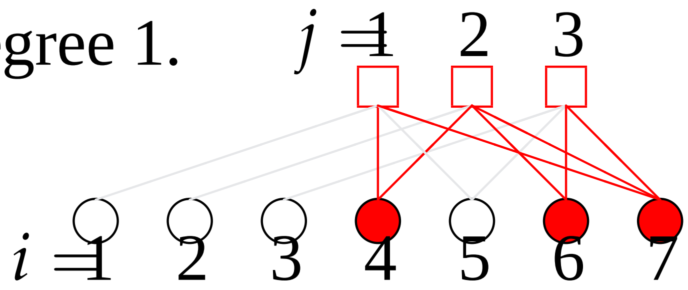
- Definition: a set of variable nodes $\Rightarrow$ the induced graph contains no check node of degree 1.



- Why exhaustive search algorithms (for small stopping sets)?
  - Error floor optimization. BECs vs. non-erasure channels.

Stopping Sets

- Definition: a set of variable nodes $\Rightarrow$ the induced graph contains no check node of degree 1.



- Why exhaustive search algorithms (for small stopping sets)?
  - Error floor optimization. BECs vs. non-erasure channels.
- Good but inexhaustive search algorithms: error floors of LDPC codes [Richardson 03], projection algebra [Yedidia *et al.* 01], the approximate minimum distance of LDPC codes [Hu *et al.* 04], [Hirotomo *et al.* 05], [Richter 06]

An NP-Hard Problem

=== The $\mathrm{SD}(H, t)$ problem ===

INPUT: A code represented by its parity-check matrix H and an integer t.

OUTPUT: Output 1 if the minimal stopping distance of H is $\leq t$. Otherwise, output 0.

The hardness results:

- [Krishnan *et al.* 06]: For arbitrary H, $\mathrm{SD}(H, t)$ is NP-complete. Proof: By reducing a VERTEX-COVER problem to $\mathrm{SD}(H, t)$.
- A byproduct of [Krishnan *et al.* 06]: With the sparsity restriction that the number of 1's in H is limited to $O(n)$ rather than $O(n^2)$, then $\mathrm{SD}(H, t)$ is still NP-complete.

Trapping Sets: Definitions

- **Operational definition:** “the set of bits that are not eventually correct" [Richardson 03].

Trapping Sets: Definitions

- **Operational definition:** “the set of bits that are not eventually correct" [Richardson 03].
- **Empirical observations:** For non-erasure channels: trapping sets are (a, b) near-codeword [MacKay *et al.* 03]

Trapping Sets: Definitions

- **Operational definition:** “the set of bits that are not eventually correct" [Richardson 03].
- **Empirical observations:** For non-erasure channels: trapping sets are (a, b) near-codeword [MacKay *et al.* 03]
  - (a, b) near codeword: A set of a variable nodes such the induced graph has b odd-degree check nodes.

Trapping Sets: Definitions

- **Operational definition:** "the set of bits that are not eventually correct" [Richardson 03].
- **Empirical observations:** For non-erasure channels: trapping sets are (a, b) near-codeword [MacKay *et al.* 03]
  - (a, b) near codeword: A set of a variable nodes such the induced graph has b odd-degree check nodes.
  - A $(a, 0)$ near codeword $\overset{\not\Leftarrow}{\Rightarrow}$ a stopping set

Trapping Sets: Definitions

- **Operational definition:** "the set of bits that are not eventually correct" [Richardson 03].
- **Empirical observations:** For non-erasure channels: trapping sets are (a, b) near-codeword [MacKay *et al.* 03]
  - (a, b) near codeword: A set of a variable nodes such the induced graph has b odd-degree check nodes.
  - A $(a, 0)$ near codeword $\overset{\not\Leftarrow}{\Rightarrow}$ a stopping set
- We propose a new graph-theoretic definition:

  Definition 1 (k-out Trapping Sets) *A subset of* $\{v_1, \ldots, v_n\}$ *such that in the induced subgraph, there are exactly k check nodes of degree one.*

k-Out Trapping Sets vs. Near-Codeword

Definition 1 (k-out Trapping Sets) *A subset of variables such that in the induced subgraph, there are exactly k check nodes of degree one.*

- k-out trapping sets $\longleftrightarrow$ stopping sets
  (a, b) near-codewords $\longleftrightarrow$ valid codewords
- 0-out trapping sets $\Longleftrightarrow$ stopping sets
  $(a, 0)$ near-codewords $\Longleftrightarrow$ valid codewords

k-Out Trapping Sets vs. Near-Codeword

Definition 1 (k-out Trapping Sets) *A subset of variables such that in the induced subgraph, there are exactly k check nodes of degree one.*

- k-out trapping sets $\longleftrightarrow$ stopping sets
  (a, b) near-codewords $\longleftrightarrow$ valid codewords
- 0-out trapping sets $\Longleftrightarrow$ stopping sets
  $(a, 0)$ near-codewords $\Longleftrightarrow$ valid codewords

Why this definition?

k-Out Trapping Sets vs. Near-Codeword

Definition 1 (k-out Trapping Sets) *A subset of variables such that in the induced subgraph, there are exactly k check nodes of degree one.*

- k-out trapping sets $\longleftrightarrow$ stopping sets
  (a, b) near-codewords $\longleftrightarrow$ valid codewords
- 0-out trapping sets $\Longleftrightarrow$ stopping sets
  $(a, 0)$ near-codewords $\Longleftrightarrow$ valid codewords

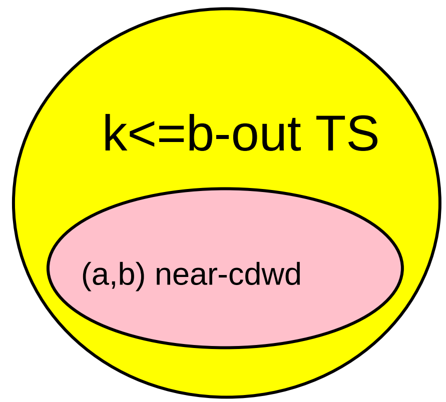
Why this definition?

- Better analogy to stopping sets.

k-Out Trapping Sets vs. Near-Codeword

Definition 1 (**k-out Trapping Sets** **)** *A subset of variables such that in the induced subgraph, there are exactly k check nodes of degree one.*

- k-out trapping sets $\longleftrightarrow$ stopping sets
  (a, b) near-codewords $\longleftrightarrow$ valid codewords
- 0-out trapping sets $\Longleftrightarrow$ stopping sets
  $(a, 0)$ near-codewords $\Longleftrightarrow$ valid codewords



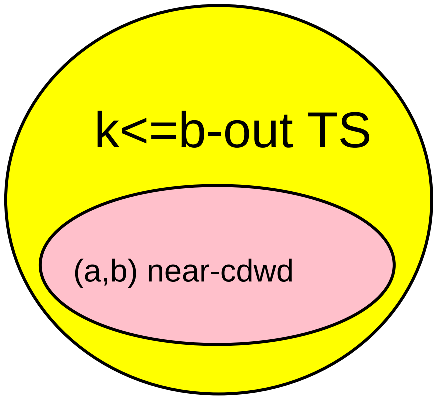
Why this definition?

- Better analogy to stopping sets.
- An (a, b) near-codeword $\stackrel{\nLeftarrow}{\Rightarrow}$ "$k \leq b$"-out trapping set.
- Our goal: With fixed b, search all min. $k \leq b$-out TSs.

k-Out Trapping Sets vs. Near-Codeword

Definition 1 (k-out Trapping Sets) *A subset of variables such that in the induced subgraph, there are exactly k check nodes of degree one.*

- k-out trapping sets $\longleftrightarrow$ stopping sets
  (a,b) near-codewords $\longleftrightarrow$ valid codewords
- 0-out trapping sets $\Longleftrightarrow$ stopping sets
  $(a,0)$ near-codewords $\Longleftrightarrow$ valid codewords



Why this definition?

- Better analogy to stopping sets.
- An (a,b) near-codeword $\substack{\not\Leftarrow \\ \Rightarrow}$ "$k \leq b$"-out trapping set.
- Our goal: With fixed b, search all min. $k \leq b$-out TSs.
- Empirically, all error bits consist of only degree 1 & 2 check nodes. (The elementary trapping set [Landner *et al.* 05].)

The Hardness of k-OTD(H, t)

=== The k-OTD(H, t) problem ===

INPUT: A code represented by its parity-check matrix H and an integer t.

OUTPUT: Output 1 if the minimal k-out trapping distance of H is $\leq t$. Otherwise, output 0.

The Hardness of k-OTD(H, t)

=== The k-OTD(H, t) problem ===

INPUT: A code represented by its parity-check matrix H and an integer t.

OUTPUT: Output 1 if the minimal k-out trapping distance of H is $\leq t$. Otherwise, output 0.

- When $k = 0$, then 0-OTD(H, t) = SD(H, t) is NP-complete.

The Hardness of k-OTD(H, t)

=== The k-OTD(H, t) problem ===

INPUT: A code represented by its parity-check matrix H and an integer t.

OUTPUT: Output 1 if the minimal k-out trapping distance of H is $\leq t$. Otherwise, output 0.

- When $k = 0$, then 0-OTD(H, t) = SD(H, t) is NP-complete.
- Is the hardness the same for any fixed $k > 0$ values?

Our First Result

Theorem 1 *Consider a fixed $k > 0$. For arbitrary H, k-OTD(H, t) is NP-complete.*

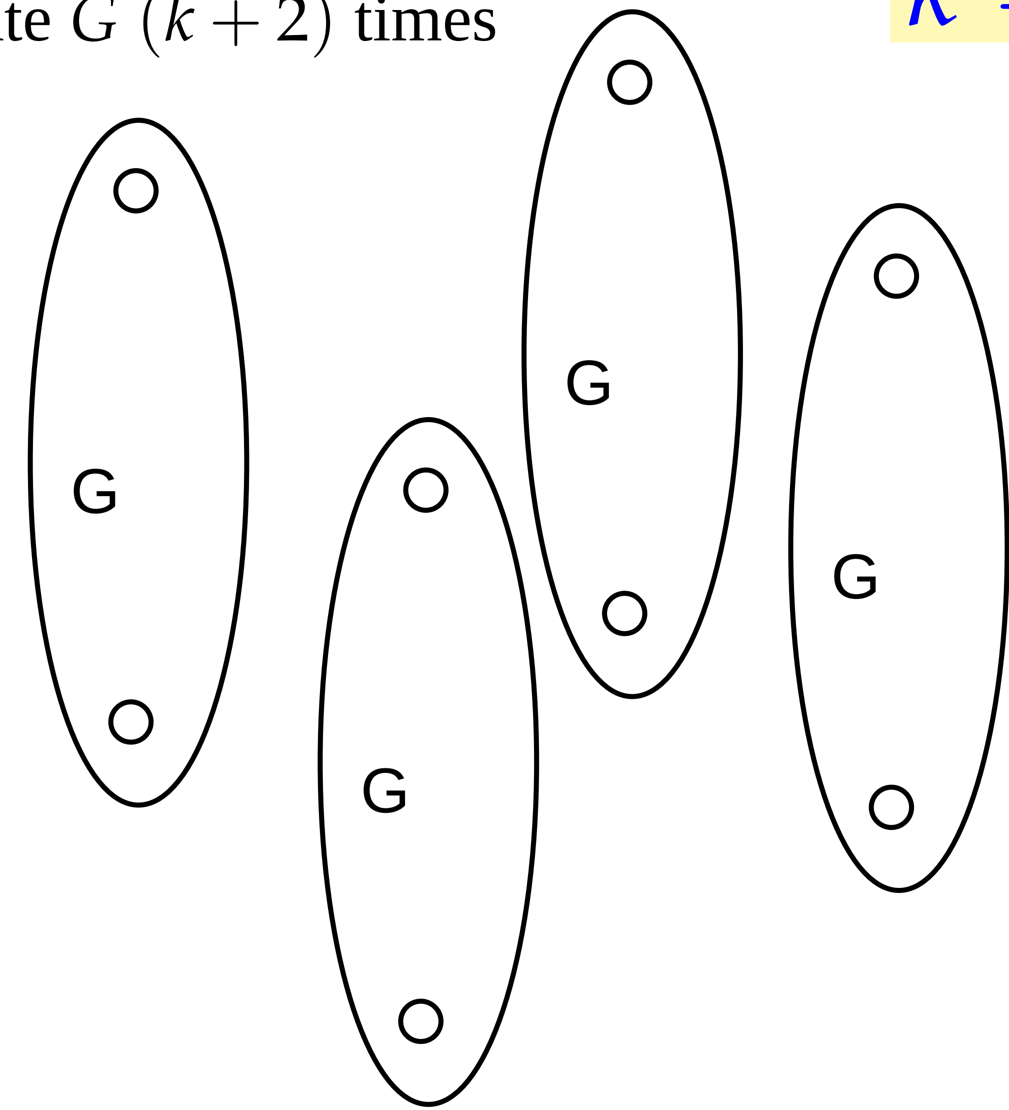
Theorem 2 *Consider a fixed $k > 0$. With the sparsity restriction that the number of 1's in H is limited to $O(n)$ rather than $O(n^2)$, then k-OTD(H, t) is still NP-complete.*

Proof: Reduction from SD(H, t).

SD(H,t) By k-OTD(H',t')

$k = 2$

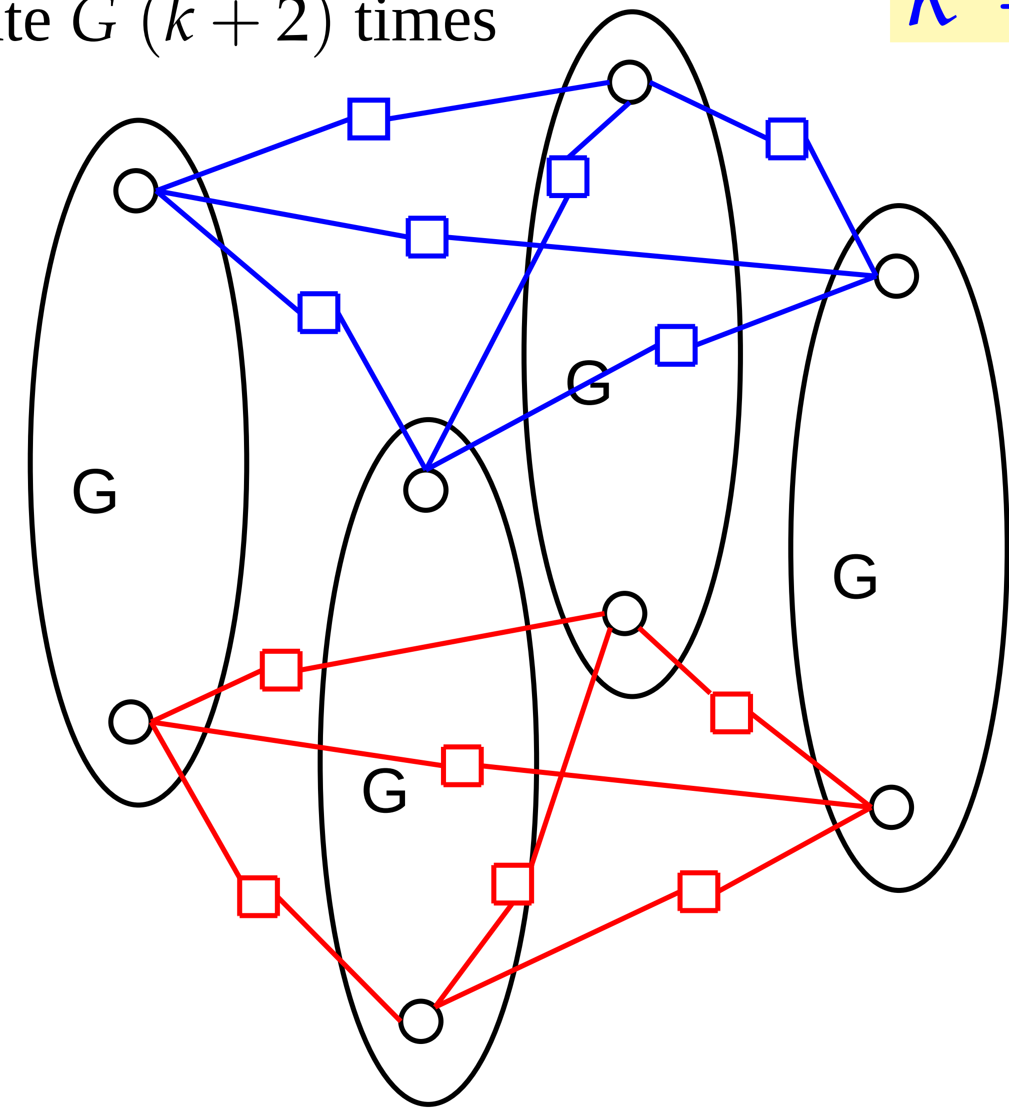
Step 1: Duplicate G $(k+2)$ times



SD(H, t) By k-OTD(H', t')

$k = 2$

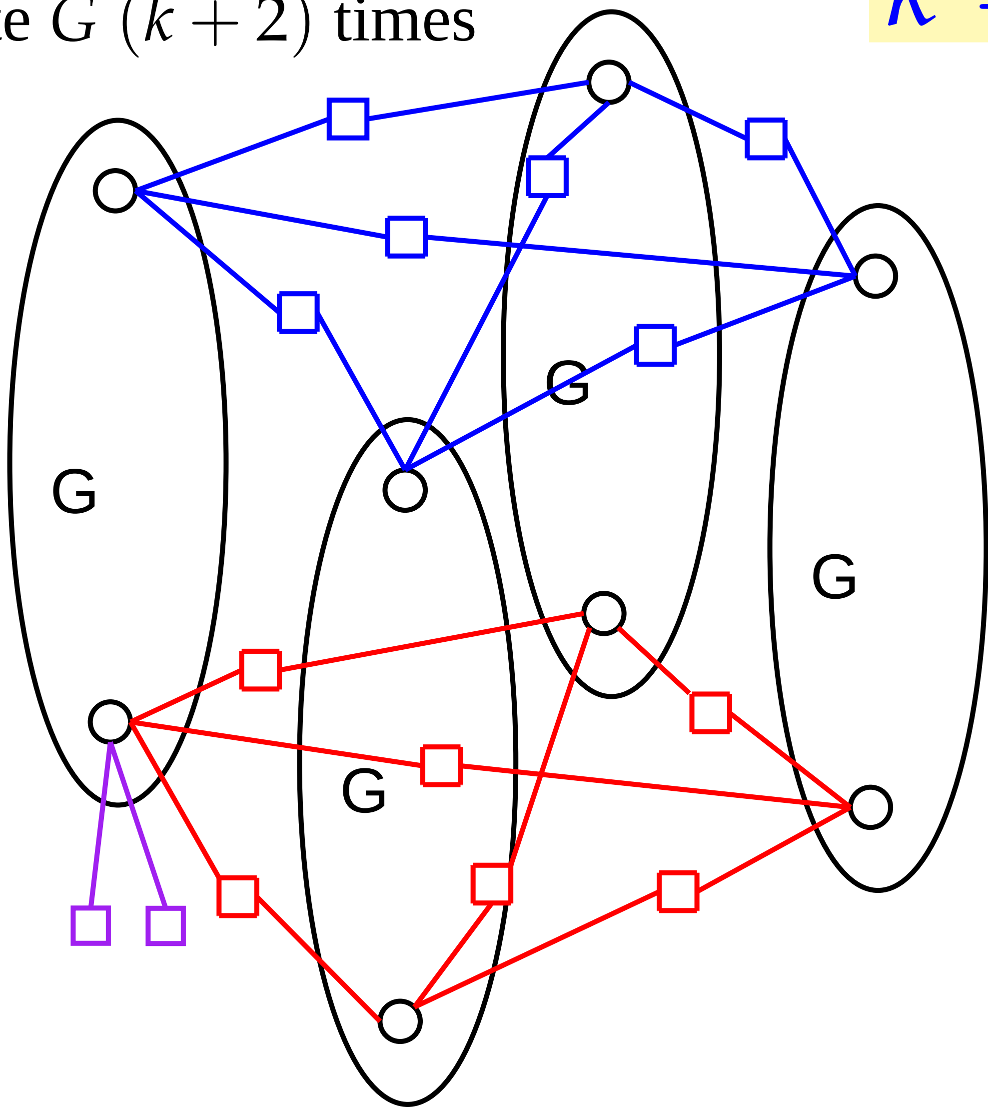
Step 1: Duplicate G $(k+2)$ times



SD(H, t) By k-OTD(H', t')

$k = 2$

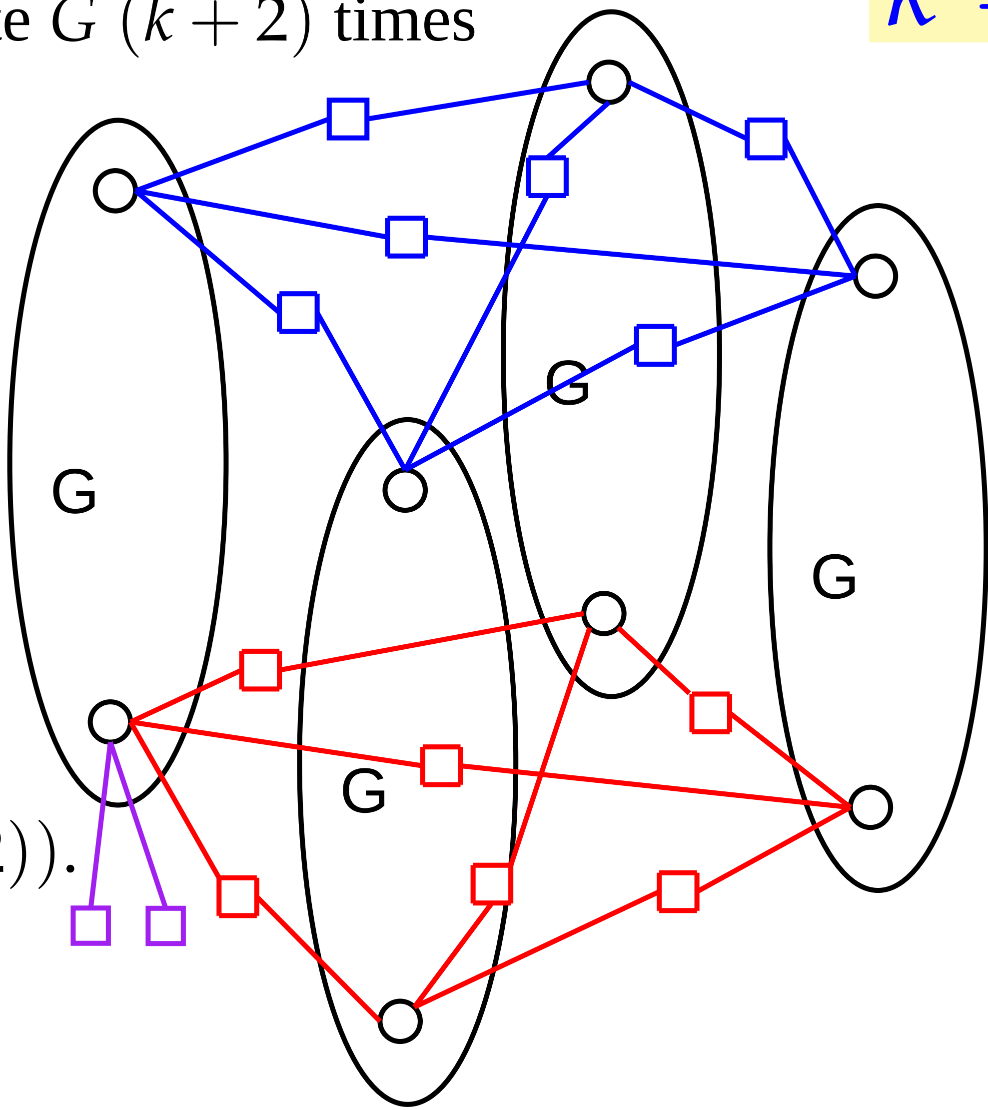
Step 1: Duplicate G $(k+2)$ times



SD(H,t) By k-OTD(H',t')

$k=2$

Step 1: Duplicate G $(k+2)$ times

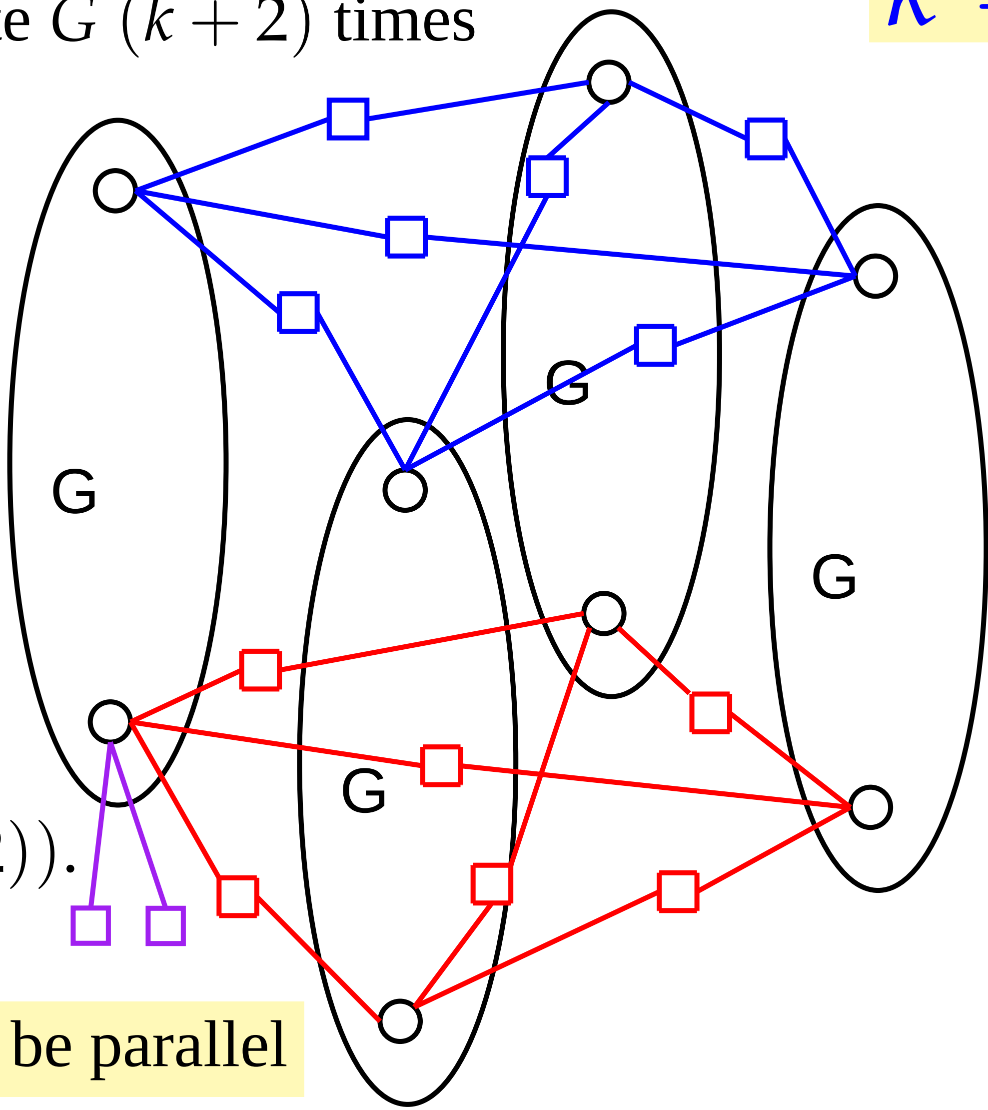


Run
k-OTD$(H', t(k+2))$.

SD(H,t) By k-OTD(H',t')

$k=2$

Step 1: Duplicate G $(k+2)$ times



Run k-OTD$(H',t(k+2))$.

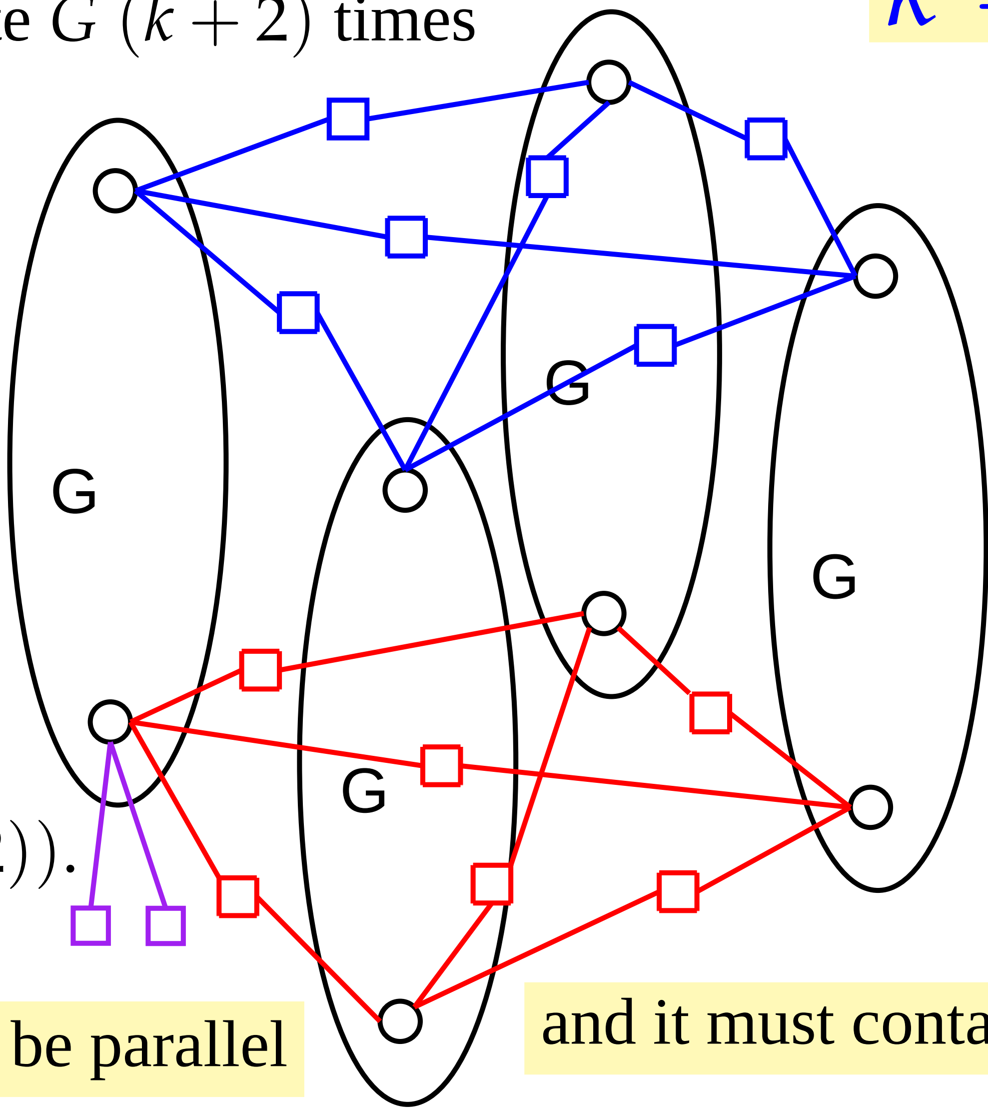
Claim:

any k-out TS must be parallel

SD(H,t) By k-OTD(H',t')

$k=2$

Step 1: Duplicate G $(k+2)$ times



Run
k-OTD$(H',t(k+2))$.

Claim:

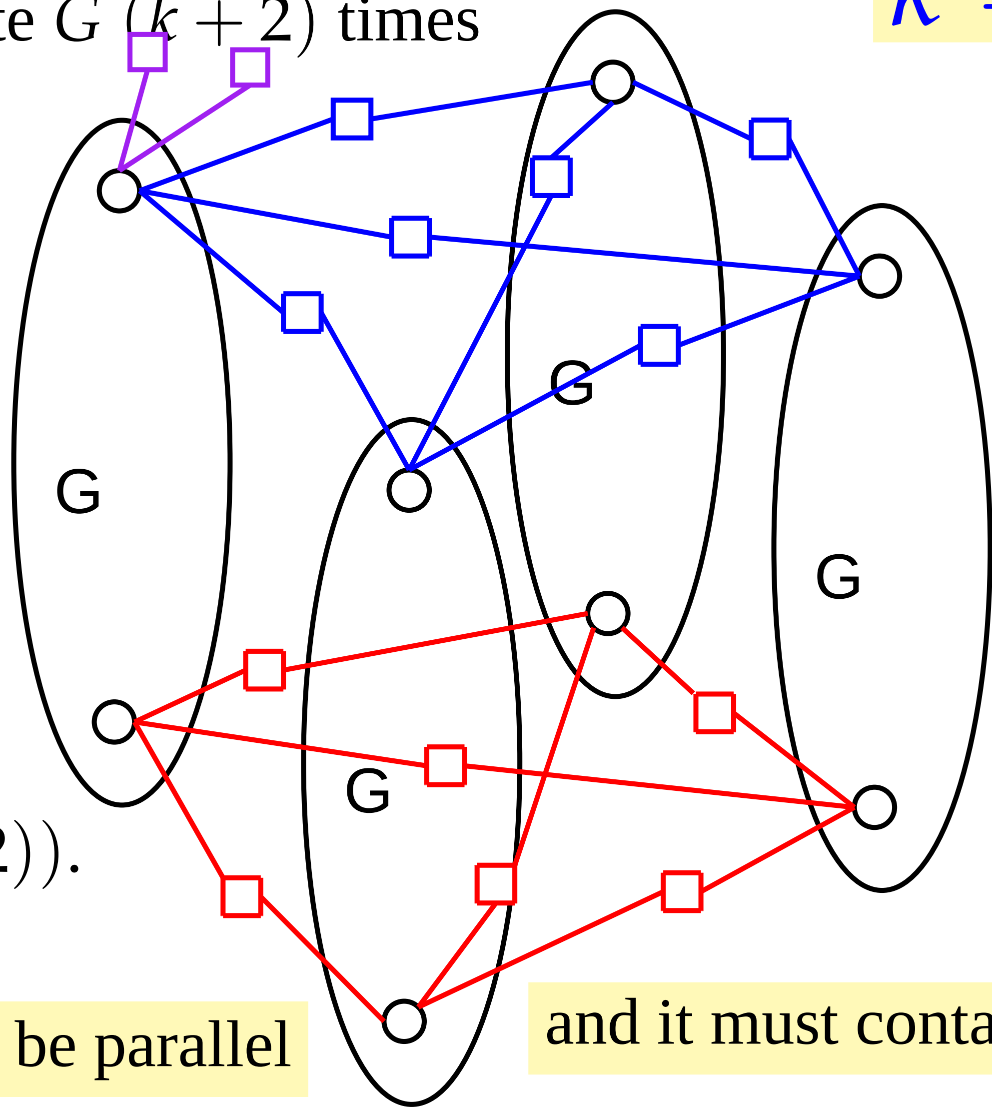
any k-out TS must be parallel

and it must contain the target bit.

SD(H,t) By k-OTD(H',t')

$k = 2$

Step 1: Duplicate G $(k+2)$ times



Run
k-OTD$(H', t(k+2))$.

Claim:

any k-out TS must be parallel and it must contain the target bit.

NP-hard problem = Impossible?

NP-hard problem = Impossible?

- Most approaches use heuristics instead for error-floor optimization.
  - The girth, the Approximate Cycle Extrinsic (ACE) message degree, partial stopping set elimination, and ensemble-inspired upper bounds.

NP-hard problem = Impossible?

- Most approaches use heuristics instead for error-floor optimization.
  - The girth, the Approximate Cycle Extrinsic (ACE) message degree, partial stopping set elimination, and ensemble-inspired upper bounds.

Is there anything else we can do?

NP-hard problem = Impossible?

- Most approaches use heuristics instead for error-floor optimization.
  - The girth, the Approximate Cycle Extrinsic (ACE) message degree, partial stopping set elimination, and ensemble-inspired upper bounds.

Is there anything else we can do?

- NP-completeness $\Longrightarrow$ the asymptotic complexity.

NP-hard problem = Impossible?

- Most approaches use heuristics instead for error-floor optimization.
  - The girth, the Approximate Cycle Extrinsic (ACE) message degree, partial stopping set elimination, and ensemble-inspired upper bounds.

Is there anything else we can do?

- NP-completeness $\Longrightarrow$ the asymptotic complexity.
- NP-completeness has relatively less predictability for finite n.

NP-hard problem = Impossible?

- Most approaches use heuristics instead for error-floor optimization.
  - The girth, the Approximate Cycle Extrinsic (ACE) message degree, partial stopping set elimination, and ensemble-inspired upper bounds.

Is there anything else we can do?

- NP-completeness $\Longrightarrow$ the asymptotic complexity.
- NP-completeness has relatively less predictability for finite n.
- For practical codes, we only need $n \approx 500$–5000.

NP-hard problem = Impossible?

- Most approaches use heuristics instead for error-floor optimization.
  - The girth, the Approximate Cycle Extrinsic (ACE) message degree, partial stopping set elimination, and ensemble-inspired upper bounds.

Is there anything else we can do?

- NP-completeness $\Longrightarrow$ the asymptotic complexity.
- NP-completeness has relatively less predictability for finite n.
- For practical codes, we only need $n \approx 500$–5000.
- An encouraging example: The travelling salesman problem. Optimal solution for 24,978 cities in Sweden is found in 2004.

Leverage Upon SD(H, t)

=== The SD(H, t) problem ===

OUTPUT: Output an exhaustive list of minimum stopping sets if the minimal stopping distance is $\leq t$. Otherwise, output $\emptyset$.

- In our previous work [ISIT 06], a good exhaustive search SD(H, t) is provided.
  - Capable of exhausting $t = 11$–13 for codes of $n \approx 500$.

Leverage Upon SD(H, t)

=== The SD(H, t) problem ===

OUTPUT: Output an exhaustive list of minimum stopping sets if the minimal stopping distance is $\leq t$. Otherwise, output $\emptyset$.

- In our previous work [ISIT 06], a good exhaustive search SD(H, t) is provided.
  - Capable of exhausting $t = 11$–13 for codes of $n \approx 500$.
- On this Friday 4:45pm [Rosnes & Ytrehus, ISIT07], a more efficient exhaustive search SD(H, t) will be introduced.
  - Capable of exhausting $t = 18$–26 for codes of $n = 150$–5000.

Leverage Upon SD(H, t)

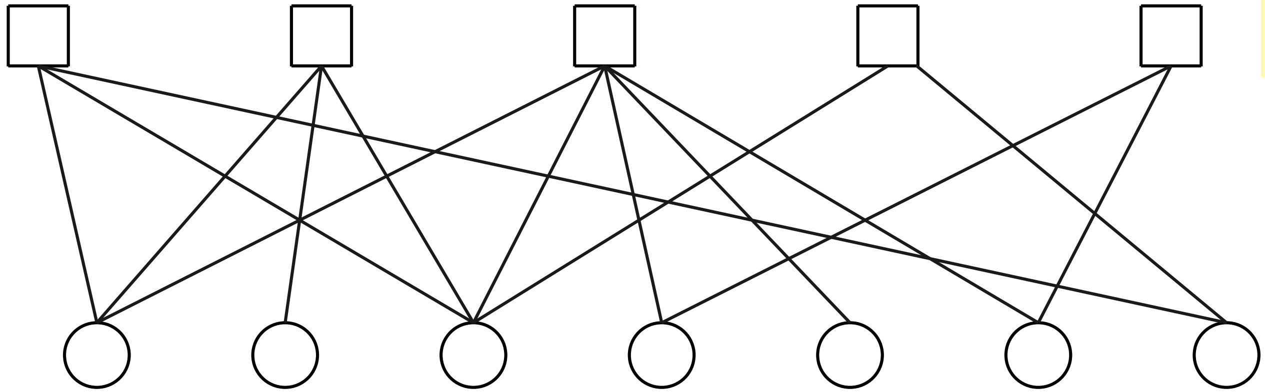
=== The SD(H, t) problem ===

OUTPUT: Output an exhaustive list of minimum stopping sets if the minimal stopping distance is $\leq t$. Otherwise, output $\emptyset$.

- In our previous work [ISIT 06], a good exhaustive search SD(H, t) is provided.
  - Capable of exhausting $t = 11$–13 for codes of $n \approx 500$.
- On this Friday 4:45pm [Rosnes & Ytrehus, ISIT07], a more efficient exhaustive search SD(H, t) will be introduced.
  - Capable of exhausting $t = 18$–26 for codes of $n = 150$–5000.
- Good SD$(H, t) \overset{?}{\Rightarrow}$ good k-OTD(H, t)

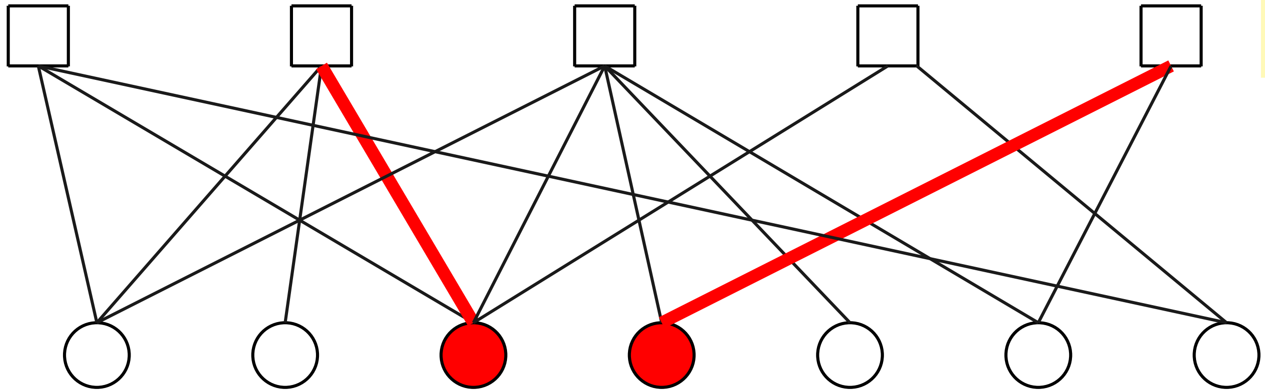
k-OTD(H, t') By SD(H, t)

$k = 2$



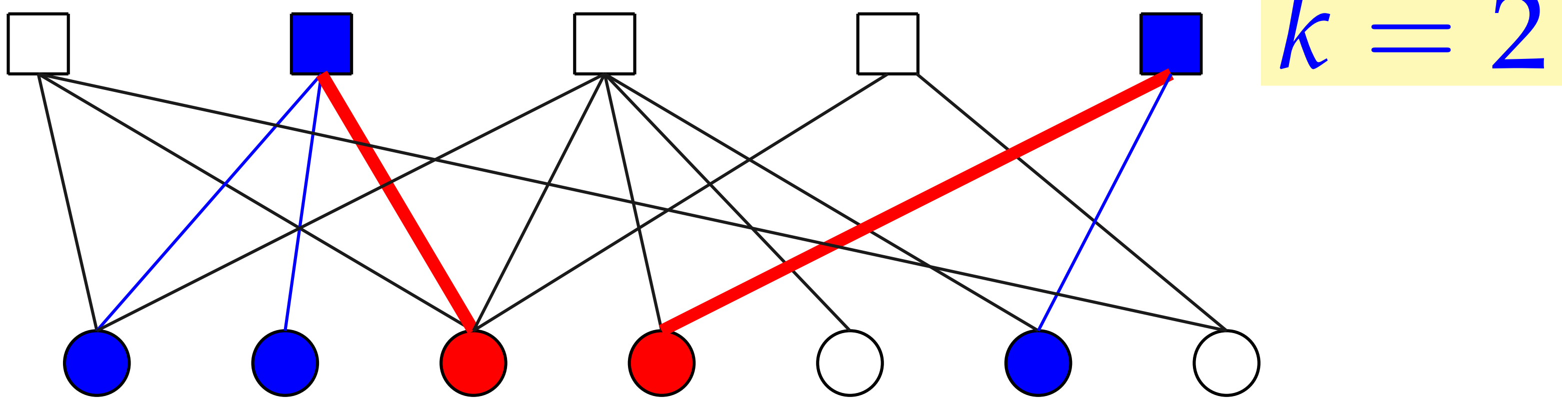
k-OTD(H, t') By SD(H, t)

$k = 2$



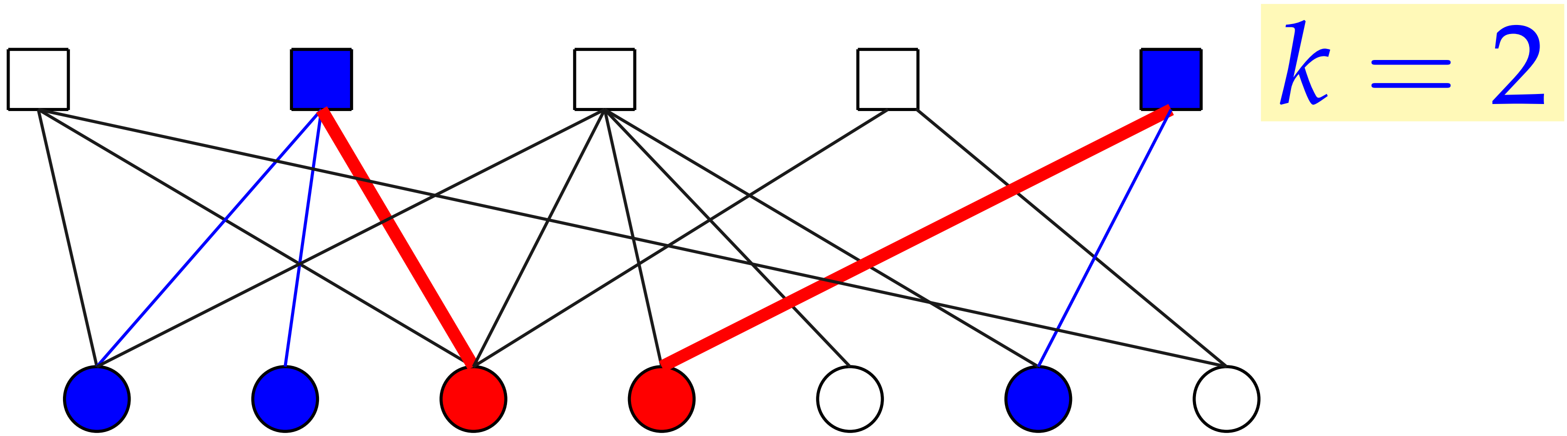
1. Select k edges.

k-OTD(H, t') By SD(H, t)



1. Select k edges.
2. Based on the k check nodes, identify the neighbor variables.

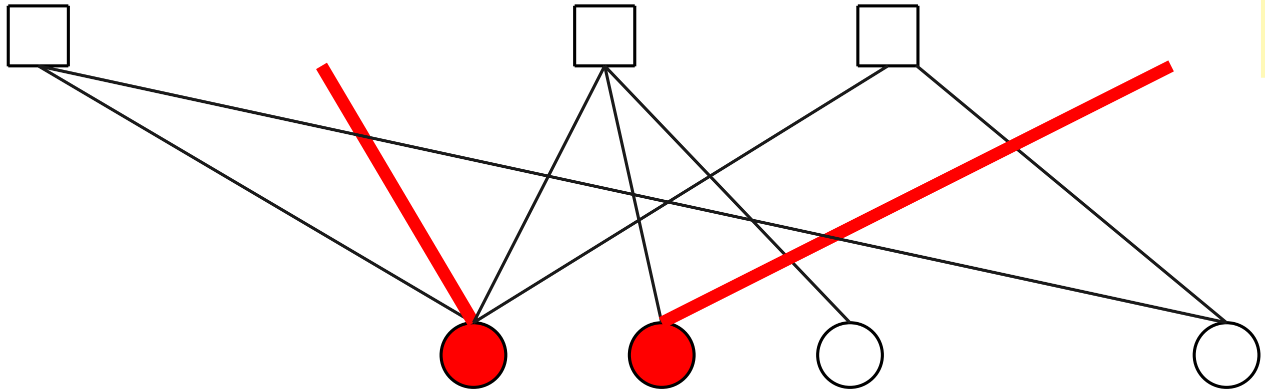
k-OTD(H, t') By SD(H, t)



1. Select k edges.
2. Based on the k check nodes, identify the neighbor variables.
3. Remove the check nodes and neighbor variables.

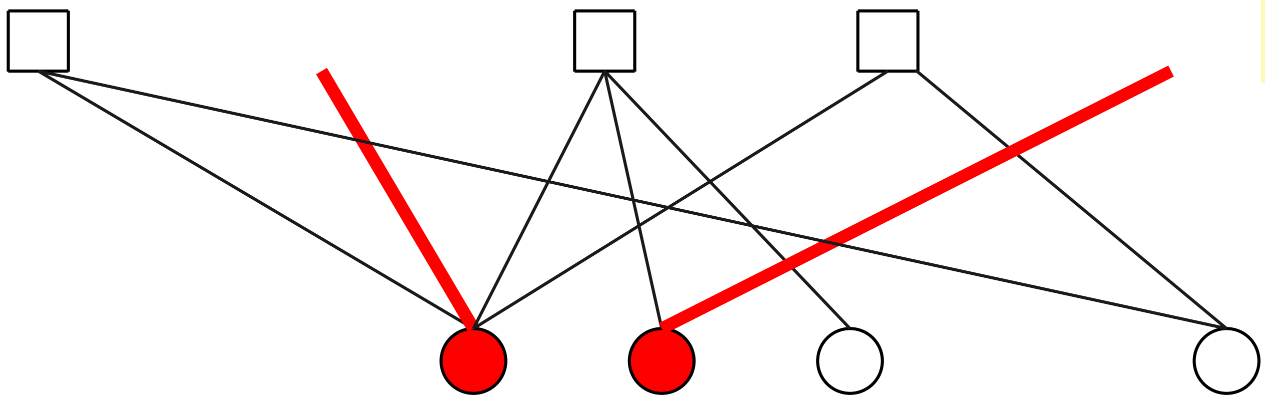
k-OTD(H, t') By SD(H, t)

$k = 2$



1. Select k edges.
2. Based on the k check nodes, identify the neighbor variables.
3. Remove the check nodes and neighbor variables.

k-OTD(H, t') By SD(H, t)

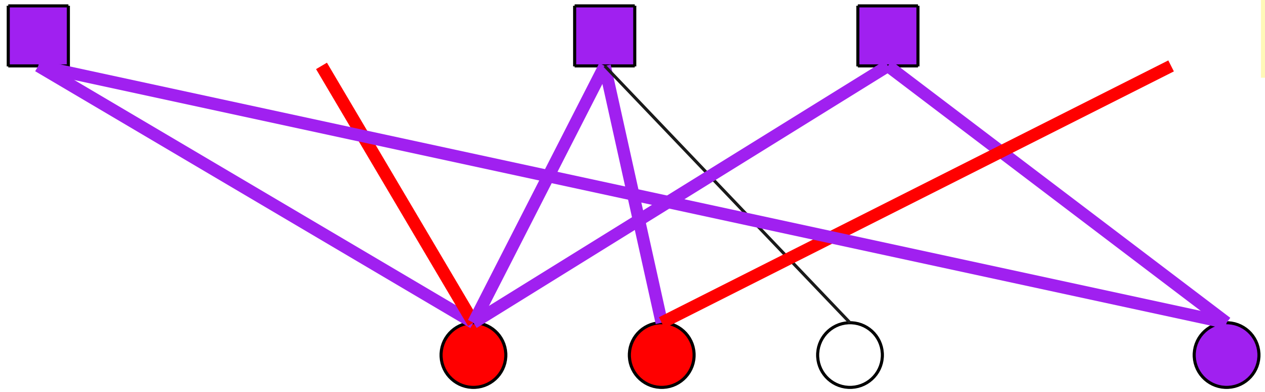


$k = 2$

1. Select k edges.
2. Based on the k check nodes, identify the neighbor variables.
3. Remove the check nodes and neighbor variables.
4. Run SD(H, t) to find the minimal stopping sets containing the interested variables.

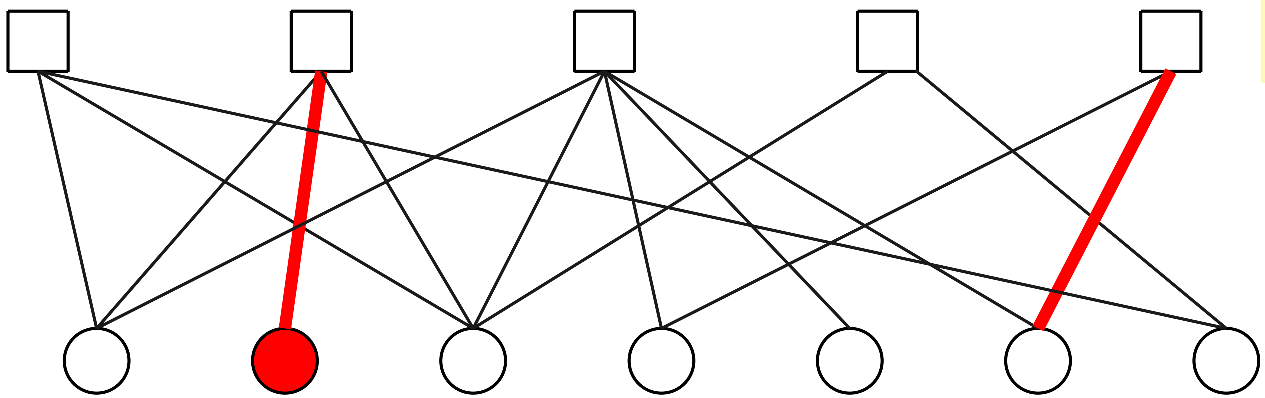
k-OTD(H, t') By SD(H, t)

$k = 2$



1. Select k edges.
2. Based on the k check nodes, identify the neighbor variables.
3. Remove the check nodes and neighbor variables.
4. Run SD(H, t) to find the minimal stopping sets containing the interested variables.

k-OTD(H, t') By SD(H, t)



$k = 2$

1. Select k edges.
2. Based on the k check nodes, identify the neighbor variables.
3. Remove the check nodes and neighbor variables.
4. Run SD(H, t) to find the minimal stopping sets containing the interested variables.
5. Select another k edges and repeat the procedure.

Empirical Study of k-OTD(H, t)

- Complexity grows $O(n^k)$.

Empirical Study of k-OTD(H,t)

- Complexity grows $O(n^k)$. A harder problem than SD(H,t).

Empirical Study of k-OTD(H,t)

- Complexity grows $O(n^k)$. A harder problem than SD(H,t).
- For codes of interest, 50% FER from $k \leq 2$ TS [Richardson 03].

Empirical Study of k-OTD(H, t)

- Complexity grows $O(n^k)$. A harder problem than SD(H, t).
- For codes of interest, 50% FER from $k \leq 2$ TS [Richardson 03].
- When $n \approx 500$ and rate $\frac{1}{2}$ codes, $t = 10$–12 for 1-OTS(H, t). $t = 9$–11 for 2-OTS(H, t), based on our SD(H, t).

Empirical Study of k-OTD(H,t)

- Complexity grows $O(n^k)$. A harder problem than SD(H,t).
- For codes of interest, 50% FER from $k \leq 2$ TS [Richardson 03].
- When $n \approx 500$ and rate $\frac{1}{2}$ codes, $t = 10$–12 for 1-OTS(H,t). $t = 9$–11 for 2-OTS(H,t), based on our SD(H,t).
- **Tanner (155,64,20) code** 04: Minimal 1-out TD ≥ 12, and minimal 2-out TD $= 8$ w. multiplicity 465. All from the following by automorphisms [Tanner *et al.* 04].

$$7, 17, 19, 33, 66, 76, 128, 140$$
$$7, 31, 33, 37, 44, 65, 100, 120$$
$$1, 19, 63, 66, 105, 118, 121, 140$$
$$44, 61, 65, 73, 87, 98, 137, 146$$
$$31, 32, 37, 94, 100, 142, 147, 148.$$

Empirical Study of k-OTD(H,t)

- **Ramanujan-Margulis (2184,1092) Code** w. q = 13, p = 5 [Rosenthal *et al.* 00];
- Inexhaustive results — upper bounds: analytical search [Mackay *et al.* 03], error-impulse search [Hu *et al.* 04]

$$\text{Minimum Hamming distance} \leq 14$$

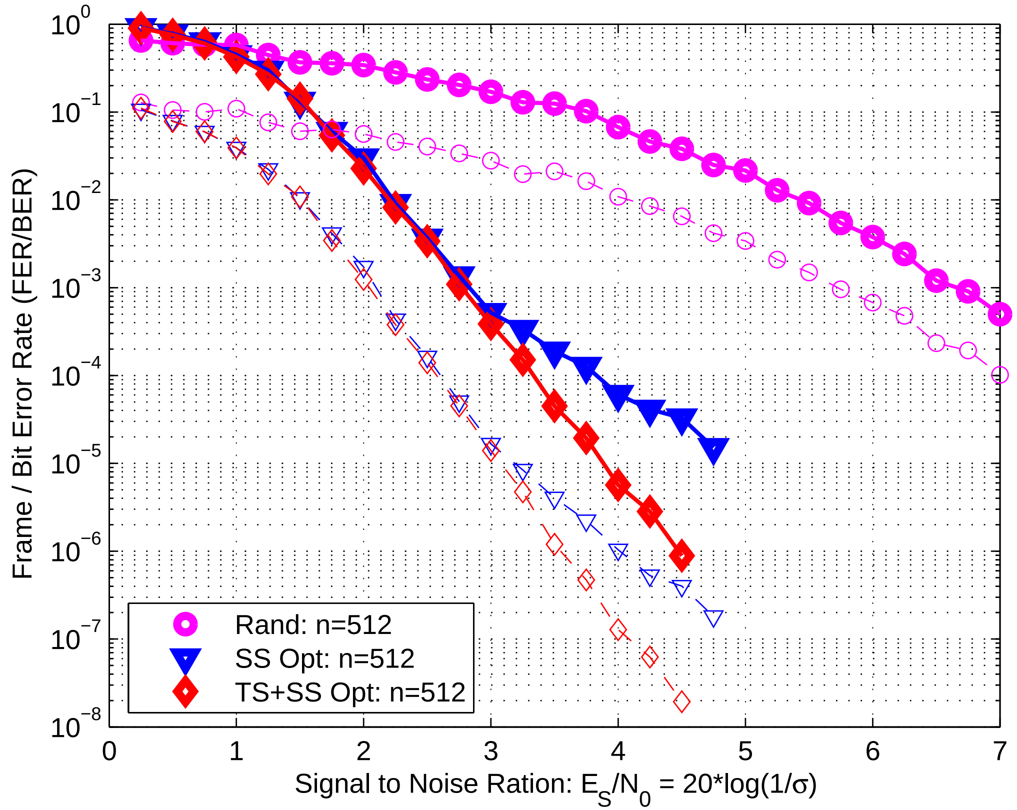
- Exhaustive results by SD(H,t) — lower bounds:

$$\text{Minimum Hamming distance} \geq \text{minimum SD} \geq 14$$
$$\text{multiplicity } 1092$$

Min. 1-out TD $\geq$ 13 and min. 2-out TD $\geq$ 10.

Impact on Error Floors

$$\lambda(x) = 0.31961x + 0.27603x^2 + 0.01453x^5 + 0.38983x^6, \;\; \rho(x) = 0.50847x^5 + 0.49153x^6$$



AWGN, $(\lambda(x), \rho(x))$, $n = 512$, 0-out/1-out trapping sets.

"Rand" $(2,1)$, $(2,8)$; "SS Opt" $(13,40)$, $(5,4)$; "SS+TS Opt" $(11,12)$, $(10,24)$.

Sum-product decoder, 80 iterations, 100 frame errors.

Insufficiency of TSs

- The relationship to error floors.
  - $n = 504$ Girth-optimized Irregular PEG code [Hu *et al.* 05], 1-out TSs of size 7 :

    52, 53, 122, 136, 178, 229, 348

    5, 42, 100, 131, 187, 199, 374

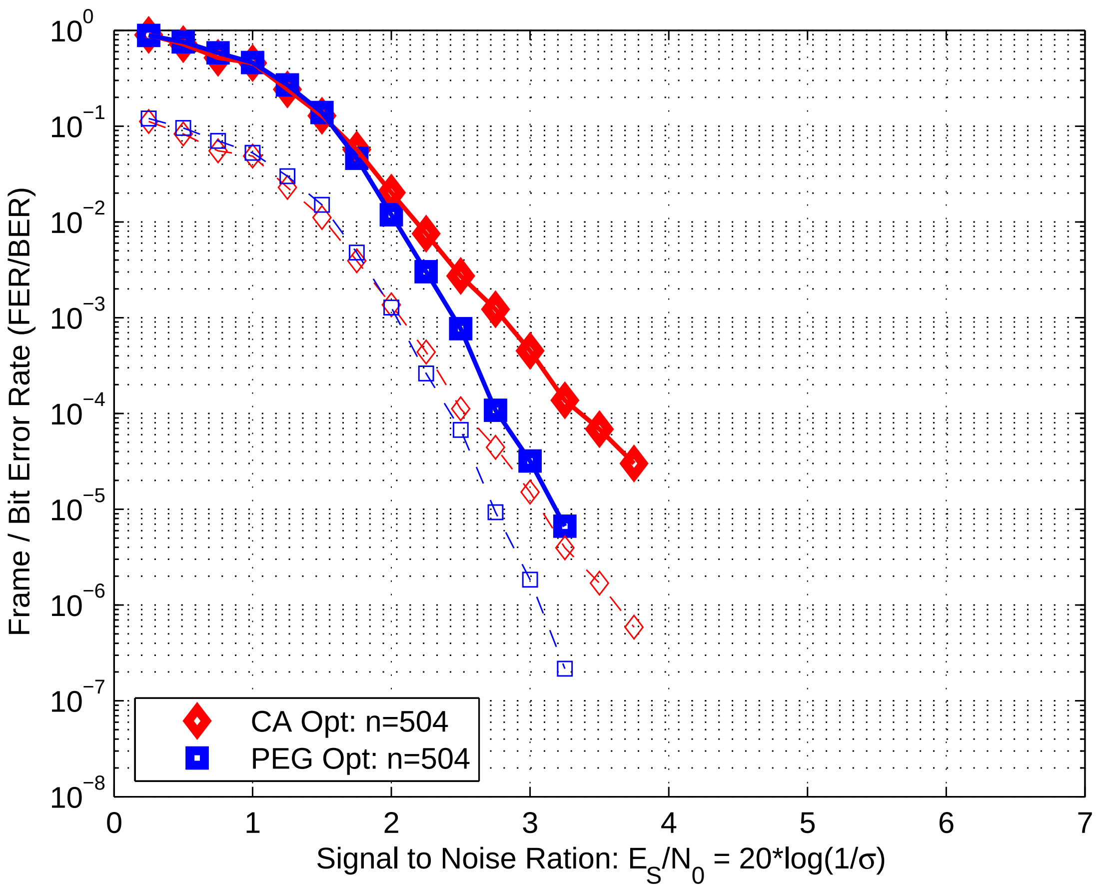
  - $n = 504$ TS-optimized irregular code w. the same deg. distr., 0/1-out TSs: $(10, 7)$/$(8, 40)$.

Insufficiency of TS

- The relationship to error fl
  - $n = 504$ Girth-optimiz
    1-out TSs of size 7 :

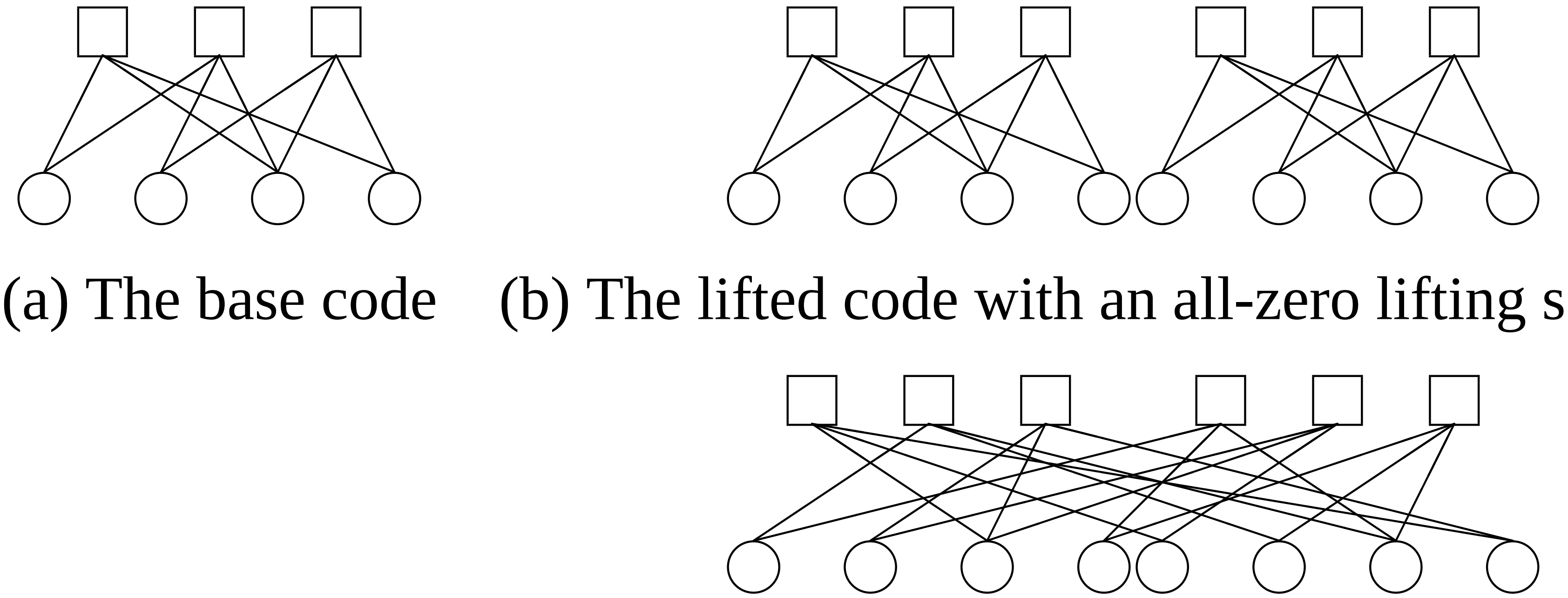
    52, 53

    5, 42
  - $n = 504$ TS-optimized
    0/1-out TSs: $(10, 7)/($



The Cyclically Lifted Ensemble

[Gross 74], [Richardson & Urbanke] and many more.

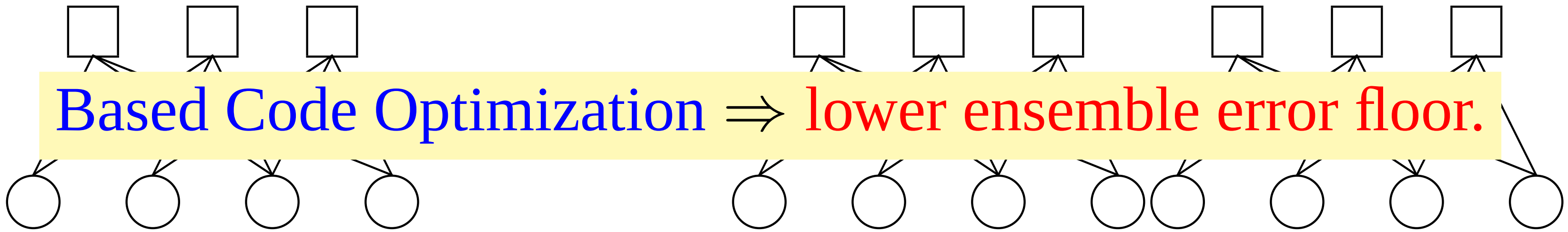


(a) The base code (b) The lifted code with an all-zero lifting sequence

(c) The lifted code with a cyclic lifting sequence.

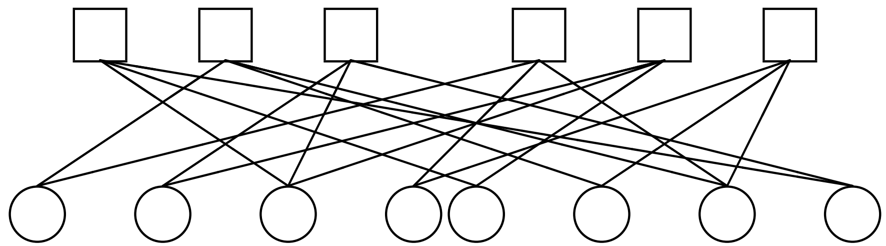
The Cyclically Lifted Ensemble

[Gross 74], [Richardson & Urbanke] and many more.



Based Code Optimization $\Rightarrow$ lower ensemble error floor.

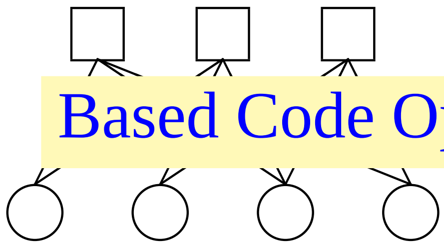
(a) The base code (b) The lifted code with an all-zero lifting sequence

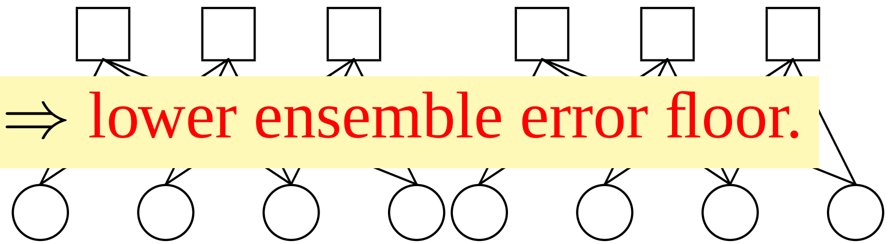


(c) The lifted code with a cyclic lifting sequence.

The Cyclically Lifted Ensemble

[Gross 74], [Richardson & Urbanke] and many more.

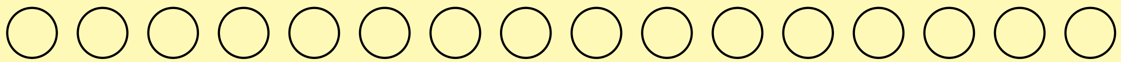




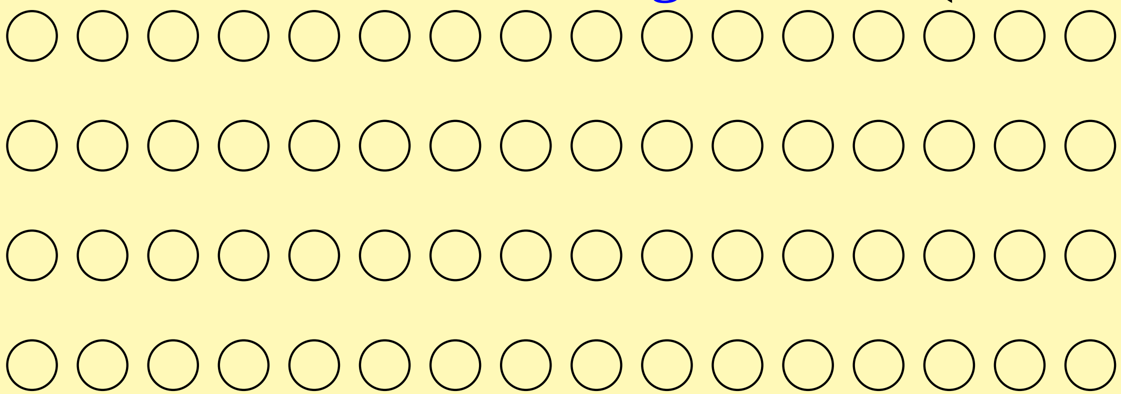
Based Code Optimization $\Rightarrow$ lower ensemble error floor.

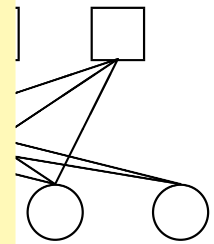
(a) The base code (b) The lifted code with an all-zero lifting sequence

Base Code — of size n ($n = 16$)



Lifted Code — of lifting factor K ($K = 4$)

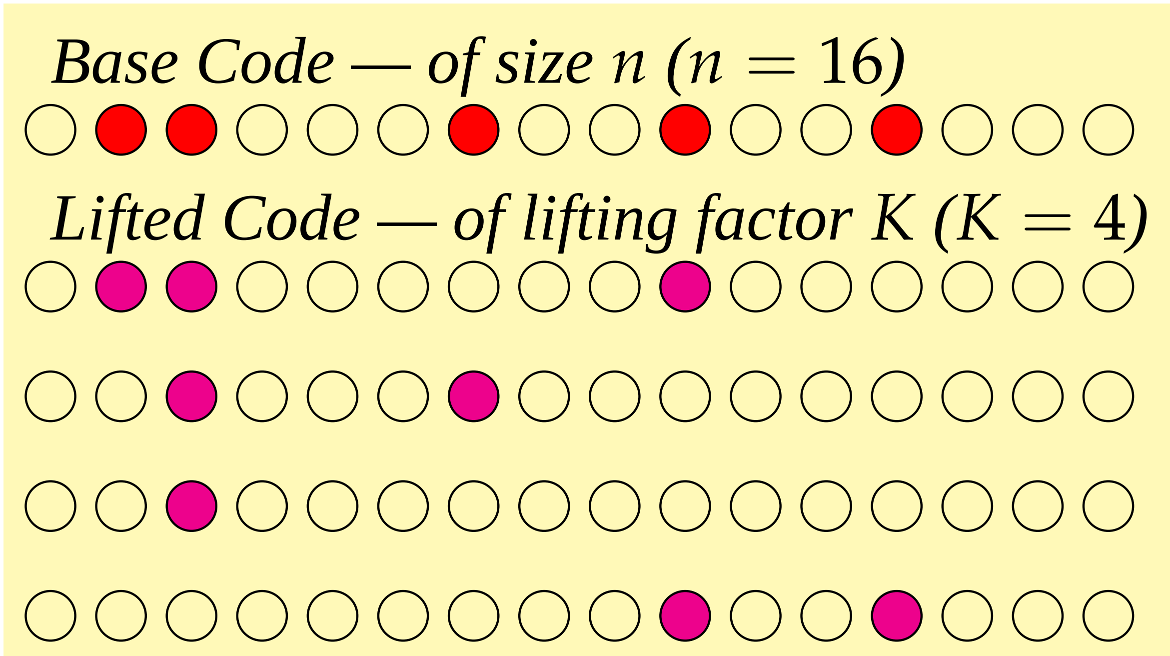




fting sequence.

Survival of Trapping Sets

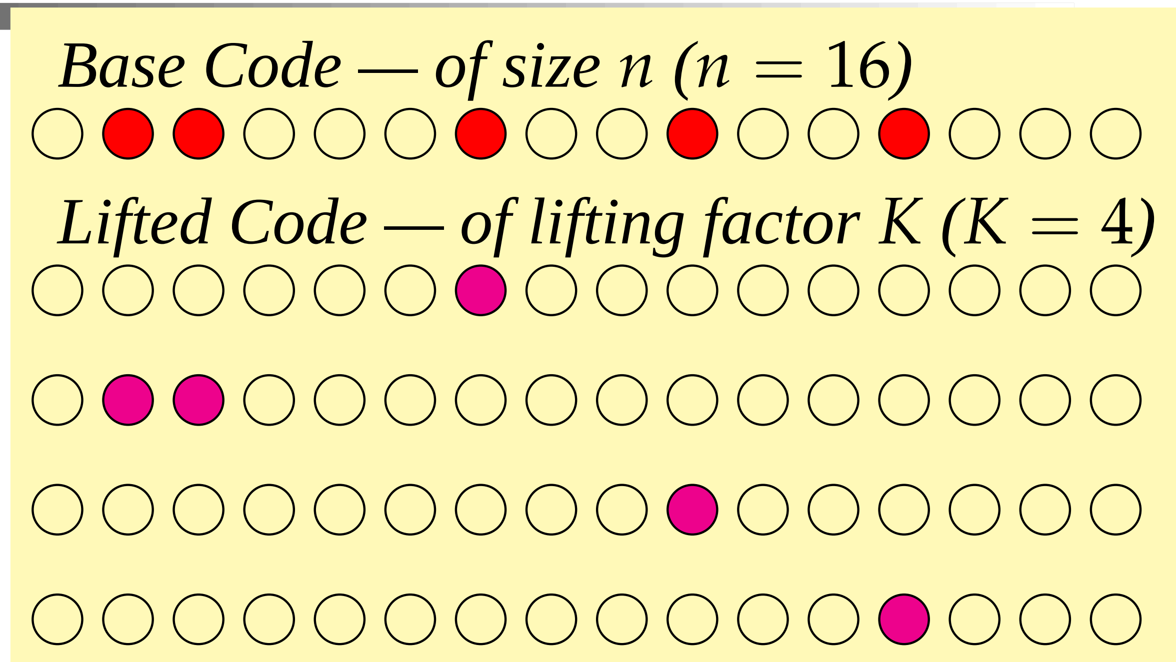
Theorem 3 *If ● forms a k_L-out trapping set for one lifted code, then ● forms a k_B-out trapping set for the base code where $k_L \geq k_B$.*



Different Orders of Survivals

Definition 2

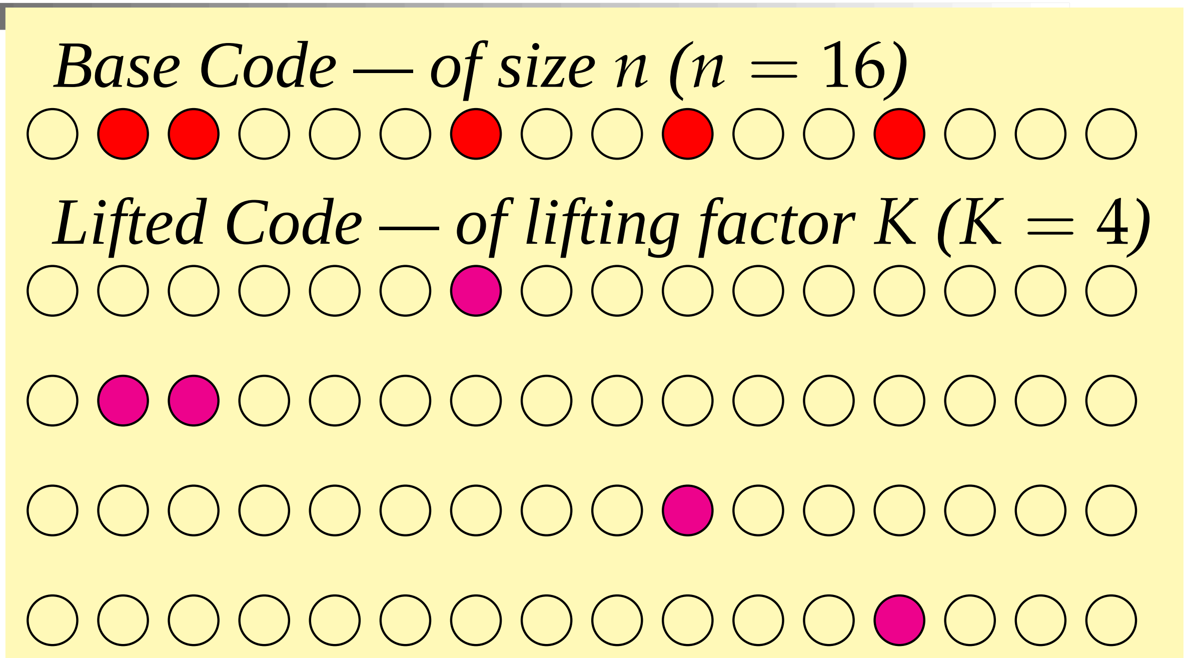
First order survivals



Different Orders of Survivals

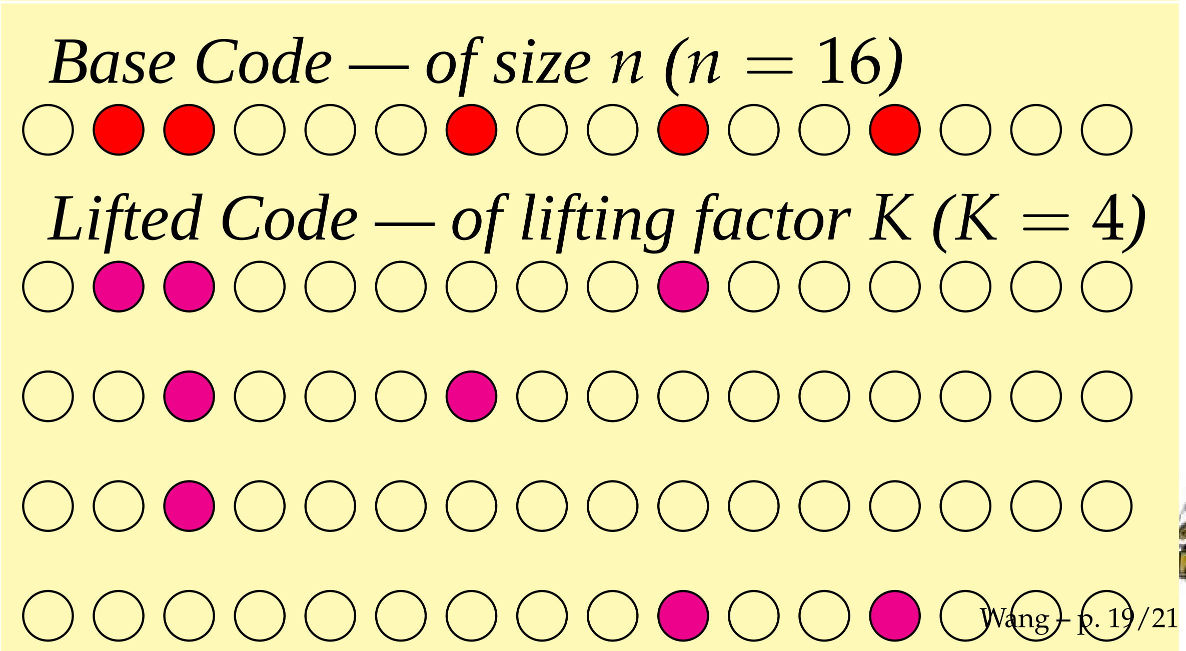
Definition 2

First order survivals



Definition 3

High order survivals

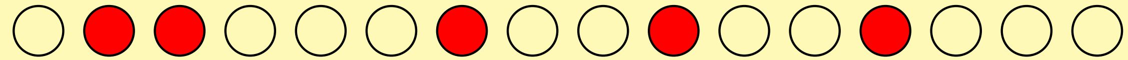


Different Orders of Survivals

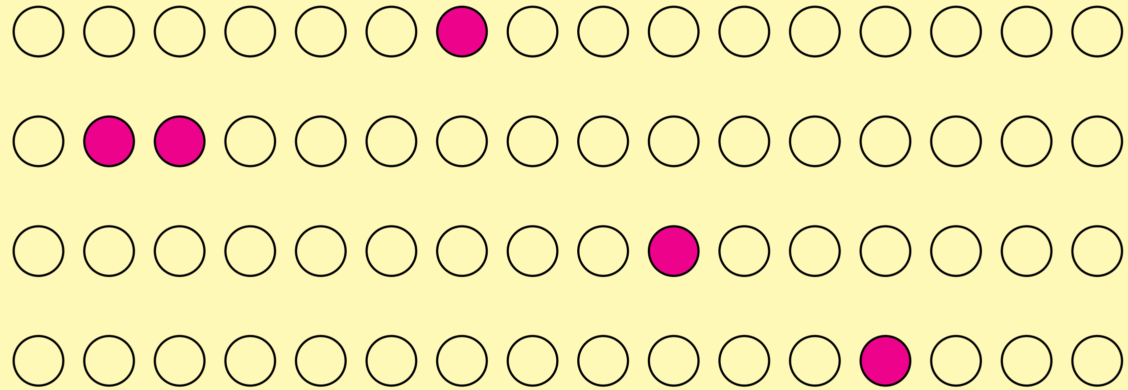
Definition 2

First order survivals

Base Code — of size n ($n = 16$)



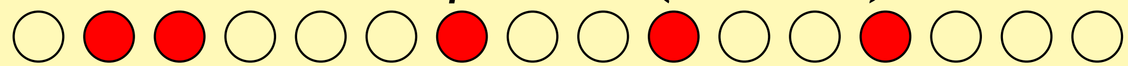
Lifted Code — of lifting factor K ($K = 4$)



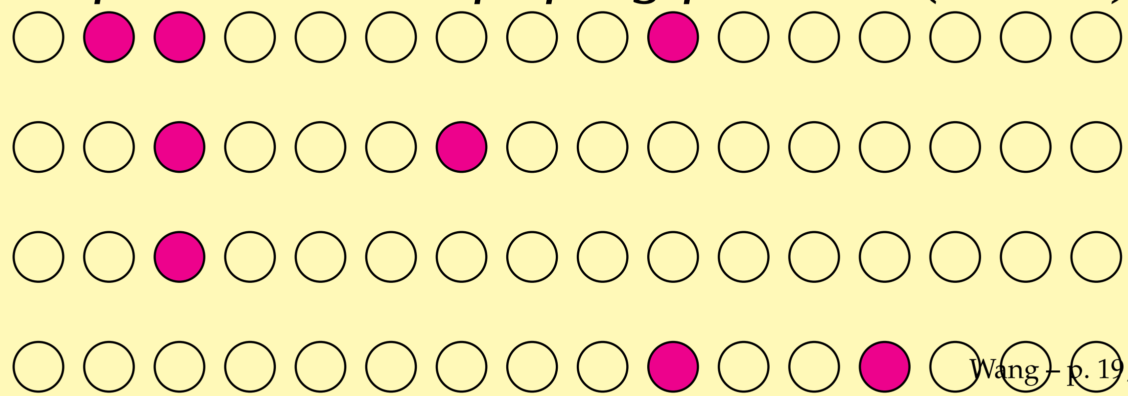
Definition 3

High order survivals

Base Code — of size n ($n = 16$)



Lifted Code — of lifting factor K ($K = 4$)



Empirically, almost all small trapping sets are of first order.

[Wang 06, Ländner 05]

First order survival

Theorem 4 ($k_L = k_B = 0$, a preliminary result) *For a fixed base code with a min. stopping set* $\mathbf{s}_B$,

$$\mathsf{E}\{|\textit{first order survivals}|\} \propto K^{-(0.5\#E - \#V + 0.5\#C_{\mathrm{odd},\geq 3})}$$

$$FER_{BEC,ensemble} = \mathsf{const} \cdot K^{-(0.5\#E - \#V + 0.5\#C_{\mathrm{odd},\geq 3})}.$$

where $\mathsf{const} = f(\textit{the min. stp. dist.}, \textit{multi.})$.

First order survival

Theorem 4 ($k_L = k_B = 0$, a preliminary result) *For a fixed base code with a min. stopping set* $\mathbf{s}_B$,

$$\mathsf{E}\{|\textit{first order survivals}|\} \propto K^{-(0.5\#E-\#V+0.5\#C_{\text{odd},\geq 3})}$$

$$FER_{BEC,ensemble} = \mathsf{const} \cdot K^{-(0.5\#E-\#V+0.5\#C_{\text{odd},\geq 3})}.$$

where const $= f(\textit{the min. stp. dist.}, \textit{multi.})$.

Theorem 5 ($k_L = k_B > 0$) *For a* *base-code k-out trapping set* $\mathbf{t}_B$,

$$\mathsf{E}\{|\textit{first order survivals}|\} \propto K^{0.5k_B} K^{-(0.5\#E-\#V+0.5\#C_{\text{odd},\geq 3})}.$$

First order survival

Theorem 4 ($k_L = k_B = 0$, a preliminary result) *For a fixed base code with a min. stopping set* $\mathbf{s}_B$,

$$\mathsf{E}\{|\textit{first order survivals}|\} \propto K^{-(0.5\#E - \#V + 0.5\#C_{\text{odd},\geq 3})}$$

$$FER_{BEC,ensemble} = \mathsf{const} \cdot K^{-(0.5\#E - \#V + 0.5\#C_{\text{odd},\geq 3})}.$$

where const $= f(\textit{the min. stp. dist.}, \textit{multi.})$.

Theorem 5 ($k_L = k_B > 0$) *For a base-code k-out trapping set* $\mathbf{t}_B$,

$$\mathsf{E}\{|\textit{first order survivals}|\} \propto K^{0.5k_B} K^{-(0.5\#E - \#V + 0.5\#C_{\text{odd},\geq 3})}.$$

Theorem 6 ($k_L = k_B + 1$) *For a base-code k-out trapping set* $\mathbf{t}_B$,

$$\mathsf{E}\{|\textit{first order survivals}|\} \propto K^{0.5k_B} K^{-(0.5\#E - \#V + 0.5\#C_{\text{odd},\geq 3})} (K\#C_{\text{odd},\geq 3} + \#C_{\text{even},\geq 4}).$$

First order survival

Theorem 4 ($k_L = k_B = 0$, a preliminary result) *For a fixed base code with a min. stopping set $\mathbf{s}_B$,*

$$\mathsf{E}\{|\textit{first order survivals}|\} \propto K^{-(0.5\#E - \#V + 0.5\#C_{\text{odd},\geq 3})}$$

$$FER_{BEC,ensemble} = \mathsf{const} \cdot K^{-(0.5\#E - \#V + 0.5\#C_{\text{odd},\geq 3})}.$$

where $\mathsf{const} = f(\textit{the min. stp. dist.}, \textit{multi.})$.

Theorem 5 ($k_L = k_B > 0$) *For a base-code k-out trapping set $\mathbf{t}_B$,*

$$\mathsf{E}\{|\textit{first order survivals}|\} \propto K^{0.5k_B} K^{-(0.5\#E - \#V + 0.5\#C_{\text{odd},\geq 3})}.$$

Theorem 6 ($k_L = k_B + 1$) *For a base-code k-out trapping set $\mathbf{t}_B$,*

$$\mathsf{E}\{|\textit{first order survivals}|\} \propto K^{0.5k_B} K^{-(0.5\#E - \#V + 0.5\#C_{\text{odd},\geq 3})} (K\#C_{\text{odd},\geq 3} + \#C_{\text{even},\geq 4}).$$

Base code optimization: $0.5\#E - \#V + 0.5\#C_{\text{odd},\geq 3}$

First order survival

Theorem 4 ... *e*

code with a

E{|*f*

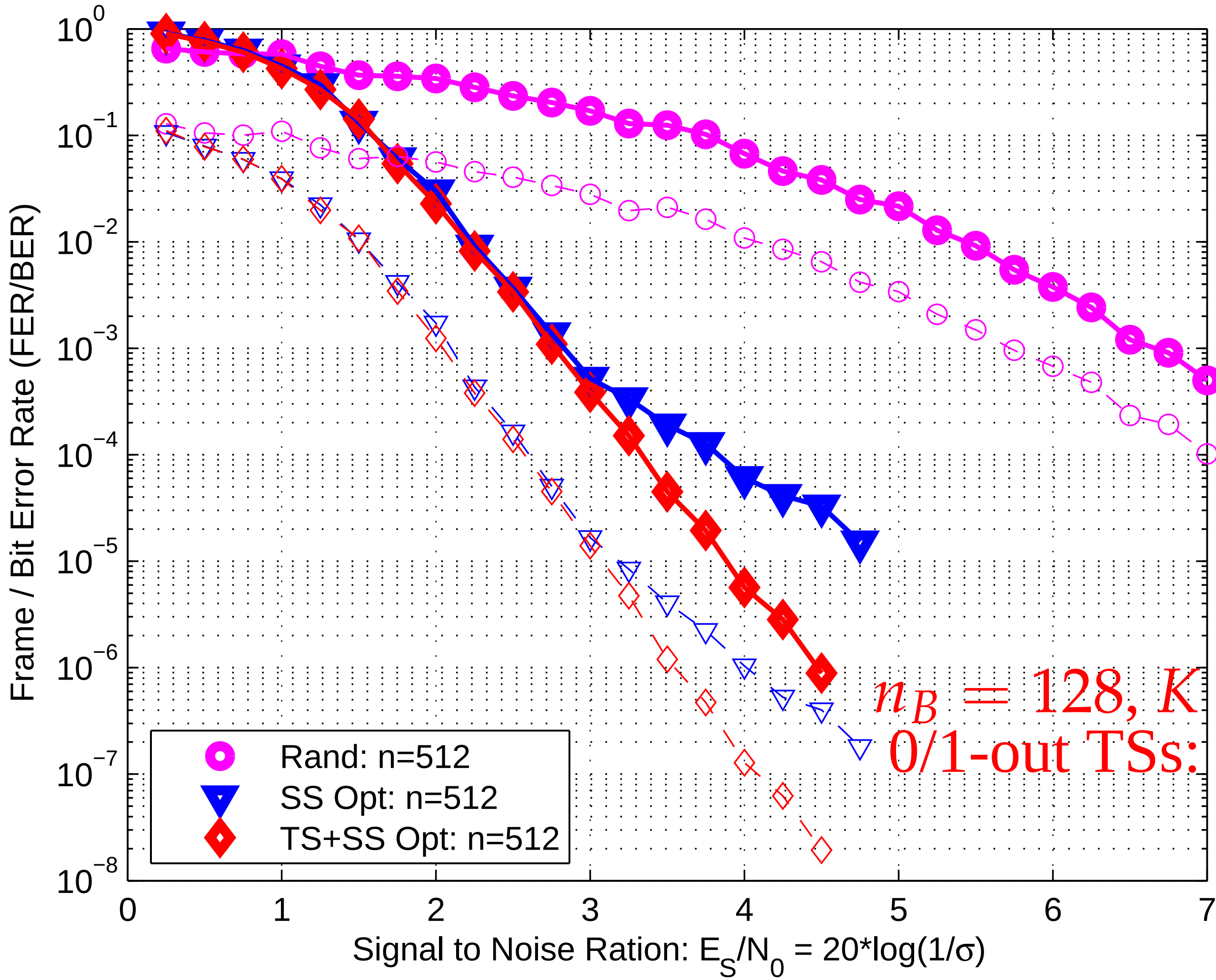
where cons

Theorem 5 ... $_3$,

E{|

Theorem 6 ... ,

E{|*first orde* ... $\#C_{\text{even},\geq 4})$.



Base code optimization: $0.5\#E - \#V + 0.5\#C_{\text{odd},\geq 3}$

Conclusion

- Define the k-out trapping set graph-theoretically.

Conclusion

- Define the k-out trapping set graph-theoretically.
- Deciding the minimal k-out trapping distance is NP-hard.

Conclusion

- Define the k-out trapping set graph-theoretically.
- Deciding the minimal k-out trapping distance is NP-hard.
- But still doable for practical code lengths $n \approx 500$.

Conclusion

- Define the k-out trapping set graph-theoretically.
- Deciding the minimal k-out trapping distance is NP-hard.
- But still doable for practical code lengths $n \approx 500$.
- Implement k-OTD(H, t) by SD(H, t).

Conclusion

- Define the k-out trapping set graph-theoretically.
- Deciding the minimal k-out trapping distance is NP-hard.
- But still doable for practical code lengths $n \approx 500$.
- Implement $k\text{-OTD}(H,t)$ by $\text{SD}(H,t)$.
- Insufficiency of the trapping set (near-codeword) .

Conclusion

- Define the k-out trapping set graph-theoretically.
- Deciding the minimal k-out trapping distance is NP-hard.
- But still doable for practical code lengths $n \approx 500$.
- Implement k-OTD(H, t) by SD(H, t).
- Insufficiency of the trapping set (near-codeword) .
- Quantifying the suppressing effect of cyclic lifting for trapping sets.

Conclusion

- Define the k-out trapping set graph-theoretically.
- Deciding the minimal k-out trapping distance is NP-hard.
- But still doable for practical code lengths $n \approx 500$.
- Implement k-OTD(H, t) by SD(H, t).
- Insufficiency of the trapping set (near-codeword) .
- Quantifying the suppressing effect of cyclic lifting for trapping sets.
- Base code optimization: $0.5\#E - \#V + 0.5\#C_{\text{odd},\geq 3}$.