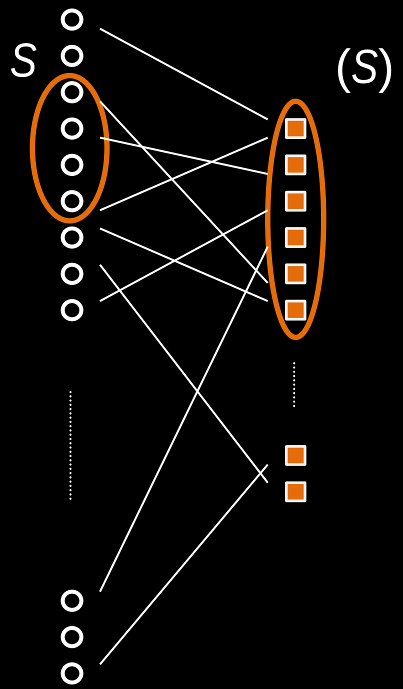
Error correction guarantees

Independence assumption

- If n is the total number of variable nodes, this puts an upper bound on l (of the order $\log(n)$)
- $l = \log(n)$ number of iterations is usually not enough to prove that the decoding process corrects all errors.
-

Expanders

- Definition: A bipartite graph with *n* variable nodes is called an (,)-expander if for any subset *S* of the variable nodes of size at most *n* the number of (check node) neighbors of *S* is at least a_S |*S*|, where a_S is the average degree of the nodes in *S*.



$$|S| \quad n \qquad | \ (S)| \quad a_S\,|S|$$

- Remark: if there are many edges going out of a subset of message nodes, then there should be many different (unshared) neighbors.

Decoding on the BEC

BEC

- Theorem: If a Tanner graph is an (,1/2)-

F

Proof - continuation

- No node in (*S*) has degree 1, since this neighbor would recover one element in *S* and would contradict the minimality of *S*. Hence, the total number of edges emanating from these nodes is at least 2| (*S*)|.

- On the other hand, the total number of edges emanating from *S* is $a_S|S|$, so $a_S\ |S|\ \ 2|\ (S)|$,

- which implies | (*S*)| $a_S\ |S|$/]TJEBB/087>]TJ5 0 0 o.22EEB/05 24

*

Decoding on BSC

- Parallel bit-flipping algorithm:
- While there are unsatisfied check bits
  - Find a bit for which more than d/2 neighboring checks are

Bit-flipping decoder on BSC

- Observation: The decoder progresses with correcting errors as long there are bits for which more than $d_v/2$ neighboring checks are unsatisfied.
- What property on the graph ensures that

Expander arguments

- Sipser and Spielman (1996): Let *G* be a $(d_v, d_c, \alpha, (¾ + Ů)d_v)$ expander over *n* variable nodes, for any Ů> 0. Then, the parallel bit flipping algorithm will correct any $Ŭ_0 < Ŭ(1+4\varepsilon)/2$ fraction of error after $\log_{1/(1-4\varepsilon)}(\alpha_0 n)$ decoding rounds

- Burshtein and Miller, (2001): "Expander graph arguments for message passing algorithms"
- Feldman *et al.* (2003): " LP Decoding corrects a constant fraction } ʿ א c b" "

Drawbacks of expander arguments

- Bounds derived using random graph arguments on the fraction of nodes having sufficient expansion are very pessimistic
  - Richardson and Urbanke (2003): In the (5,6) regular code ensemble, minimum distance is 3% of code length. But only 3.375×10^{-11} fraction of nodes have expansion of $(3/4)d_v$
- Expansion arguments cannot be used for column-weight-three codes (they work for d_v 5)

Girth and column-weight

- The expansion arguments rely on properties of random graphs and hence do not lead to explicit construction of codes.
- Ii the expansion properties can be related to the parameters of the Tanner graph, such as *g, and* d_v , then the bounds on guaranteed error correction capability can be established as function of these parameters.

Finite length analysis goals

- Establish a connection between guaranteed error correction capability and graph parameters such as *g*, girth, and d_v , variable degree
- Column weight $d_v=3$ is the main focus

Number of correctable errors and FER

- Consider the BSC, and let c_k - the number of configurations of received bits for which *k* channel error lead to a codeword (frame) error.
- Let *i* - the minimal number of channel errors that can lead to a decoding error. Then

$$\sum_{k=i}^{n} c_k \varepsilon^k (1-\varepsilon)^{(n-k)} = FER(\varepsilon)$$

- When $\varepsilon \ll 1$

$$\log(FER(\varepsilon)) \approx \log(c_i) + i\log(\varepsilon)$$

Frame error rate (FER)

- What is usually plotted (semi-log scale):

$$FER(\alpha) = \sum_{k=i}^{n} c_k \alpha^k (1-\alpha)^{n-k}$$

$$\log(FER(\alpha)) = \log(c_i) + i\log(\alpha) + \log((1-\alpha)^{n-i}) + \log\left(1 + \frac{c_{i+1}}{c_i}\alpha(1-\alpha)^{-1} + \dots + \frac{c_n}{c_i}\alpha^{n-i}(1-\alpha)^{i-n}\right)$$

- As the error probability decreases…

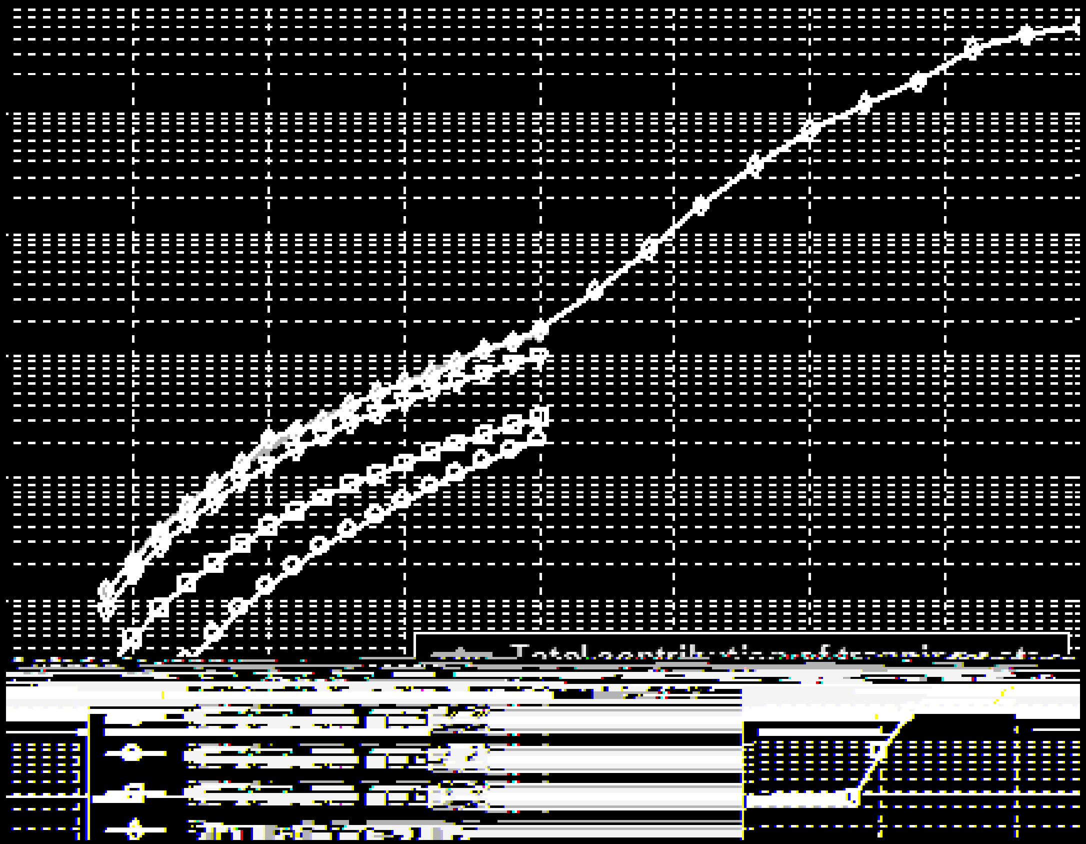
$$\lim_{\alpha \to 0}\left[\log((1-\alpha)^{n-i})\right] = 0$$

$$\lim_{\alpha \to 0}\left[\log\left(1 + \frac{c_{i+1}}{c_i}\alpha(1-\alpha)^{-1} + \dots + \frac{c_n}{c_i}\alpha^{n-i}(1-\alpha)^{i-n}\right)\right] = 0$$

$$\log(FER(\alpha)) \approx \log(c_i) + i\log(\alpha)$$

Practical problems related to error floor

FER contribution of different error patterns



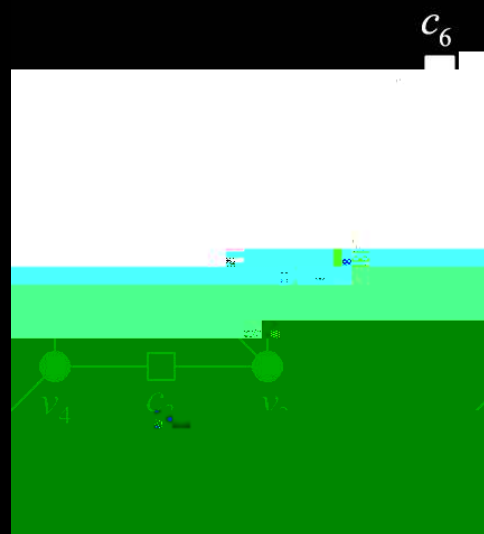
Basic concepts

- An *eventually correct* variable node
- *A fixed point* of iterative decoding
- *Inducing set*
- *Fixed set*
- The *critical number* *m* of a trapping set is the minimal number of variable nodes that have to be initially in error for the decoder to end up in that trapping set.

- An *(a,b)* trapping set: a set of not eventually correct variabe nodes of size *a,* and the *b* odd degree check nodes in the sub-graph induced by these variable nodes.

Trapping sets of various decoders

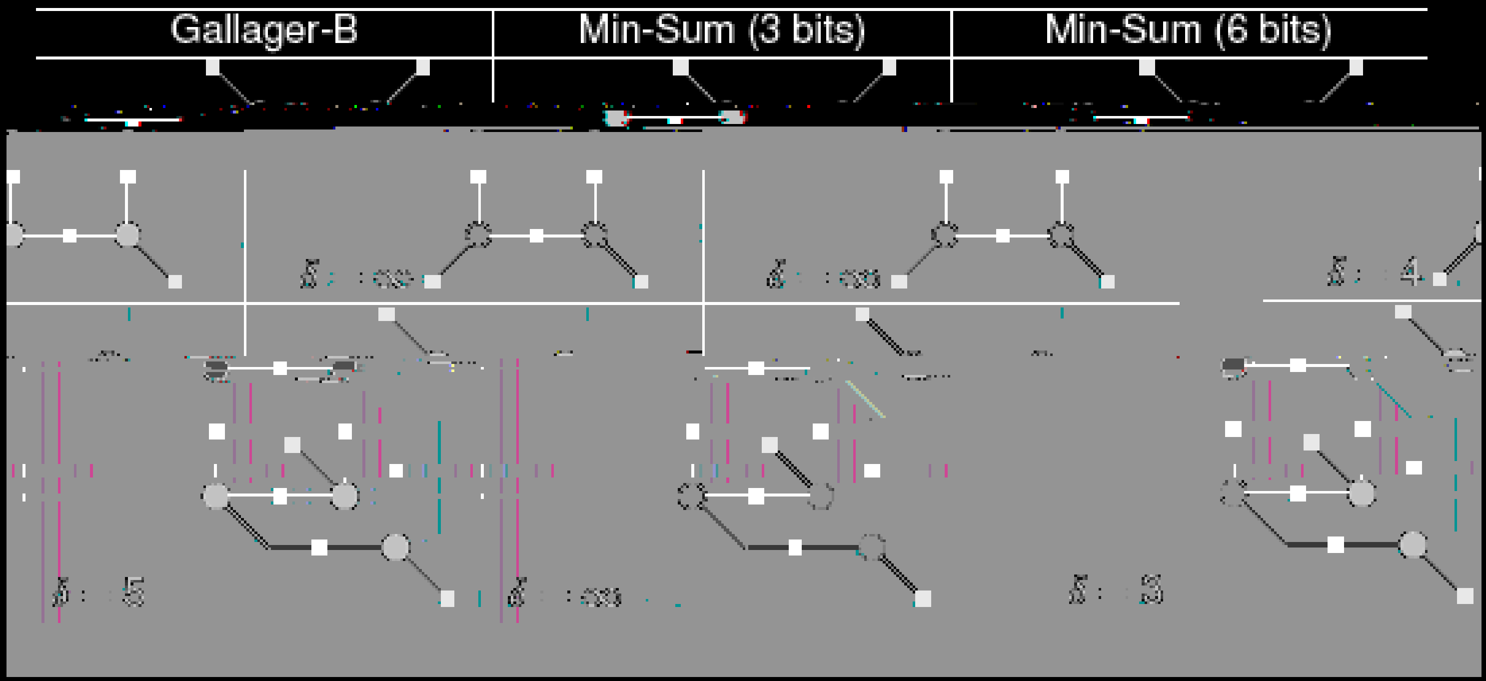
- The decoding failures for various algorithms on different channels are closely related
- Example BSC:



  - Bit flipping algorithm: $\{v_1, v_3\}$, $\{v_2, v_4\}$, $\{v_1, v_2, v_3\}$…
  - Gallager A/B algorithm: $\{v_2, v_4, v_5\}$
  - LP decoder: $\{v_1, v_2, v_3, v_4, v_5\}$

Critical number

- The *critical number m* of a trapping set (for a given decoder) is the minimal number of variable nodes that have to be initially in error for the decoder to end up in that trapping set



Inducing sets and fixed sets

- *Definition 2:* Let *T* be a trapping set. If *( )* = *T* then supp( ) is an <u>*inducing set*</u> of *T*.
- *Definition 3: Let T* be a trapping set and let (*T*) = { | ( ) = *T* }. The <u>*critical number*</u> *m*(*T*) of trapping set *T* is the minimal number of variable nodes that have to be initially in error for the decoder to end up in the trapping set *T* , i.e. $m(T) = \min_{Y(T\ )} |\mathrm{supp}(\ )|$
- *Definition 4:* The vector is *a* <u>*fixed point*</u> of the decoding algorithm if supp( ) = supp(^{l}) for all *l*.
- *Definition 5:* If *T( )* is a trapping set and is a fixed point, then

The (*a*,*b*) notation

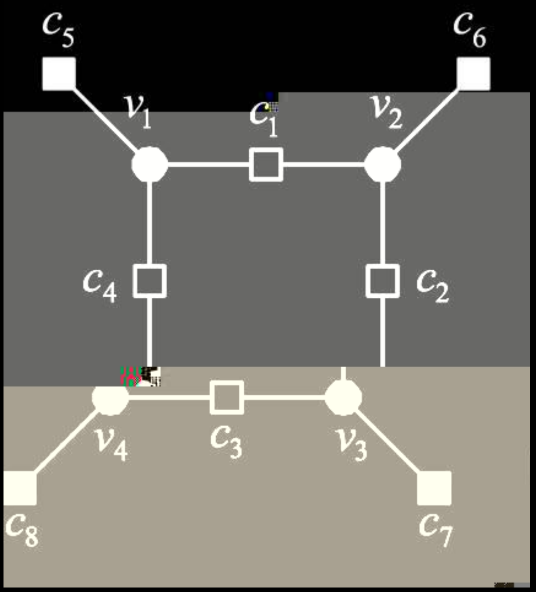
- A *(a,b) trapping set* is a set of *a* variable nodes whose induced sub-graph has *b* odd degree checks
- The most important parameter – critical number:
  - The minimal number of variable nodes that have to be initially in error for the decoder to end up in the trapping set
- To "*end up*" in a trapping set means that (after a finite number of iterations) the decoder will be in error, on at least one variable node at every iteration

Trapping sets for column weight-three codes

- *Theorem* [Chillapagari *et al.,* (2009)]: (sufficient conditions) Let be a subgraph induced by the set of variable nodes *T*. Let the checks in can be partitioned into two disjoint subsets: *E* consisting of checks with even degree, and *O* consisting of checks with odd degree. The vector

Graphical Representation

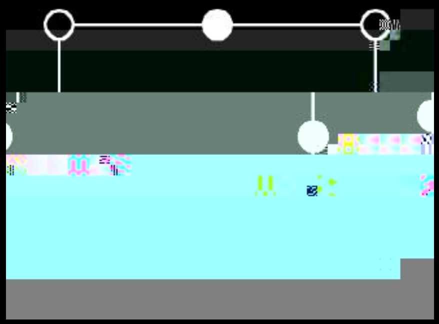
- Tanner graph representation



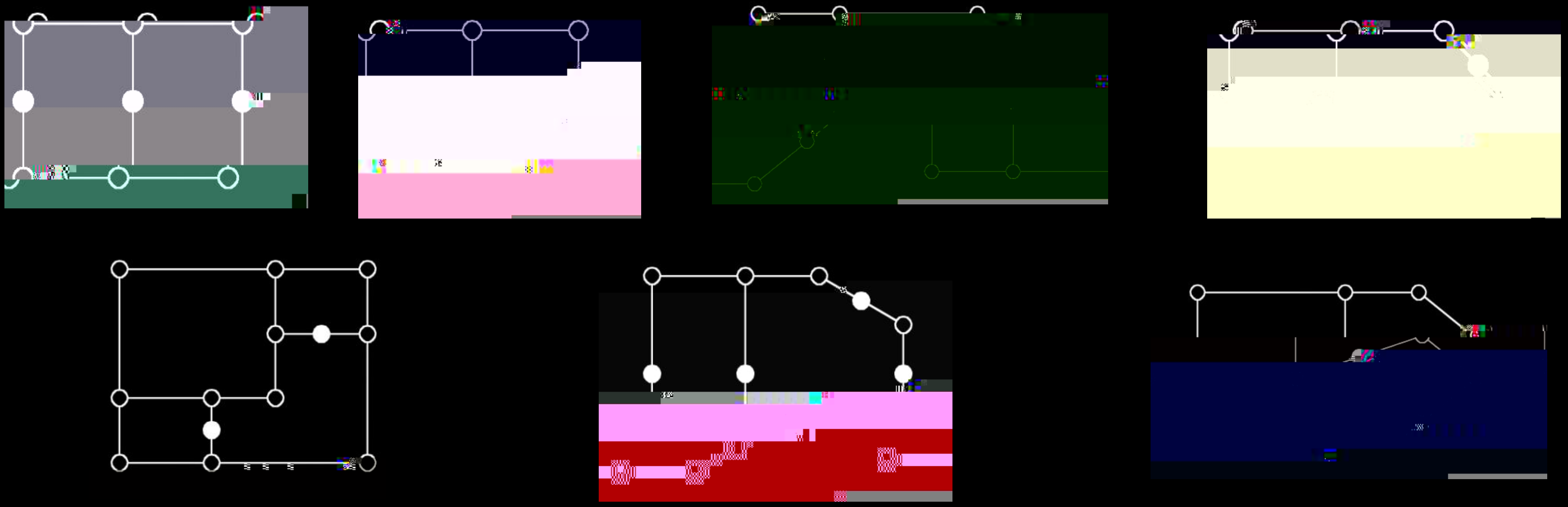
- Line and point representation

Trapping set ontology

- Parent

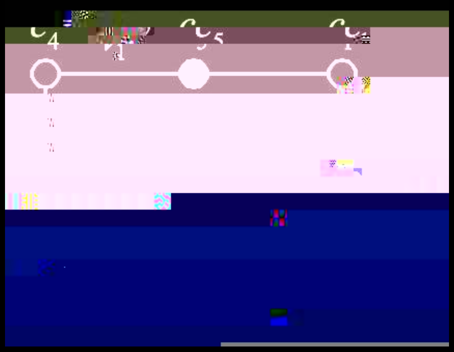


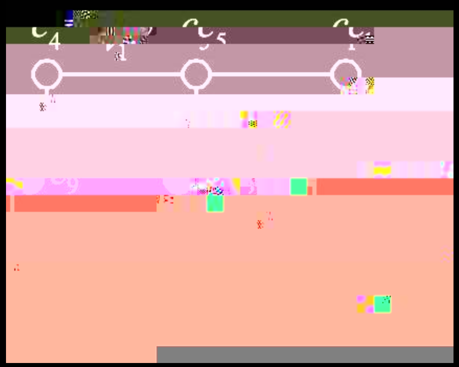
- Children

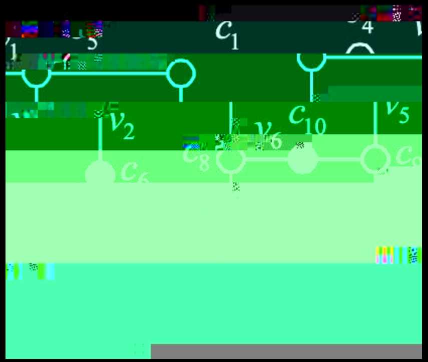


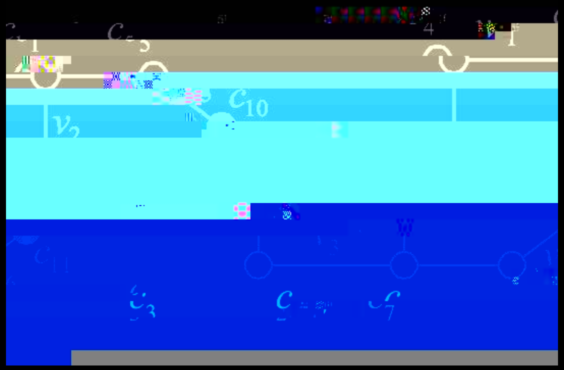
Trapping Set Ontology

- Children are obtained by adding lines to parents, changing the color of the points accordingly.
- Examples:









Evolution

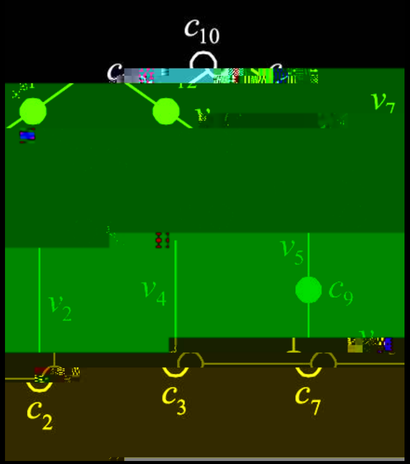
On the critical number of trapping sets

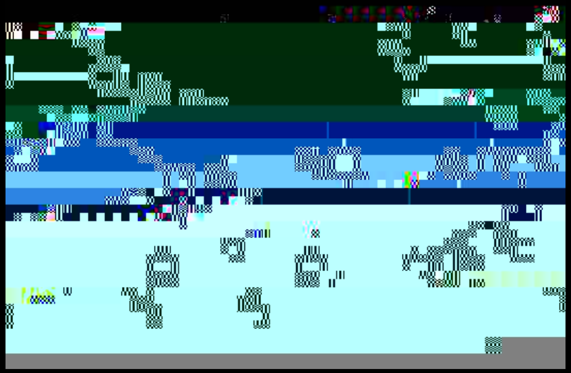
- Conjecture:

On the critical number of trapping sets

- Examples:

- Two (7,3) trapping sets: different in critical number



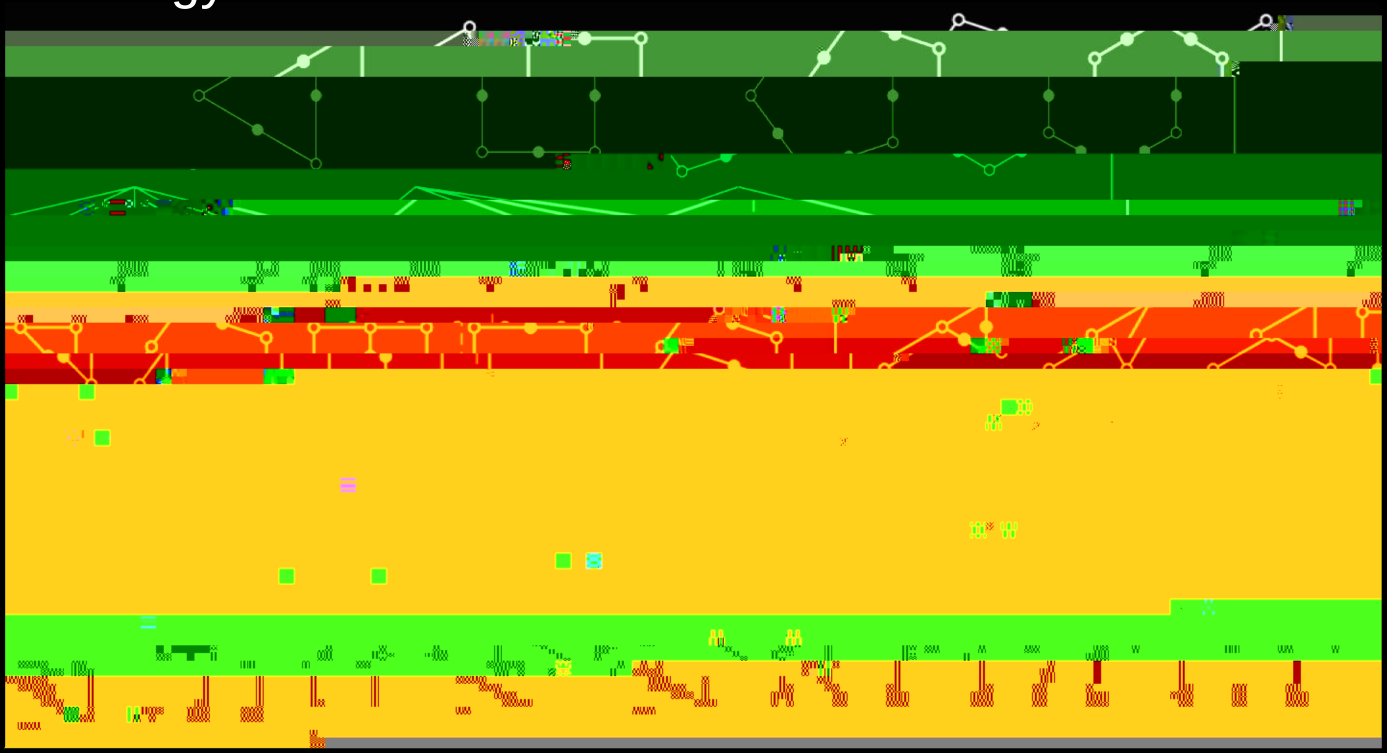


Child of (5,3)
Critical number = 3
(more harmful)

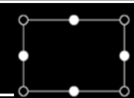
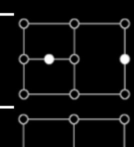
Child of (6,4)
Critical number = 4
 (less harmful)

Trapping set ontology

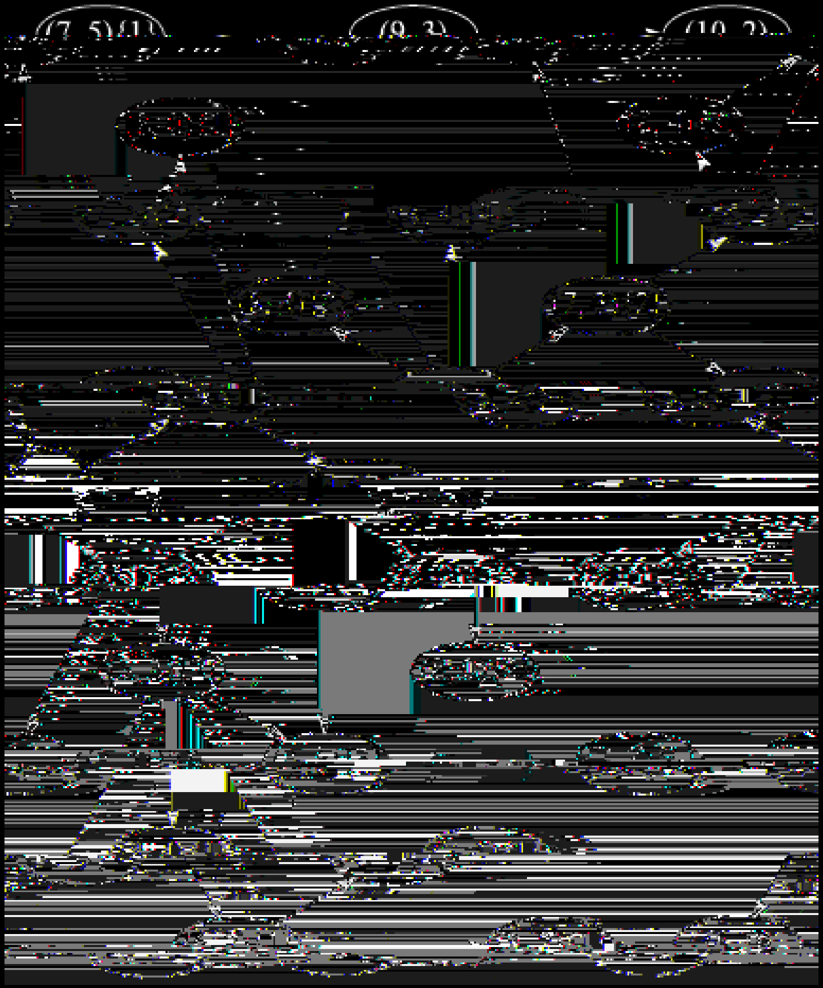
- Allerton 2009: trapping set ontology
- A database and software for systematic study of failures of iterative decoders on BSC
  http://www.ece.arizona.edu/vasiclab/Projects/CodingTheory/TrappingSetOntology.html



Number of trapping sets

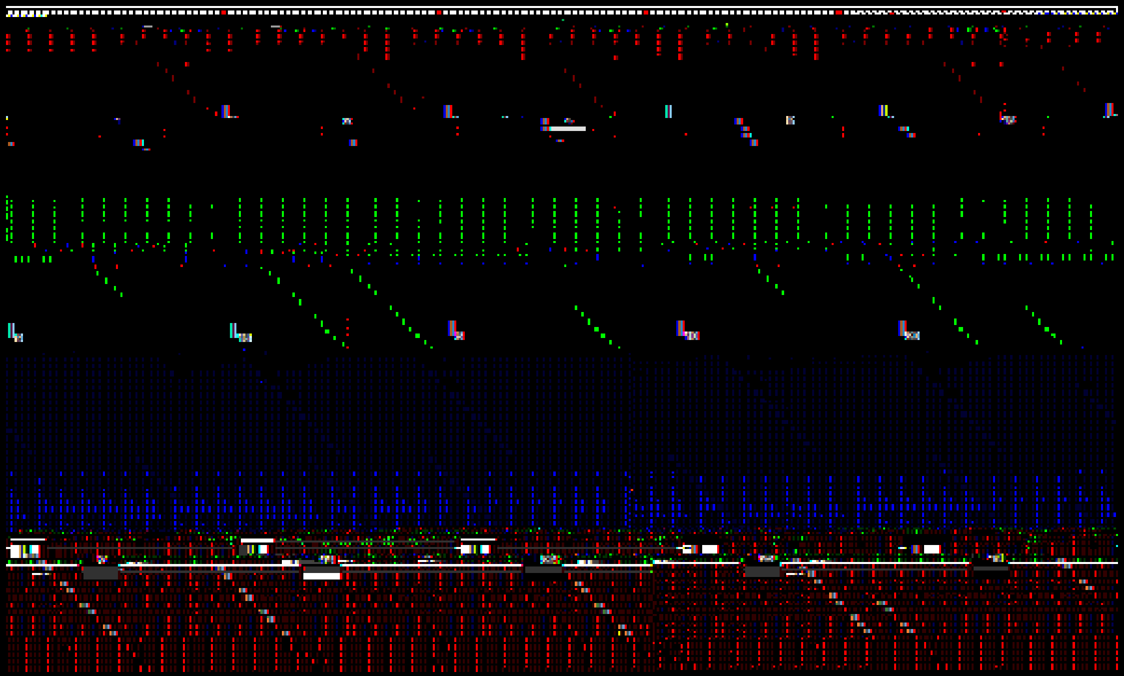
TS	#TS	g	g	g	g
					
					

Trapping Set Ontology

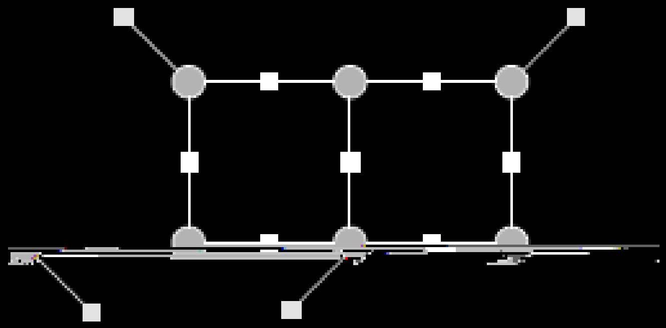


Example: Tanner code

- A good test case (d_{min}=20, blocks of size 31, all codewords, trapping sets repeat 31 times)



Cycle inventory in different (*a*,*b*) topologies

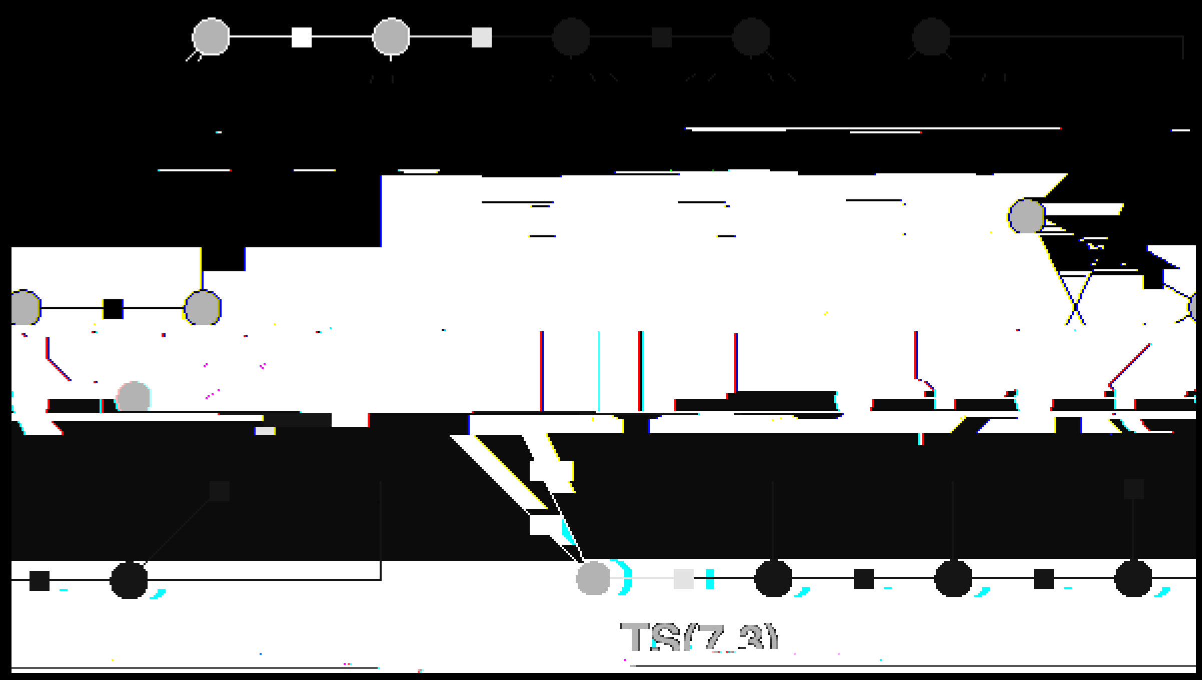


TS(6,4) 2-0-1-0-0

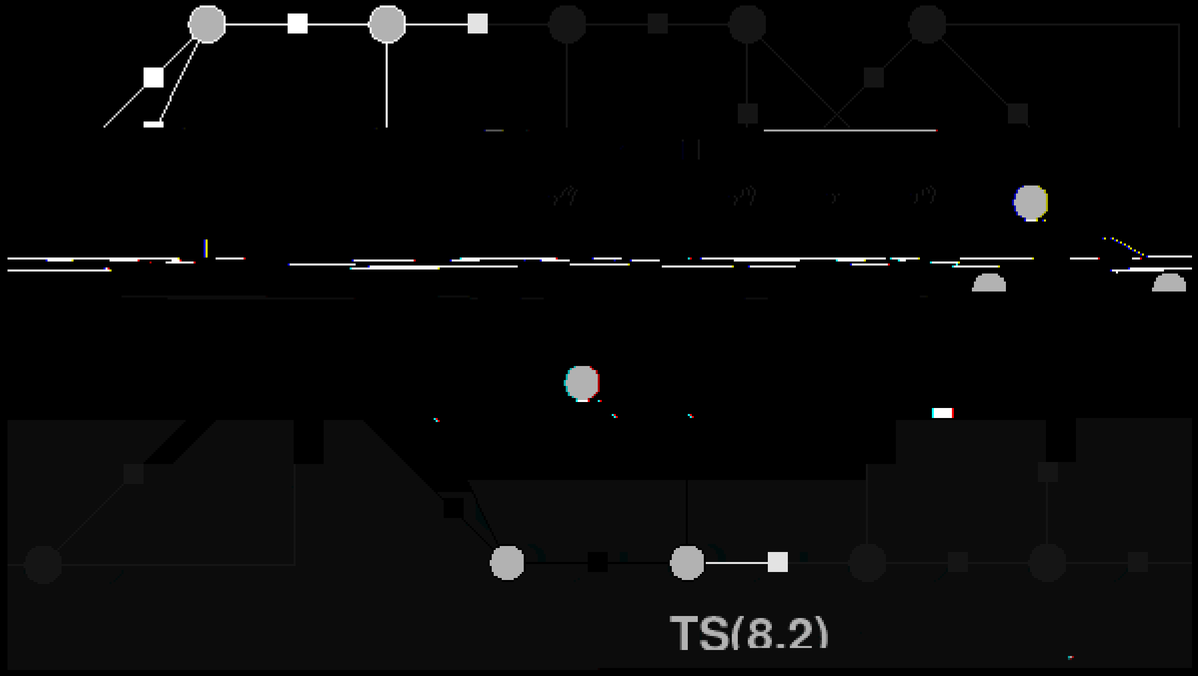
TS(6,4) 1-2-0-0-0

- 1023 weight-20 codewords belong in total
- Only 3 non

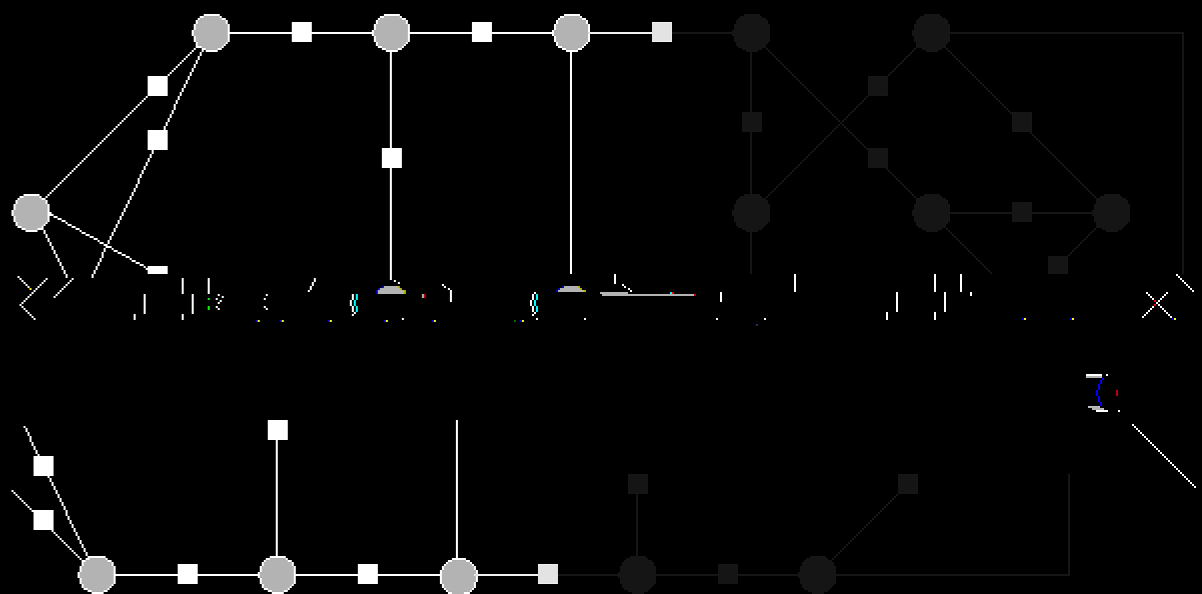
Codeword structure in Tanner code (2)



Codeword structure in Tanner code (3)

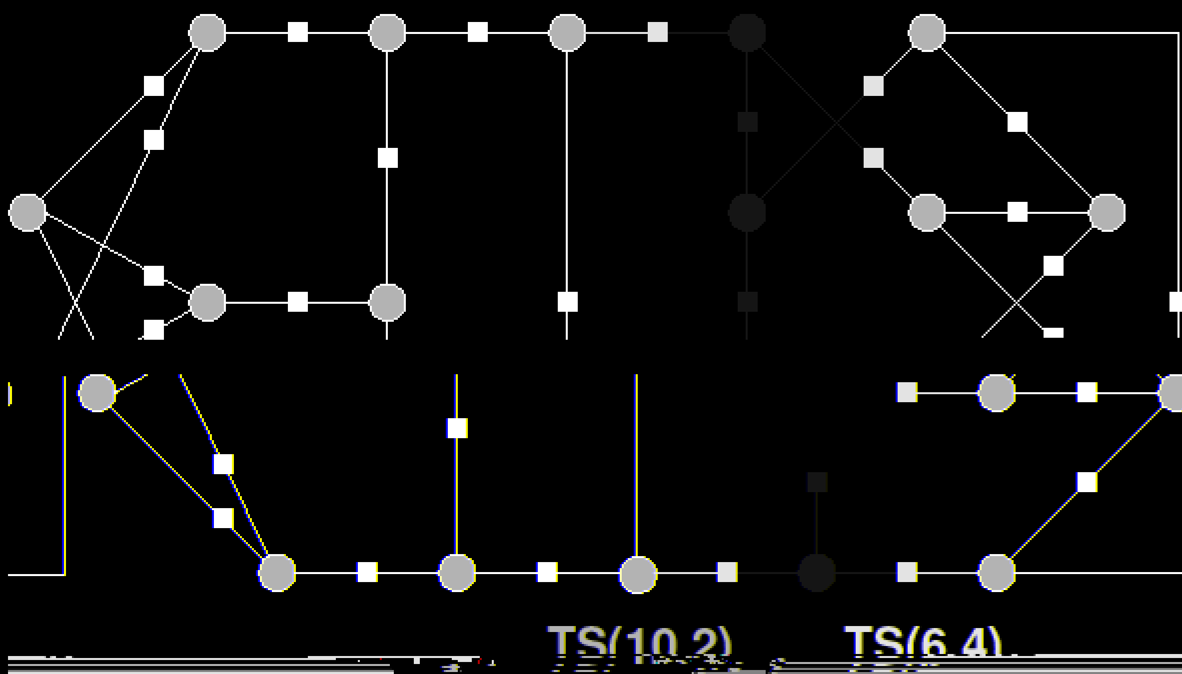


Codeword structure in Tanner code (4)

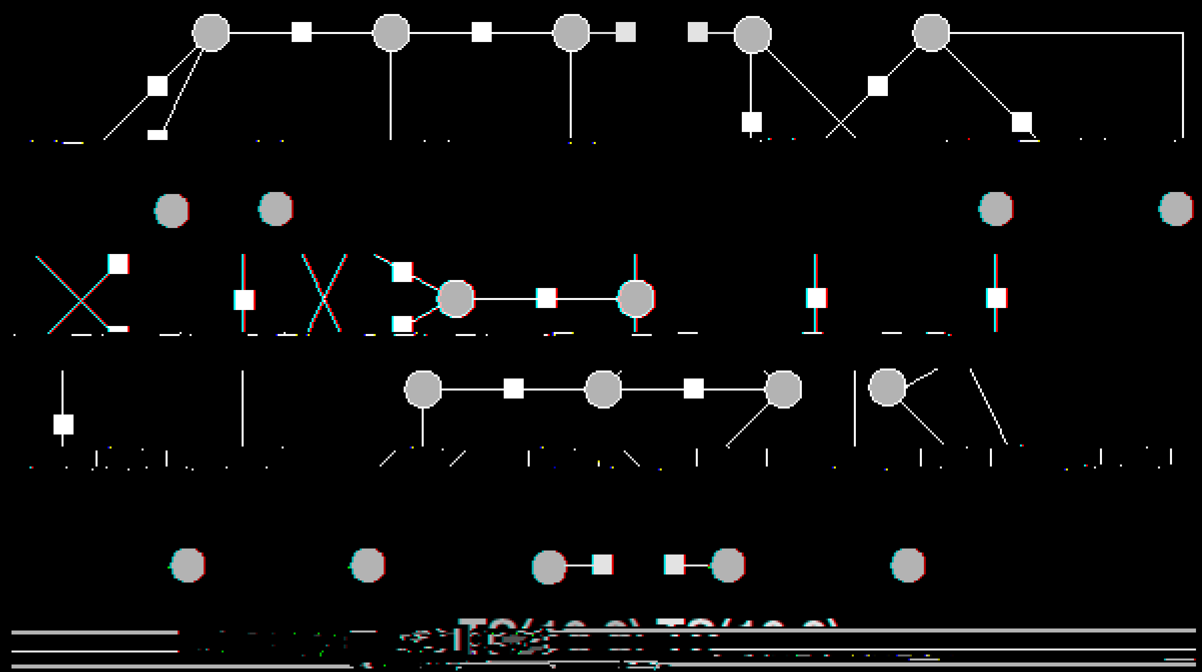


TS(10,2)

Codeword structure in Tanner code (5)



Codeword structure in Tanner code (6)



Searching for trapping sets

- Trapping sets are searched for in a way similar to how they have evolved in the Trapping Set Ontology.
- Since the induced subgraph of every trapping set contains at least a cycle, the search for trapping sets begins with enumerating cycles.
- After cycles are enumerated, they will be used in the search for bigger trapping sets.
- A bigger trapping set can be found in a Tanner graph by expanding a smaller trapping set.

Searching for trapping sets

- For example: Suppose that $\mathcal{T}_2$ is a direct successor of $\mathcal{T}_1$, and that all $\mathcal{T}_1$ trapping sets have been enumerated. In order to enumerate $\mathcal{T}_2$ trapping sets, we search for sets of variable nodes such that the union of such a set with a trapping set $\mathcal{T}_1$ form a $\mathcal{T}_2$ trapping set.

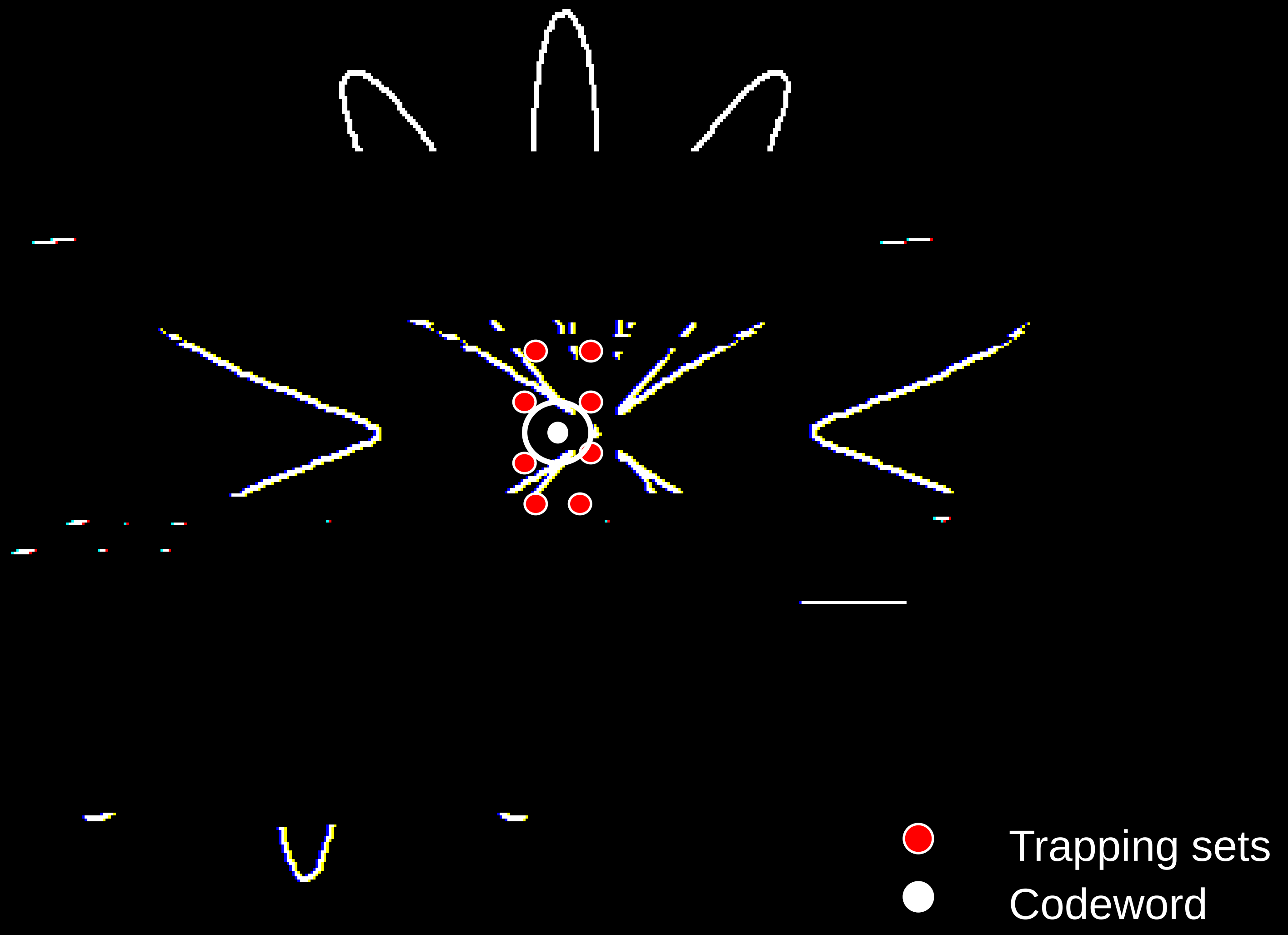
- The complexity of the search for trapping sets in the Tanner graph of a structured code can be greatly reduced by utilizing the structural property of its parity-check matrix.

Searching for Trapping Sets

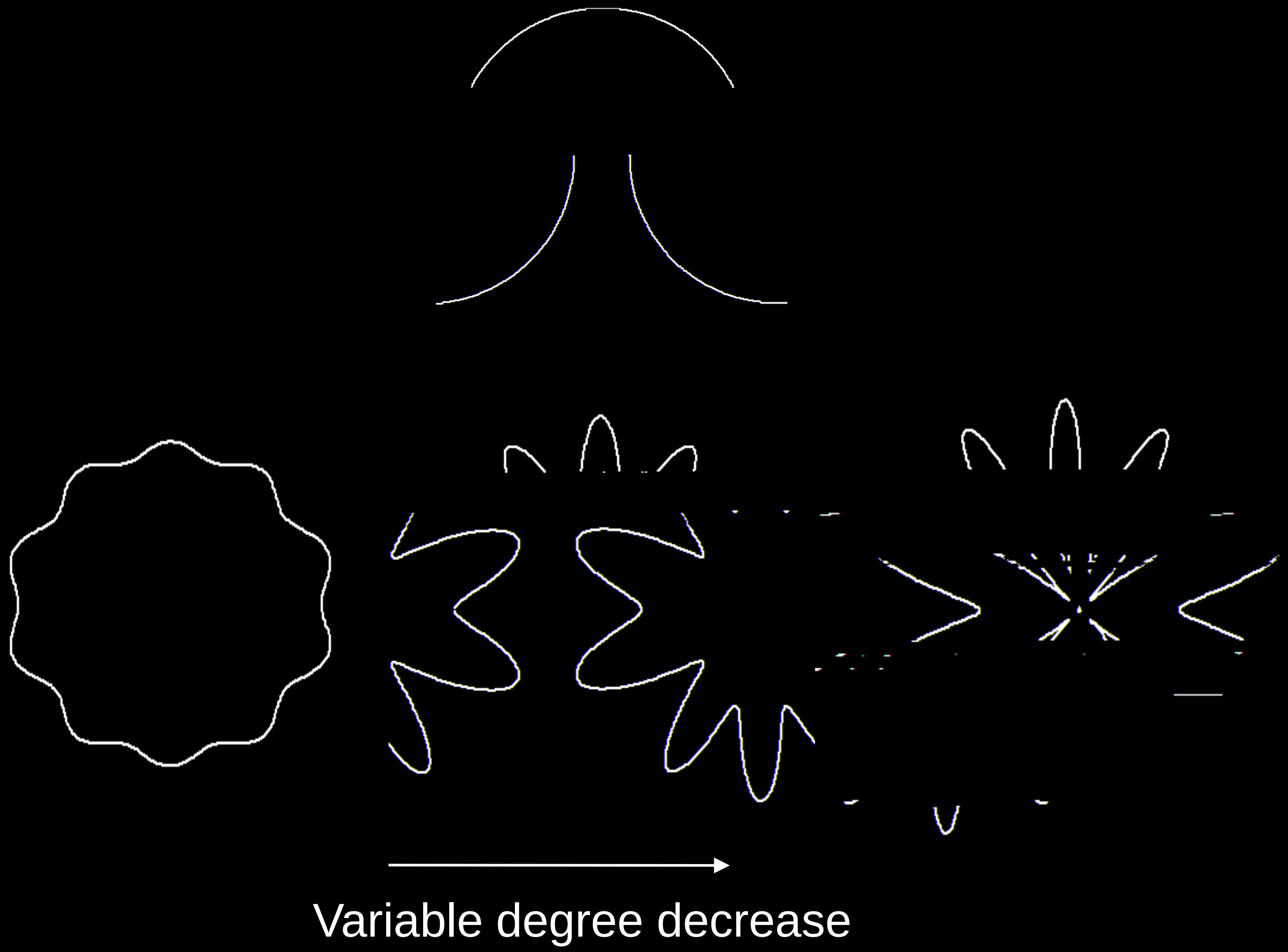
8-cycles	17066	1.362	0.109
10-cycles	183433		
(8, 2){1}	1166	0.160	0.002

How many errors can a column weight three code correct under iterative decoding?

Instantons and trapping sets



Failures of Iterative Decoders



The curious case of $d_v = 3$ codes

- Gallager showed that the minimum distance of ensembles of (d_v, d_c) regular LDPC codes with d_v 3 grows linearly with the code length
- This implies that under ML decoding, $d_v = 3$ codes <u>are capable</u> of correcting a number of errors linear in the code length
- Gallager also showed that under his algorithms A and B the bit error probability approaches zero whenever we operate below the threshold
- But, the correction of a linear fraction of errors was not shown

Other complications with d_v

Correcting fixed number of errors

- Bounded distance decoders (trivial)
  - A code with minimum distance $2t+1$ can correct t errors
- Iterative decoding on BEC (solved)
  - Can recover from t erasures if the size of minimum *stopping set* is at least $t+1$
- Iterative message passing decoding on BSC (unknown)
  - Error floor

$$\log(FER(\alpha)) \approx \log(c_i) + i\log(\alpha)$$

c_k - the number of configurations of received bits for which k channel error lead to a codeword (frame) error

Trapping sets - sufficient conditions

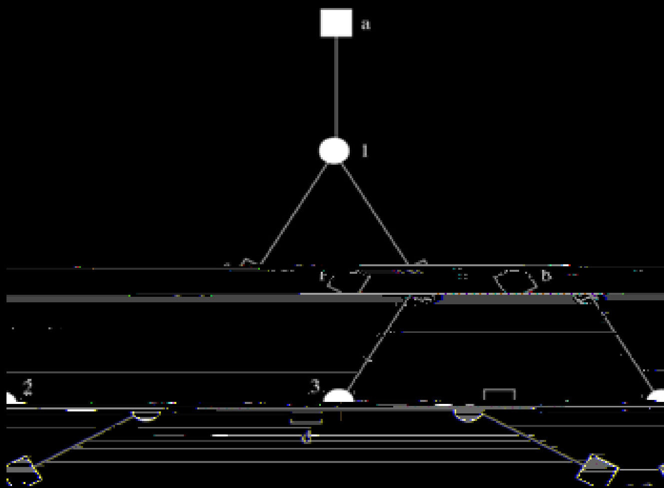
- *Theorem 1: Let C be a code in the ensemble of (3, ρ)* regular LDPC codes. Let be a subgraph induced by the set of variable nodes *T*. Let the checks in can be partitioned into two disjoint subsets: *E* consisting of checks with even degree, and *O* consisting of checks with odd degree. y is a fixed point if :

  (a) supp(y) = *T*,

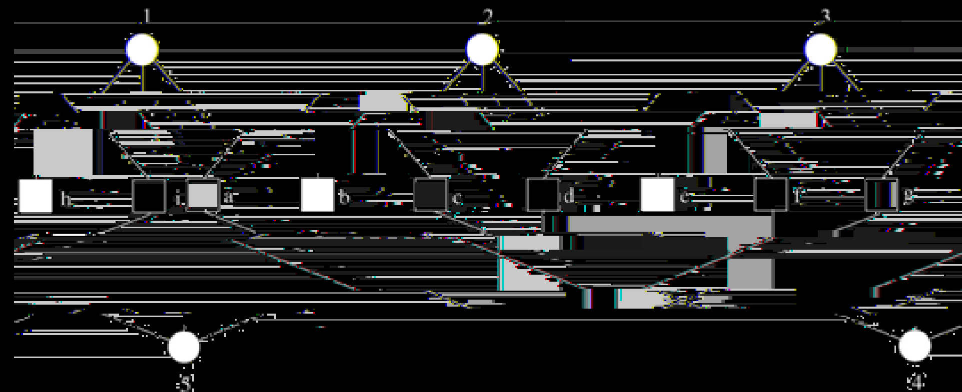
  (b) Every variable node in is connected to at least two checks in *E* ,

  (c) No two checks of *O* are connected to a variable node outside .

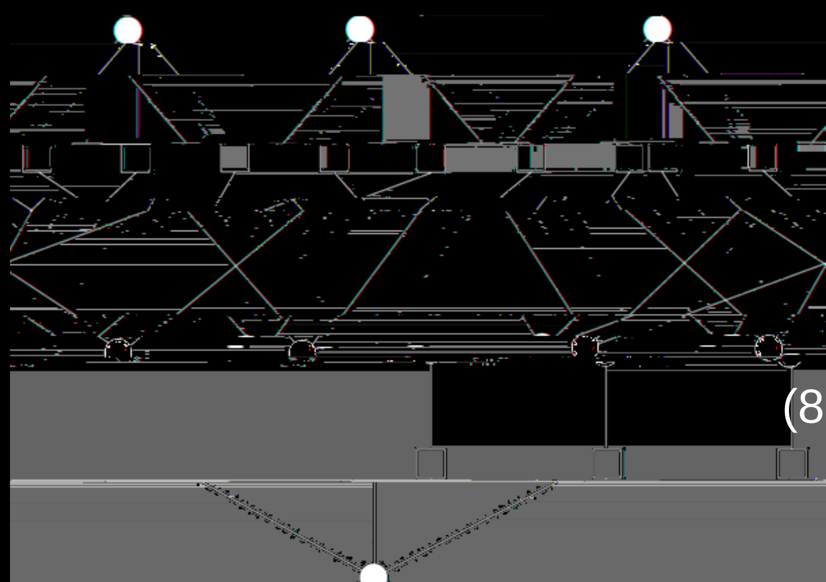
Trapping sets: examples



(3,3) trapping set



(5,3) trapping set



(8,0) Trapping Set

The upper bounds

- *Theorem 2:* Let *C* be an *(n, 3, ρ)* regular LDPC code with girth *g.* Then:
  - If g = 4, then C has at least one FS of size 2 or 3.
  - If g = 6, then C has least one FS of size 3 or 4.
  - If *g* = 8, then C has at least one FS of size 4

Consequences

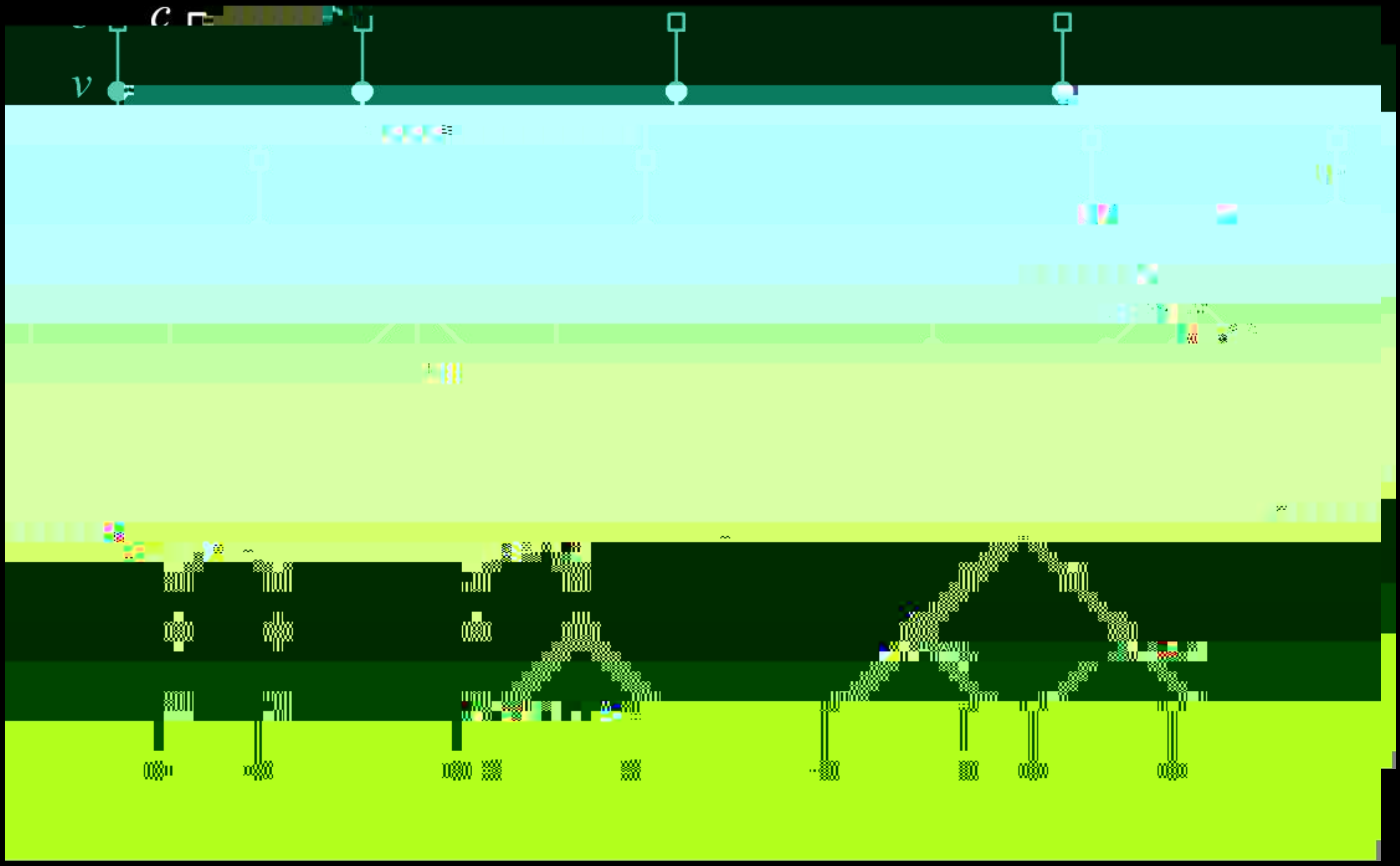
- For column weight three codes, the weight of correctable error patterns under Gallager A algorithm grows only linearly with girth

- For any $\alpha > 0$ and sufficiently large block lengths n, no code in the $C^n(3, \rho)$ ensemble can correct all αn errors under Gallager A algorithm

The lower bound lemmas

- *Theorem 3*: An *(n, 3, ρ)* code with girth g 10 can correct all error patterns of weight $g/2$-1 or less in g/2 iterations of the Gallager A algorithm.
- Equivalently, there are no trapping sets with critical number less than $g/2$.
- Proof: Finding, for a particular choice of *k*, all configurations of $g/2$-1 or less bad variable nodes which do not converge in $k+1$ iterations and then prove thatot conve

Bad configurations ($k = 1$ and $k = 2$)



Bad configurations ($k=3$)

Cage Graphs

- A (d, g)-*cage graph, G(d, g), is a d-regular* graph with girth *g* having the minimum possible number of nodes.

- *Theorem 10: Let C be an LDPC code with -left regular* Tanner graph G and girth 2gNj. Let T (, 2gNj) denote the size of smallest possible potential trapping set of C for the bit flipping algorithm. Then,

$$|T(\; , 2gNj)| = n_c(\; /2 \; , gNj)$$

- *Theorem 11: There exists a code C with γ-left regular* Tanner graph of girth 2gNj which fails to correct $n_c(\; /2 \; , gNj)$ errors.

Comments

- For =*3* and =*4*, the above bound is tight.
- Observe that for d=2, the Moore bound is $n_0(d, g)=g$ and that a cycle of length 2*g* with *g* variable nodes is always a potential trapping set.
- For a code with =3 or 4, and Tanner graph of girth greater than eight, a cycle of the smallest length is always a trapping set.

Refined Expansion

- *Theorem : An LDPC code with column-weight four and girth six can correct three errors in four* iterations of message-passing decoding if and only if the conditions, 4 Ÿ 11, 5 Ÿ 12, 6 Ÿ 14, 7 Ÿ 16 and 8 Ÿ 18 are satisfied.

- y Ÿ z means that any set of y variable nodes has at least z neighbors

Summary

- Introduced LDPC codes, Tanner graphs, iterative decoders
- For BEC showed how to analyze failures using the

Extra slides

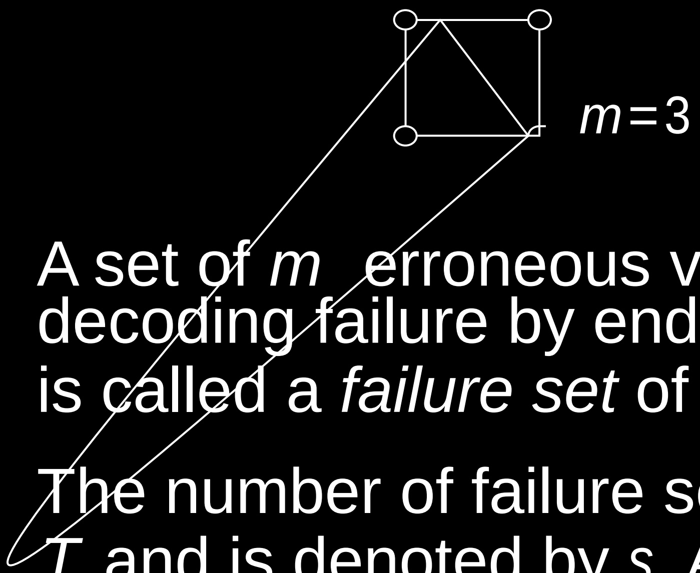
Error floor

Critical number

- With every trapping set *T* is associated a *critical number* *m* (or *m(*T*)*)

Strength of a trapping set

- Not all configurations of *m* errors in a trapping set result in a decoding failure.
  - (5, 3) TS: *m*=3, only one configuration of three errors leads to a decoding failure.
  - (4, 2) TS: *m*=3m all the four combinations of three errors lead to decoding failure.



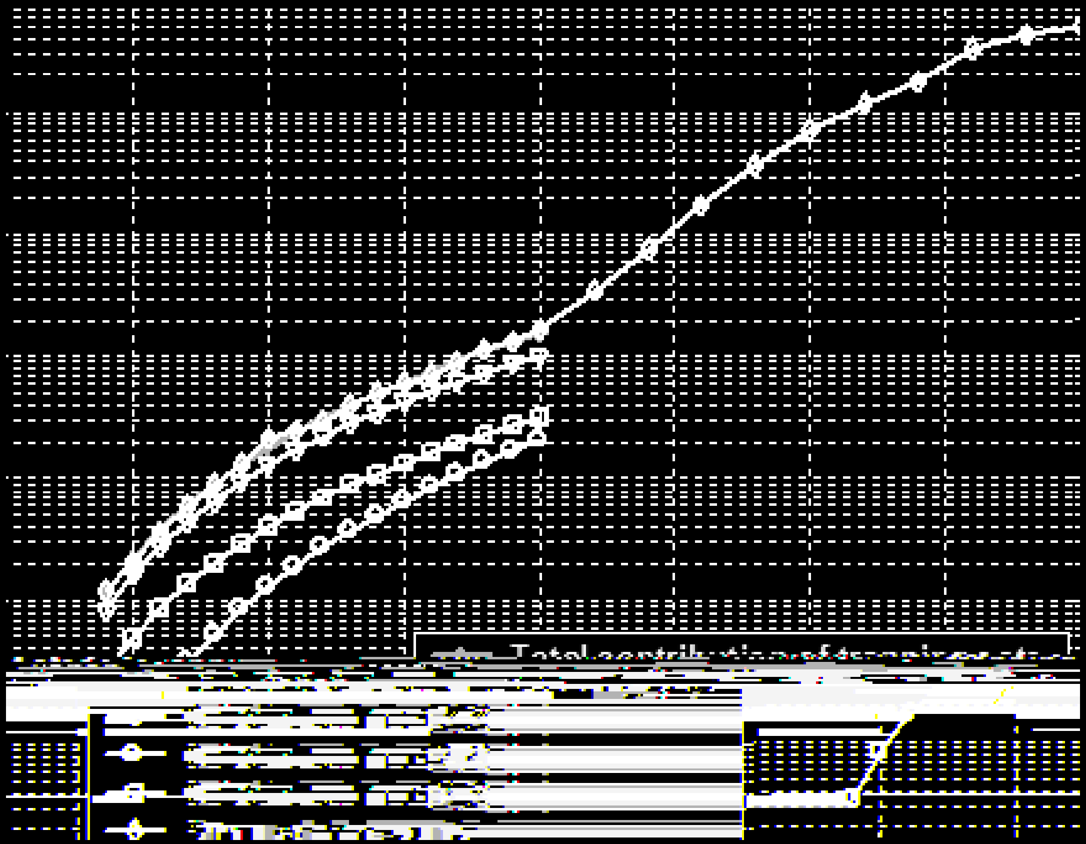
- A set of *m* erroneous variable nodes which leads to a decoding failure by ending in a trapping set of class *X* is called a *failure set* of *X* .
- The number of failure sets of *T* is called the *strength of T* and is denoted by *s*. A class *X* has *s*|*X*| failure sets.

Approximation

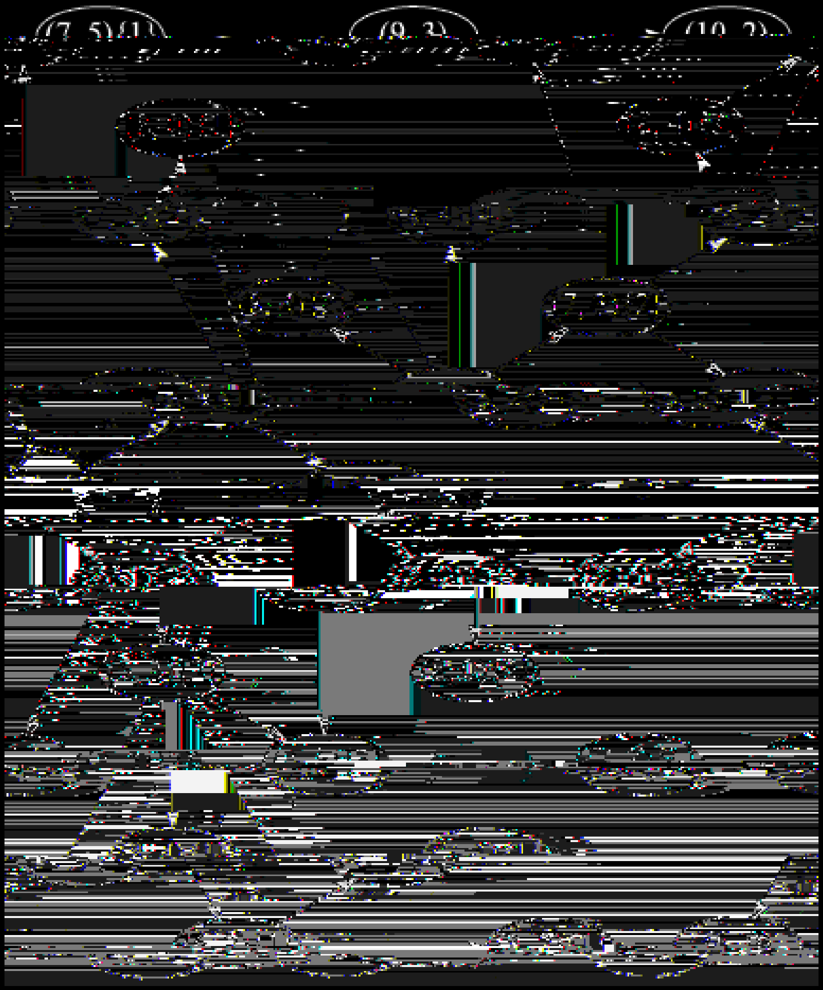
- The contribution of each class of trapping set:

- is the probability that a given set of *m* variable nodes is a failure set of class c

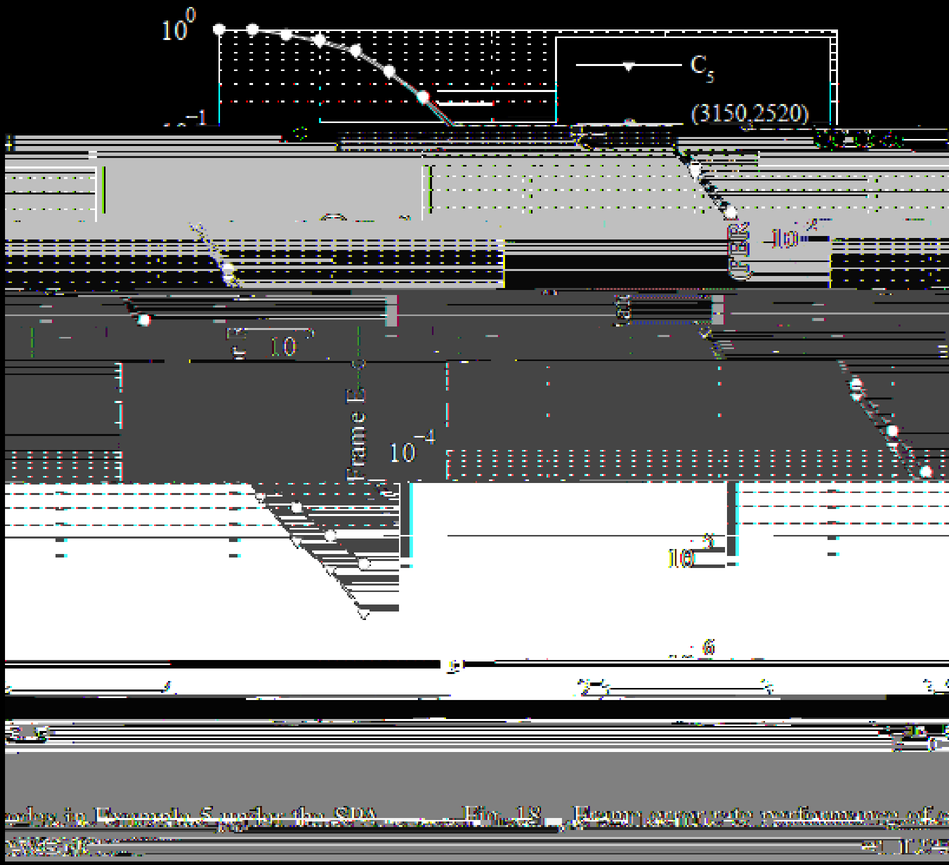
FER contribution of different error patterns



Designing better codes using trapping sets



Quasi-cyclic codes



Designing better decoders

(3 bits)

Multi-bit iterative decoders

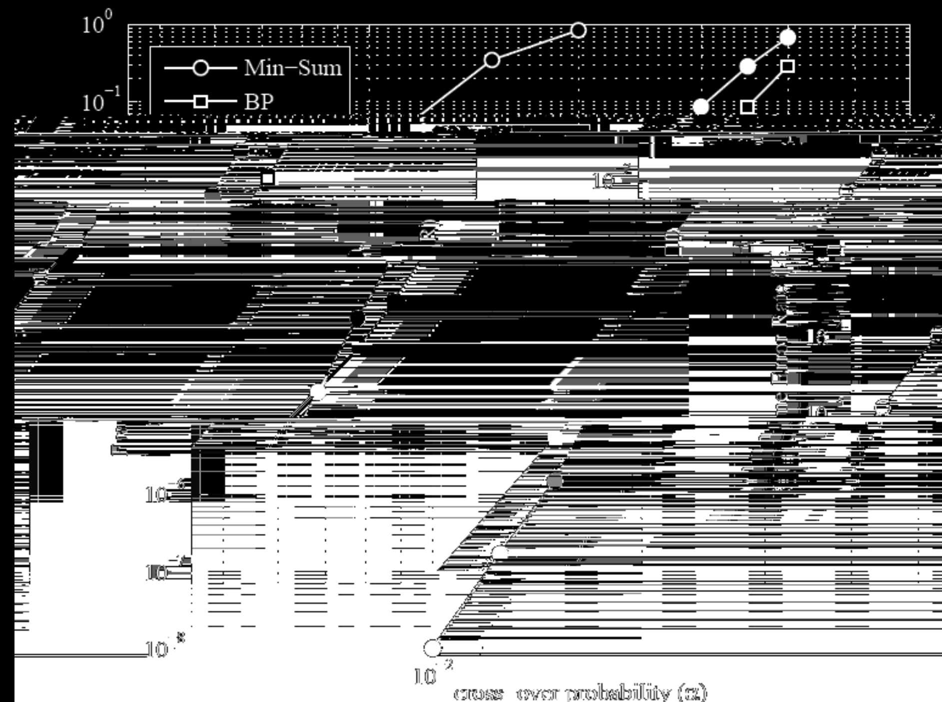
- Gallager-like algorithms, but the messages are binary vectors of length m, $m>1$.
-

3-bit decoder that surpasses BP

m_1	m_2	r	m

m_1	m_2	r	m

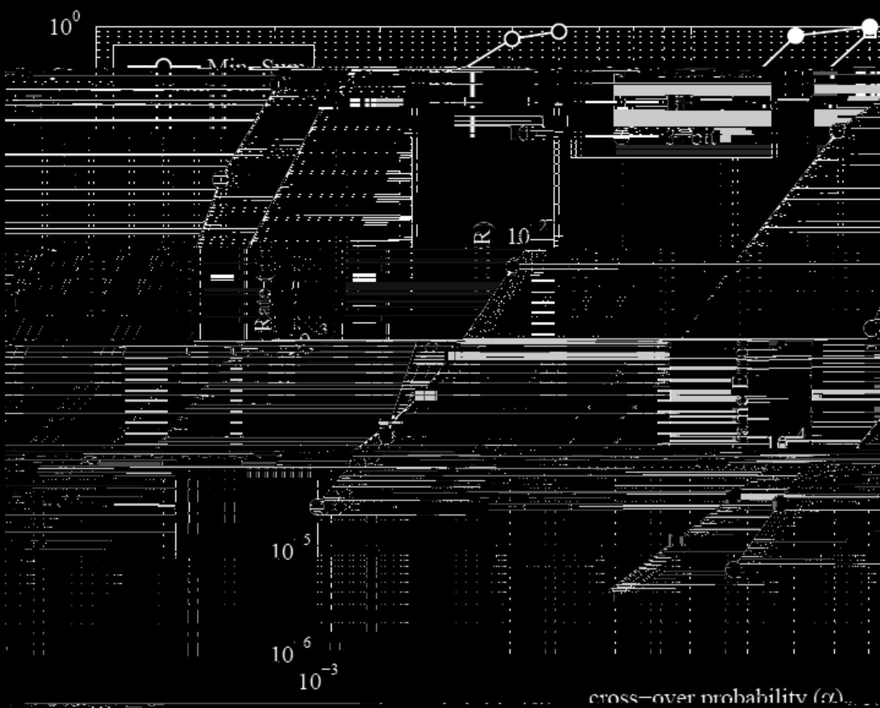
Numerical results

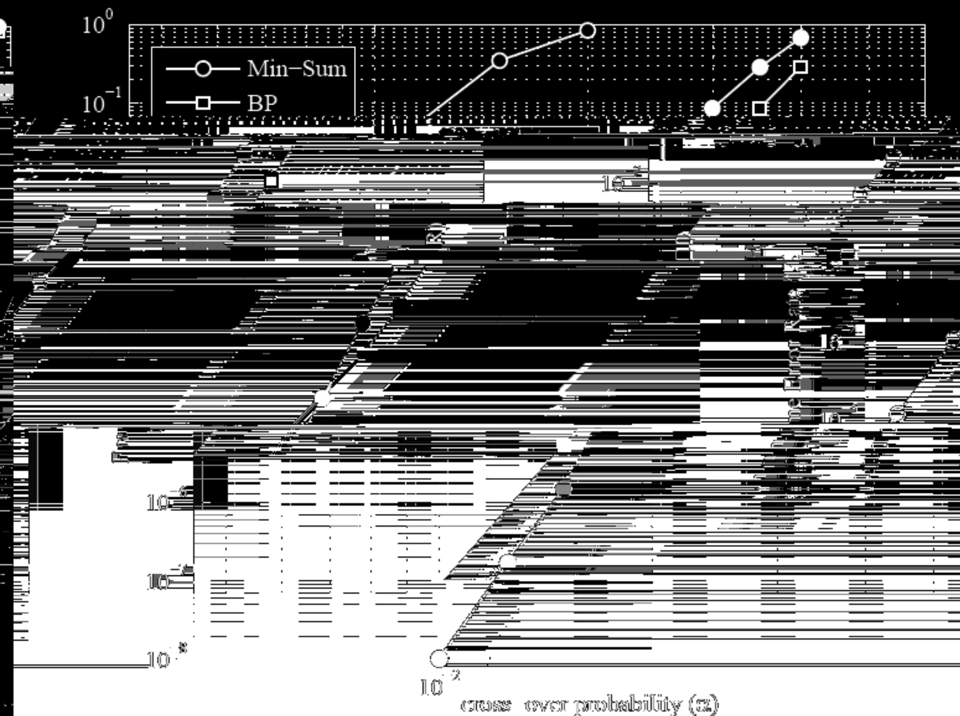


*N=155, R=0.4,*Tanner code

N=768, R=0.75, Quasicyclic code

Numerical results





*N=4085, R=0.82,*MacKay code

N=1503, R=0.668, Quasicyclic code

Note: Notice the diffference in slope of FER

Extra slides

Trapping set as decoding failures

- The all zero codeword is transmitted.
- The decoder performs *D* iterations.
  - $\boldsymbol{y} = (y_1\ y_2\ .\ .\ .\ y_n)$ - decoder input
  - $\boldsymbol{x}^l$, l≤ D -the decoder output vector at the *l*-th iteration
- A variable node *v* is *eventually correct if there exists a positive integer* d *such that for all l > d, v* supp($\boldsymbol{x}^l$).
- *A decoder failure is said to occur if there* does not exits *l* ≤ *D* such that supp(x^l) = .
  - *T(**y**)* – a nonempty set of variable nodes that are not eventually correct
  - G - subgraph induced by *T(**y**), C(*G) =*E O* (even and odd degree check nodes in)
  - T(***y***) is an (*a,b*) trapping set, where a = |T(**y**)|, *b* =|O|