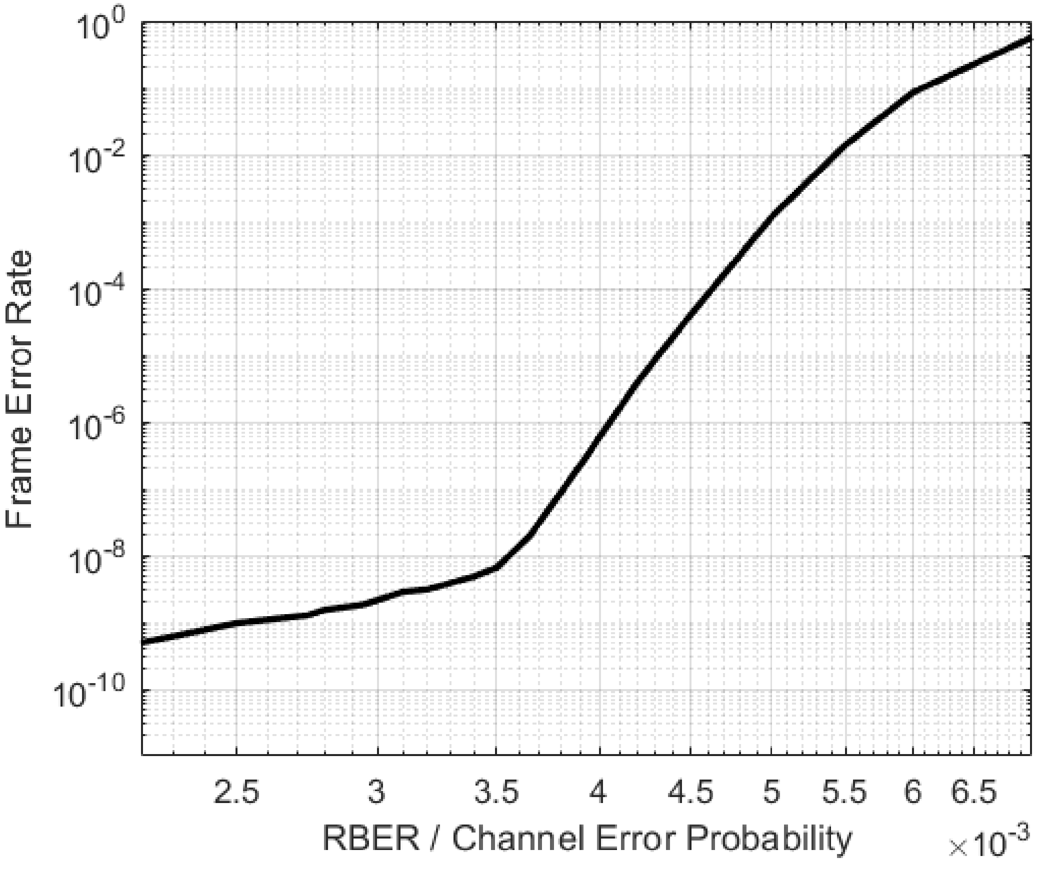
A graph expansion-contraction method for estimating error floors of LDPC codes

Bane Vasić [†‡] and David Declercq[†]

[‡] Error Correction Coding Laboratory
Department of ECE, University of Arizona
[†] Codelucida

Error floor

- An abrupt degradation of FER at low RBER caused by a failure of an iterative decoder to converge to a codeword

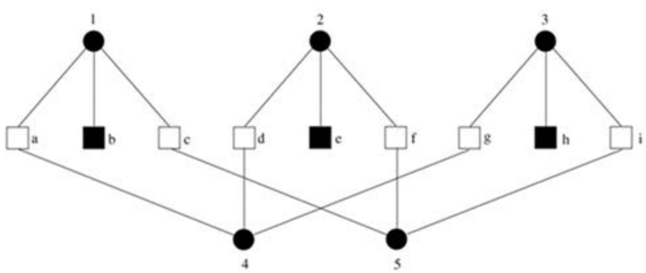


Codelucida

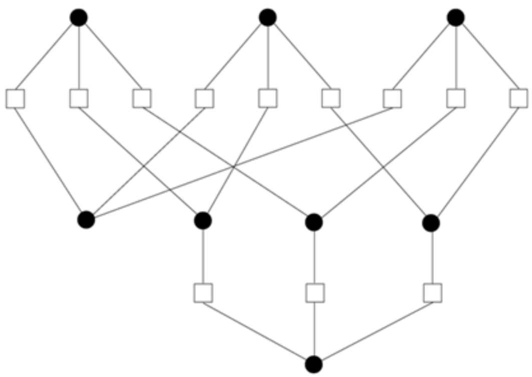
THE UNIVERSITY OF ARIZONA. TUCSON ARIZONA

Trapping sets

- Error floor is attributed to dense subgraphs present in the Tanner graph – trapping sets
- <u>An (A, B) trapping set</u>: a set of not eventually correct variabe nodes of size A, inducing a subgraph of the B odd degree check nodes.

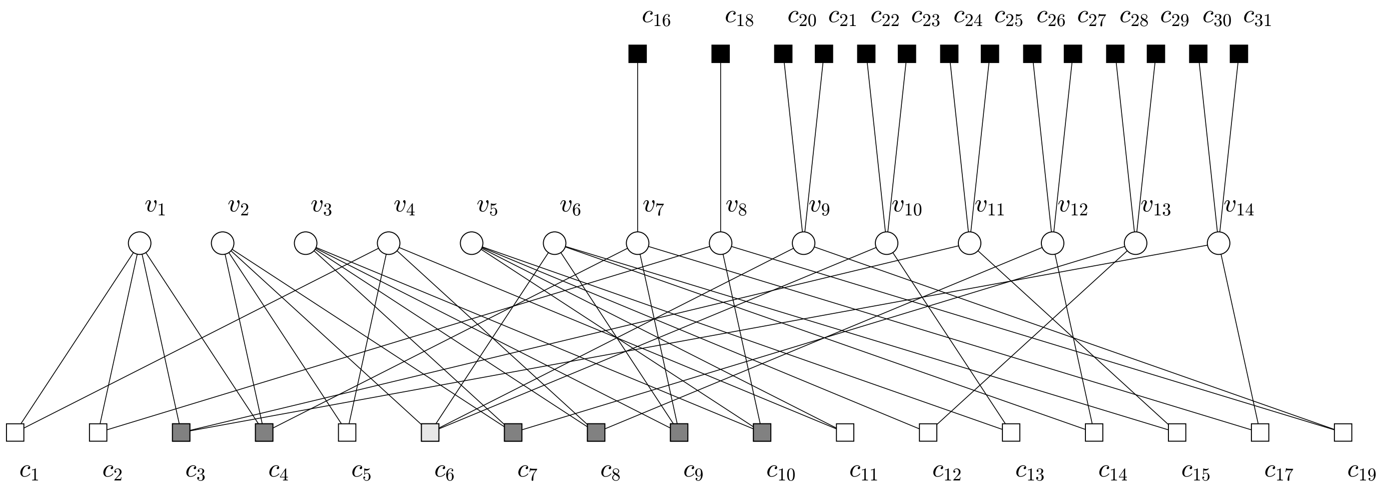


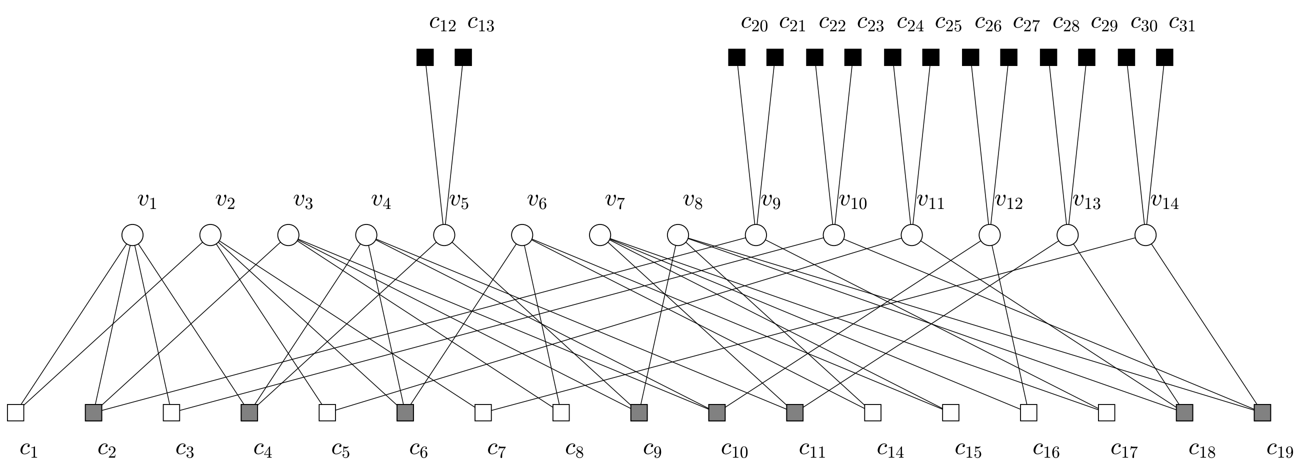
(5,3) trapping set



(8,0) Trapping set

Some large trapping sets



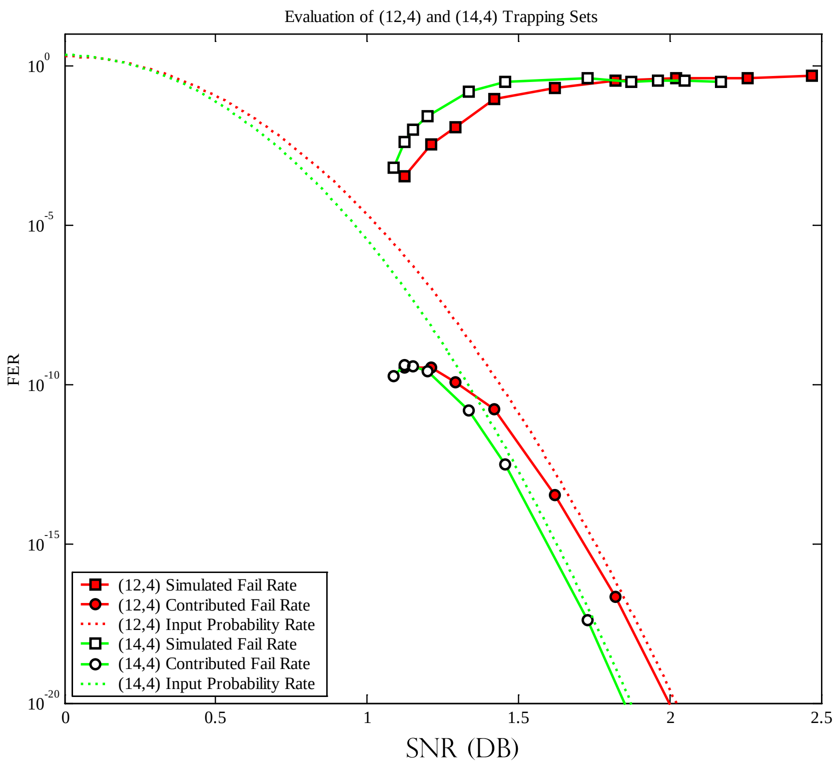


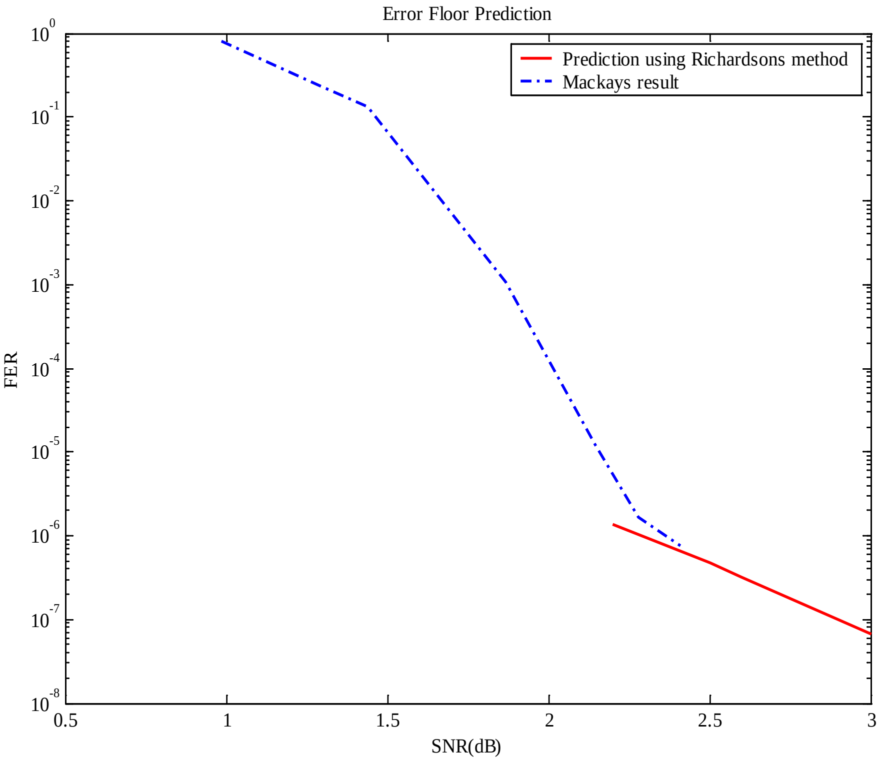
Codelucida

THE UNIVERSITY OF ARIZONA.
TUCSON ARIZONA

FER estimation using importance sampling

- Tom Richardson (Allerton 2003)
  - An experimental evidence of error floors of LDPC codes, and makes a connection with trapping set
  - Introduced the importance sampling to estimate error floor





Codelucida

THE UNIVERSITY OF ARIZONA. TUCSON ARIZONA

Importance sampling

- Input:
  - Tanner graph G and the decoding algorithm $\mathcal{D}$
  - Collection of subgraphs $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_T$ that are believed to be harmful to $\mathcal{D}$
- Algorithm:
  - Find positions of variable nodes of each of the harmful subgraphs in G (sensitive variable nodes)
  - In Monte-Carlo simulations, corrupt sensitive variable nodes in G, run $\mathcal{D}$, and record if an error patterns that lead to failure of $\mathcal{D}$
  - Obtain the contributions $\mathrm{FER}_{\mathcal{T}_1}, \mathrm{FER}_{\mathcal{T}_2}, \ldots, \mathrm{FER}_{\mathcal{T}_T}$ of each subgraph to the FER and reweight them by occurrence frequencies of subgraphs $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_T$ in G

$$FER = \sum_{\mathcal{T}} w_{\mathcal{T}} \mathrm{FER}_{\mathcal{T}}$$

Harmfulness

- Major flaw of importance sampling – "subgraphs that are believed to be harmful"
- Vasić (Allerton 2005, ICC 2006)
  - Harmfulness defined based on uncorrectable error patterns
  - Defined the critical number C and strength S as a measure of harmfulness of a trapping set
  - No assumptions on a trapping set topology were made!
  - A simple formula for calculation error floor from harmfulness of trapping sets
  - Sufficient conditions for failure of Gallager B and bit-flipping algorithms on column weight three codes (γ=3)
- Vasić (Allerton 2009): trapping set ontology - a database of topological relationship of trapping sets of simple decoders for column weight-three code

Harmfulness for stronger decoding rules

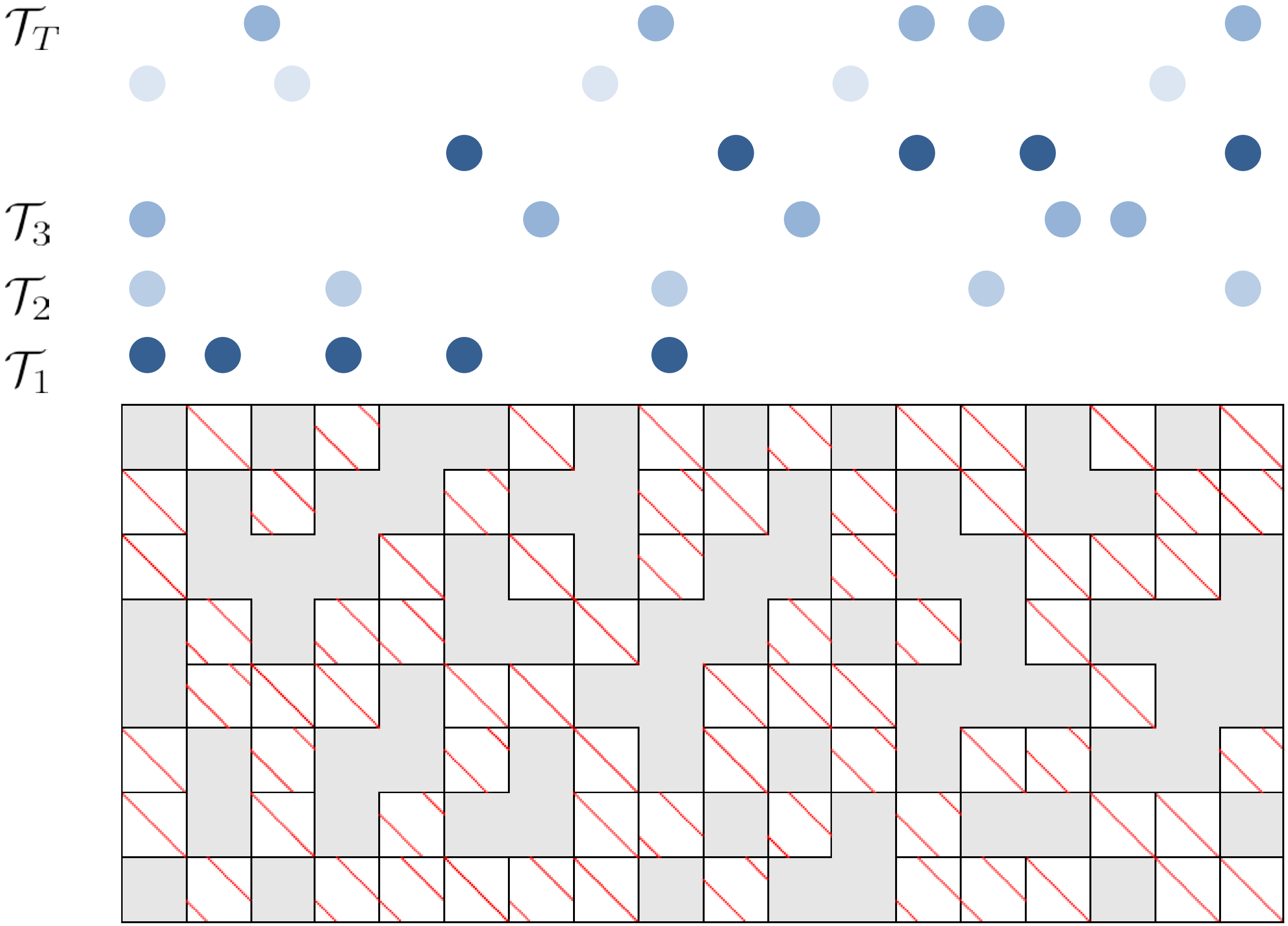
- Many papers on combinatorial characterization and search of trapping sets
  - Measuring harmfulness based solely on the value of (A,B) parameters is and their relation is wrong!
  - No assumptions on a trapping set topology must be made!
  - The only theory-supported indicators of harmfulness of a trapping set are its expansion (or "density") and cycle profile
- Whether a trapping set (subgraph) is harmful depends on a **decoder** and its **neighborhood** in the Tanner graph
  - For simple decoders, trapping sets can be treated isolated from the rest of the graph
  - For decoders of interest (offset min-sum, FAID), an isolated trapping set is not sufficient to predict its impact on error floor

FER of higher column-weight codes

- Even for γ=4, the number of nonisomorphic dense subraphs is enormous
- It is impractical to create a trapping set ontology as we did for for γ=3, instead harmfull subraphs must be searched for in the specific Tanner graph
- For accuracy of FER estimation, we cannot afford to miss any harmful trapping set, thus verification of harmfulness must be exhaustive

- But which graphs are harmful?
- How do we estimate error floor based on harmfulness?

Trapping set positions in the QC-LDPC code

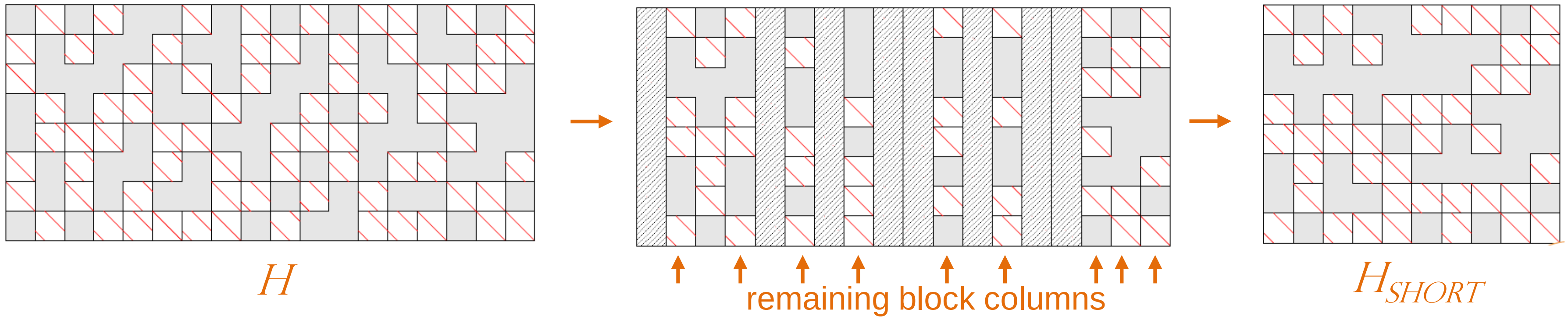


Codelucida

The University of Arizona
Tucson Arizona

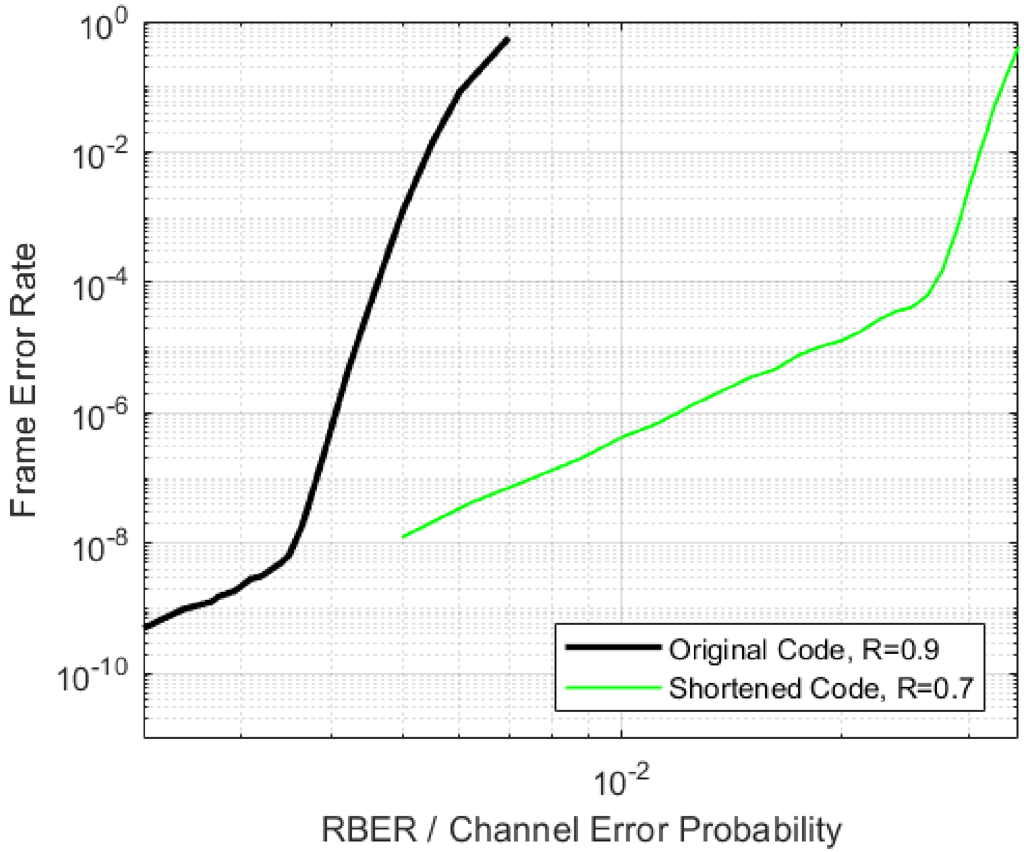
Importance sampling through code shortening

- Create a shortened code H_{SHORT} which contains the same most harmful trapping sets as the original code H, and run Monte Carlo simulations on H_{SHORT} to detect its error floor.



Decoding on the shortened code

- If the decoder fails on the same structures as in H, the error floor will have the same slope, but since H_{SHORT} has lower rate, the error floor will appear at a higher FER, resulting in computational savings



Reasons for accuracy

- Each trapping set is not treated as an isolated graph but in its “natural surrounding”
- Message from the variables outside the trapping set are realistic (not considered to be saturated)

Code shortening constrains

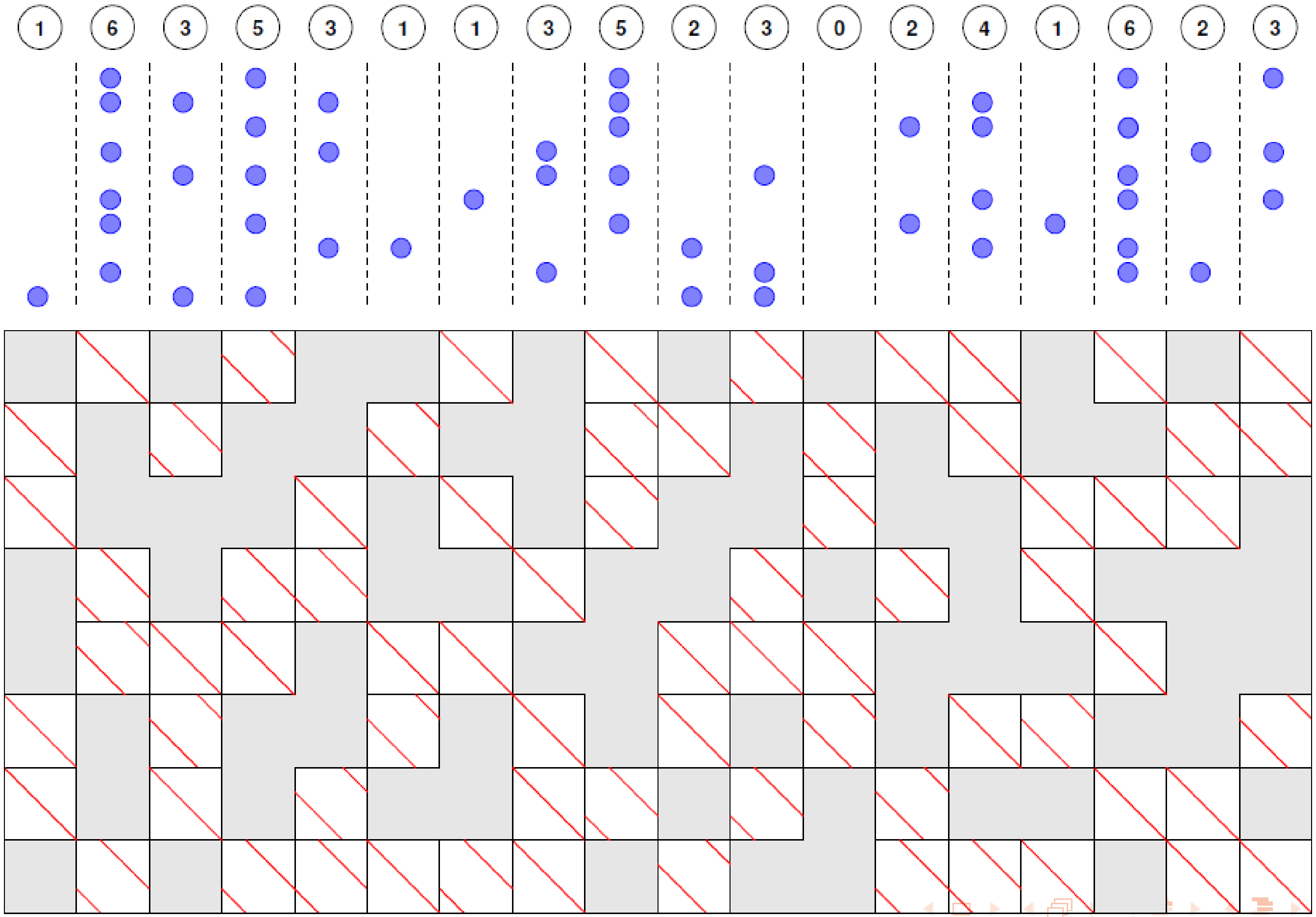
- Choosing properly which block columns to keep is critical for the efficiency
  - the less shortening - the easier to ensure that the harmful trapping sets of H will remain in H_{SHORT}, but smaller computational saving
  - with too much shortening, there is a chance that H_{SHORT} does not contain the harmful trapping sets any more, resulting in an erroneous prediction of the error floor

Optimization problem

- Design the shortest, worst possible shortened code
- Minimize the number of block columns kept, while still ensuring that the most harmful trapping sets are present in H_{SHORT}
- This optimization problem is closely related to the well studied weapon-target assignment problem and the hypergraph demand matching problem

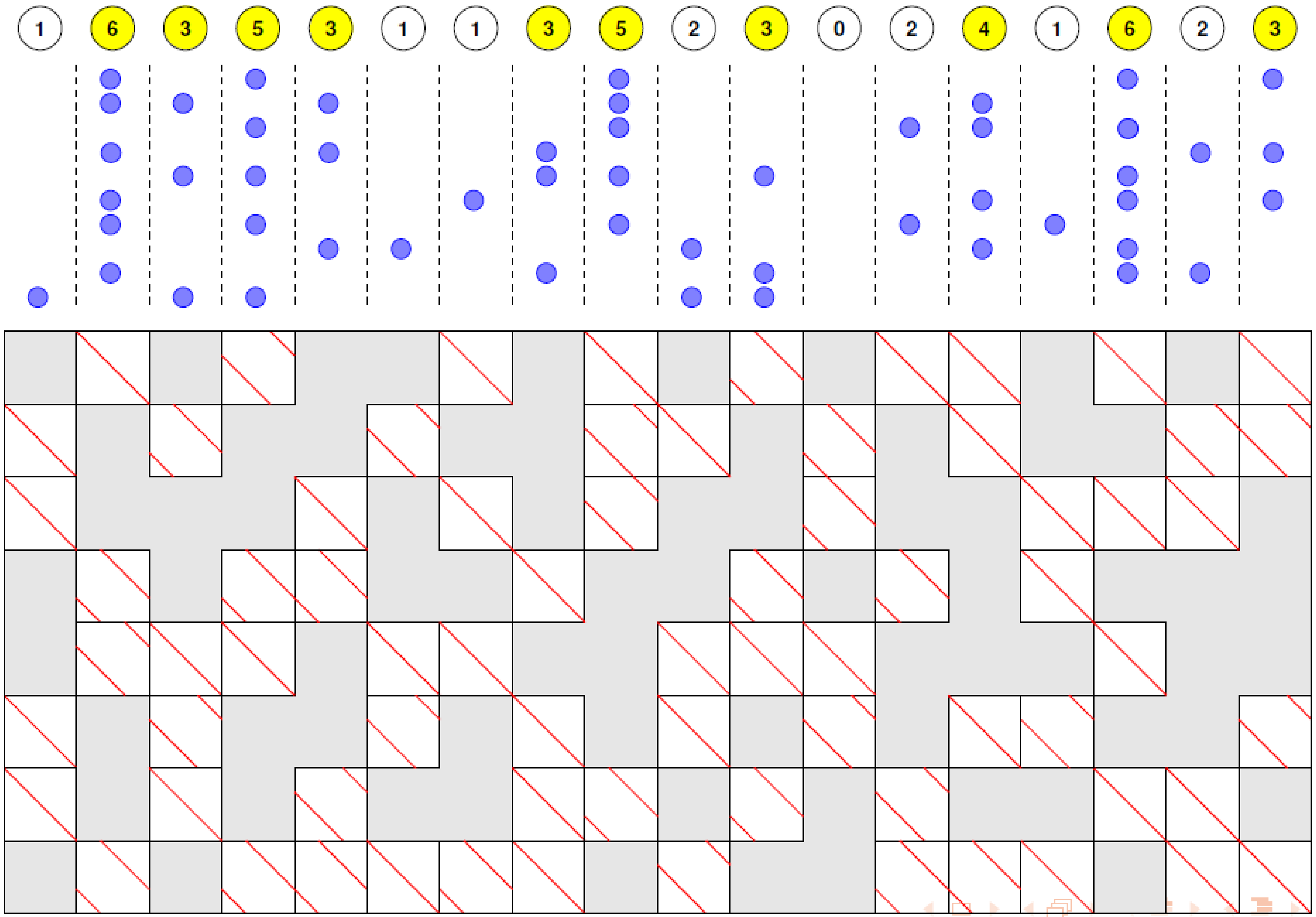
Step 1

- The l-th block harmfulness = sum of harfmulnesses of all trapping sets having a variable node in it



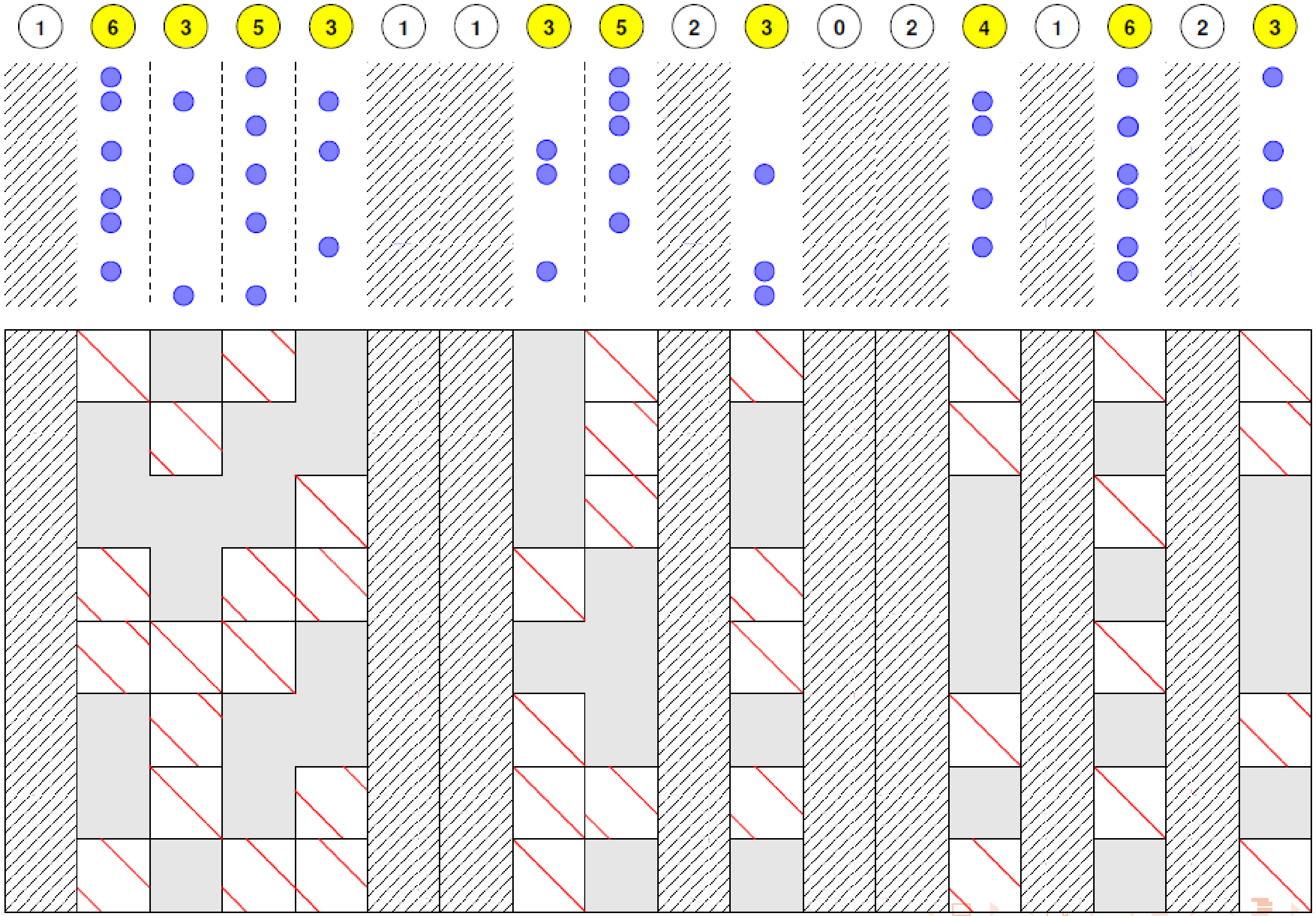
Step 2

- Select the block-columns which have total harmfulness weight $\mathbf{H}$ greater than a threshold T



Step 3

- Build a shortened version of the code, with $N_{SHORT}=9$ block-columns

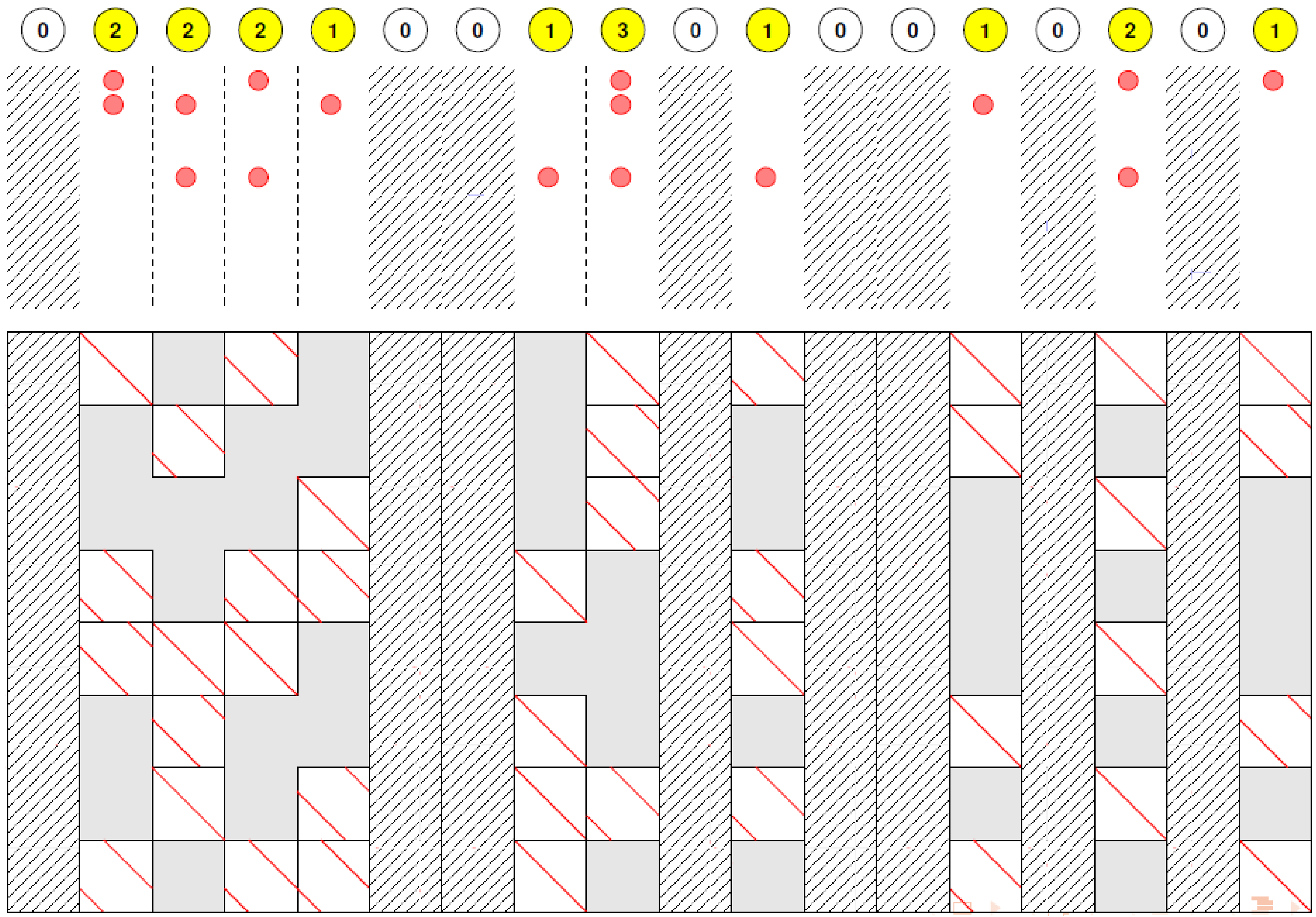


Codelucida

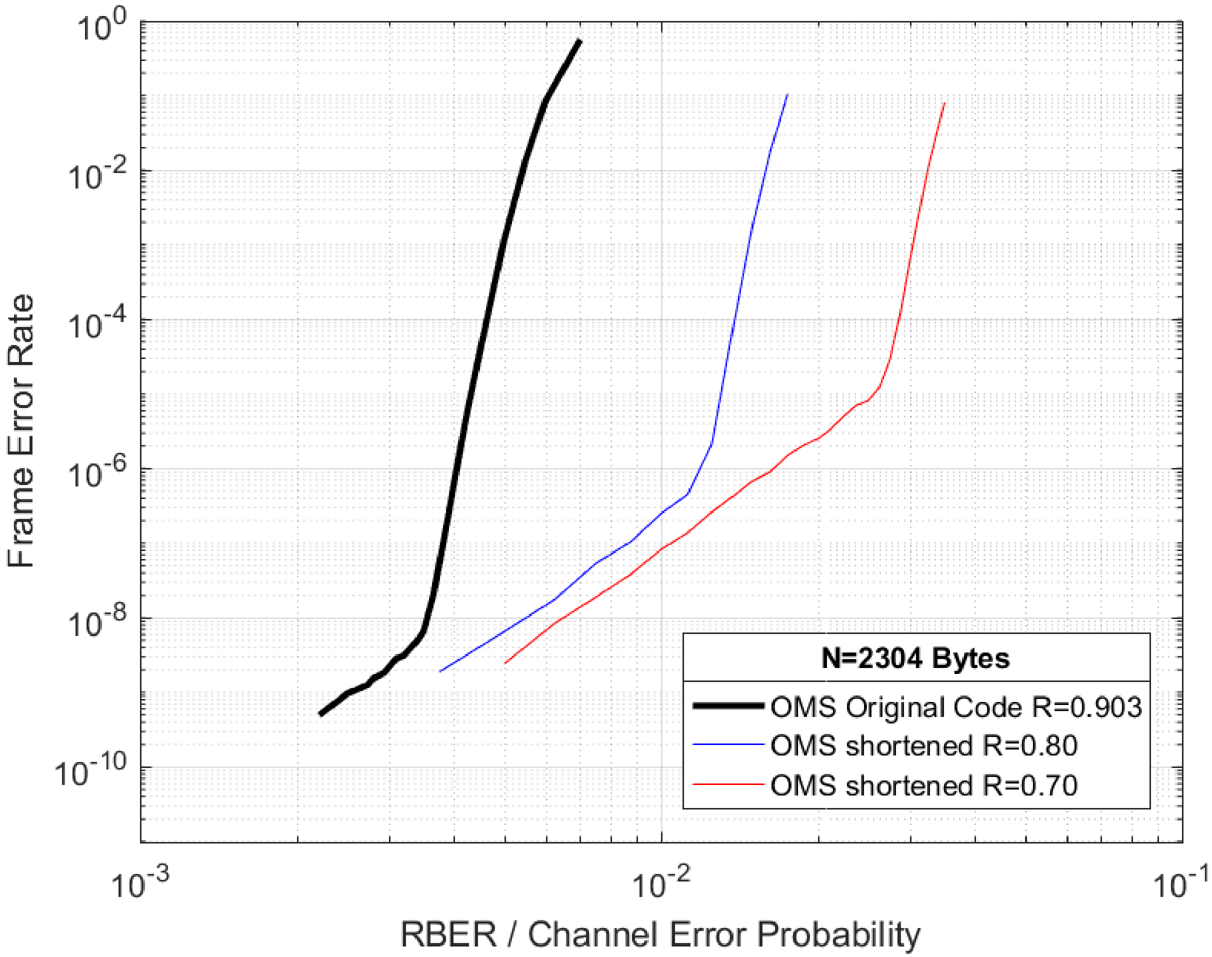
THE UNIVERSITY OF ARIZONA.
TUCSON ARIZONA

Step 4

- Correction factor - the ratio between remaining harmfulness weight and total harmfulness weight.



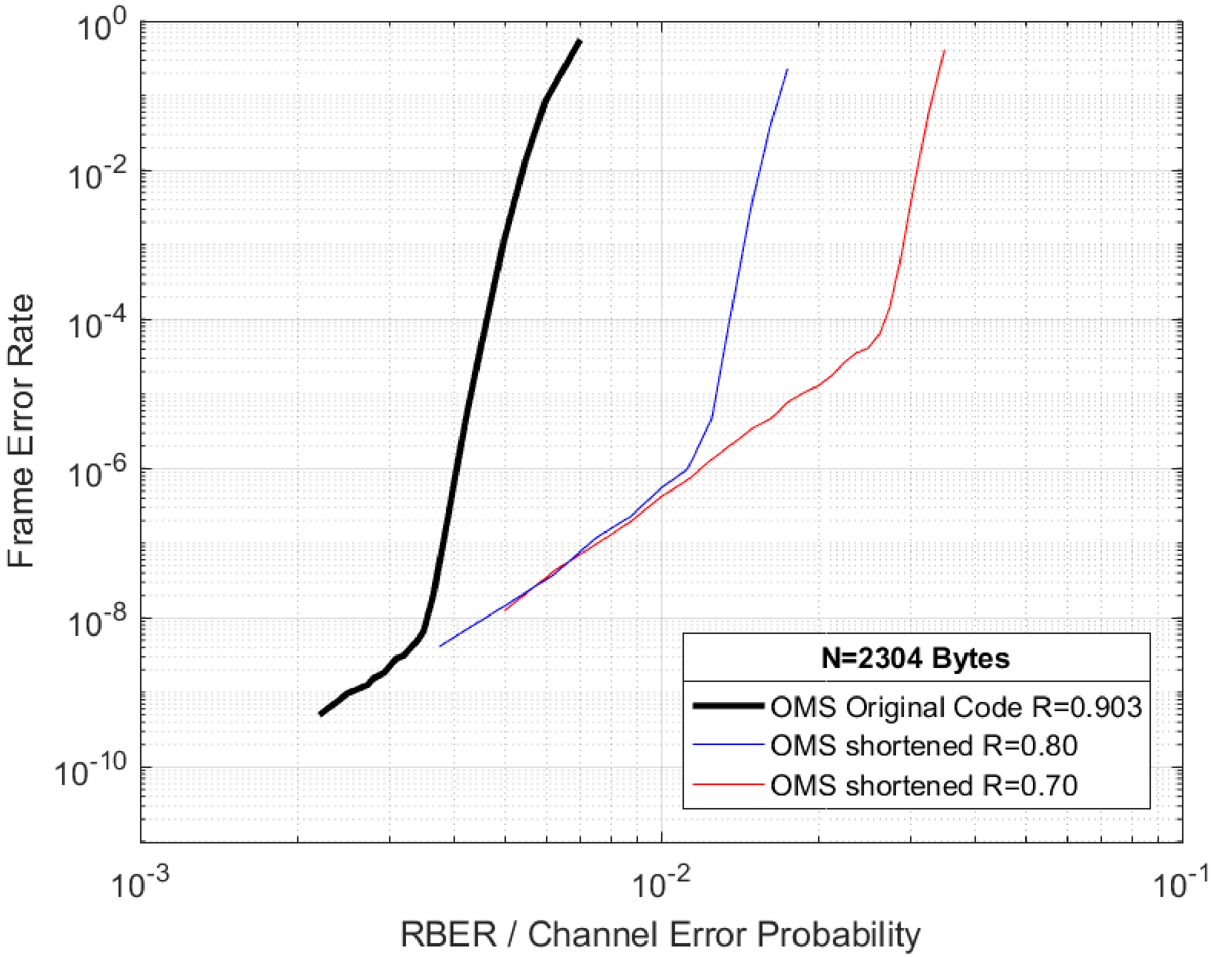
Direct simulation of the shortened codes



Codelucida

THE UNIVERSITY OF ARIZONA
TUCSON ARIZONA

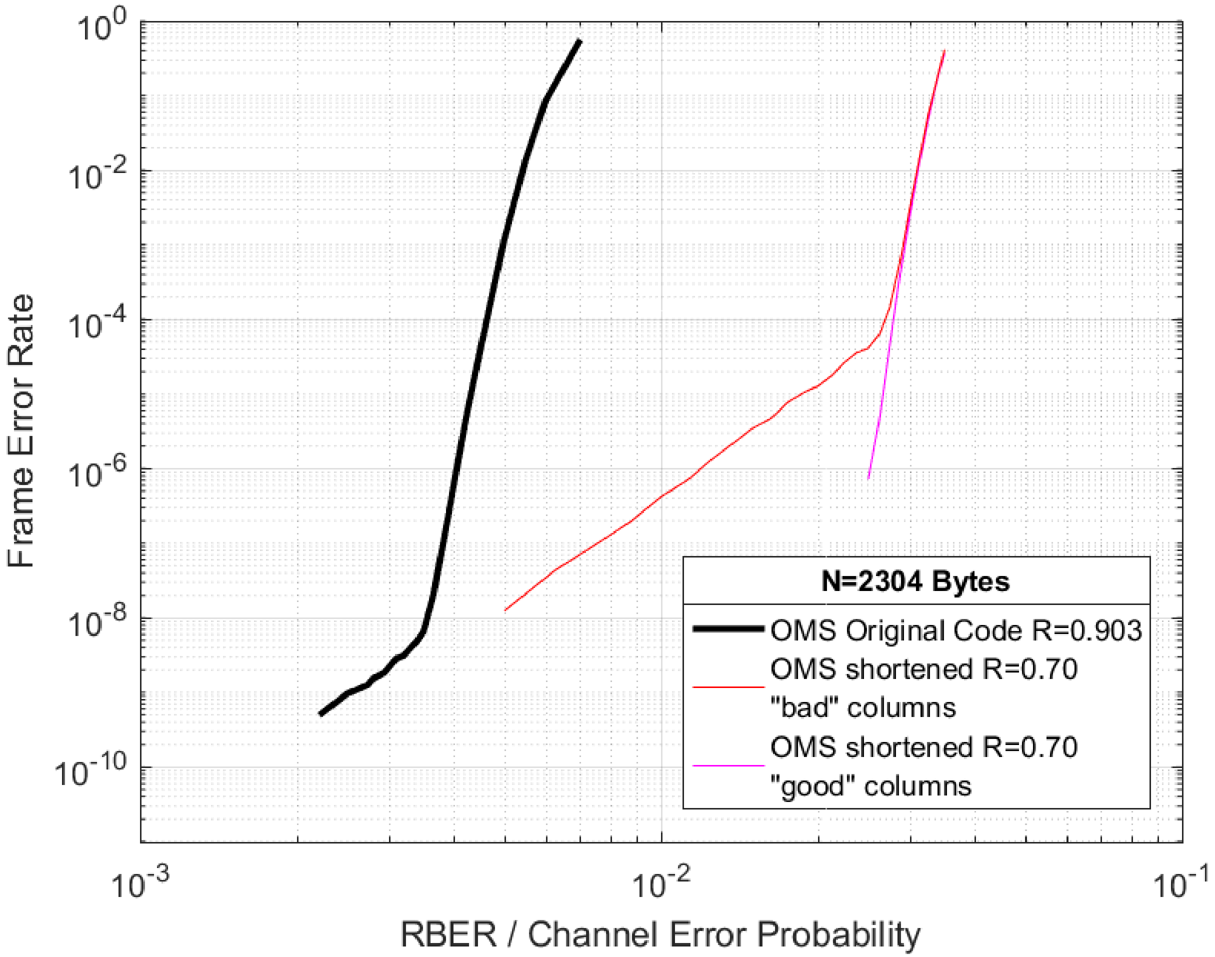
Results with the correction factor



Codelucida

THE UNIVERSITY OF ARIZONA.
TUCSON ARIZONA

Selecting right block columns is critical

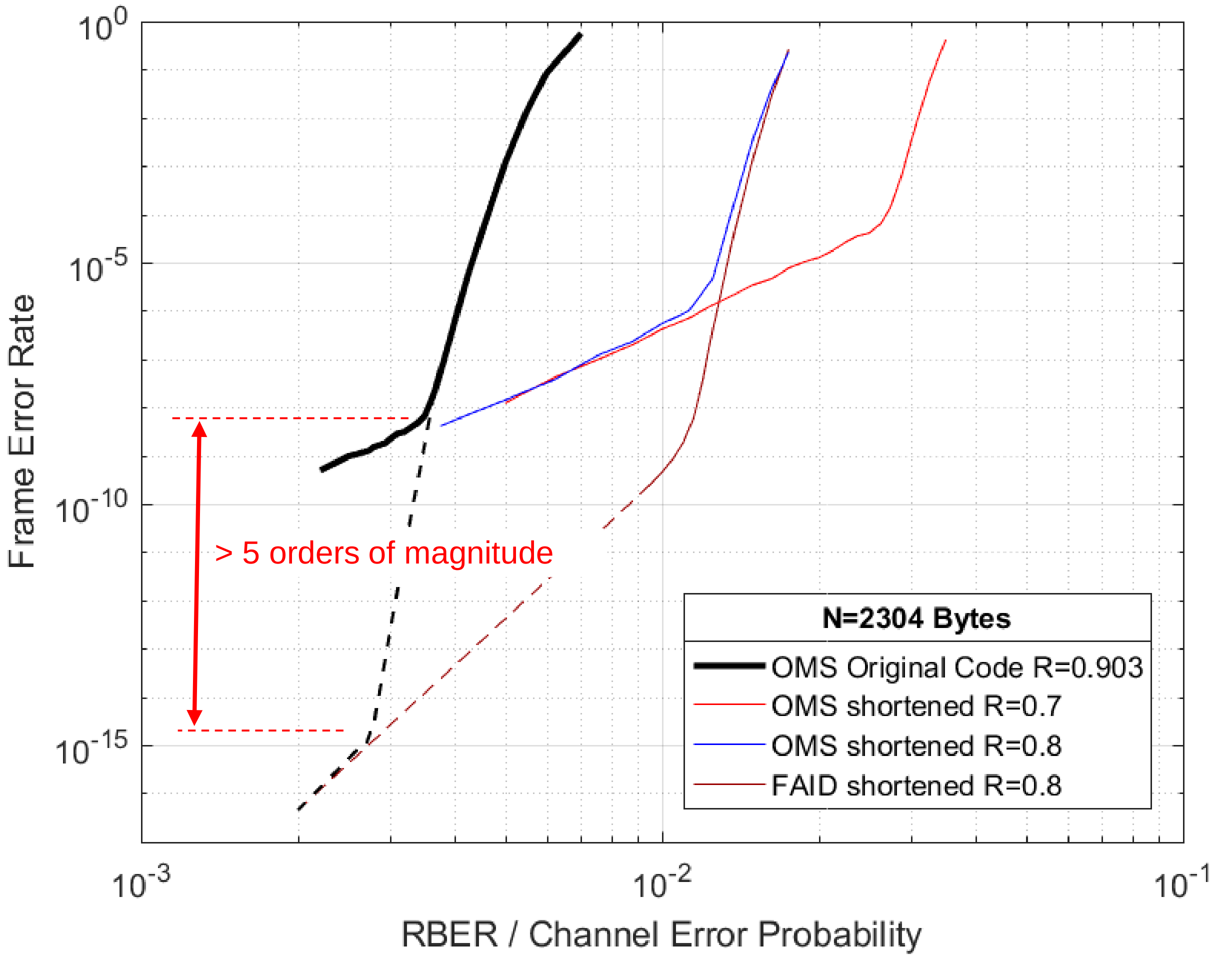


Codelucida

THE UNIVERSITY OF ARIZONA. TUCSON ARIZONA

Results for stronger decoders (FAID)

- Our prediction method is valid for ANY decoder



Codelucida

The University of Arizona
Tucson Arizona

Harmfulness

- Harmfulness of a trapping set is determined by its critical number. Relative harmfulness of two trapping sets with equal critical numbers C is a ratio of their strengths

Basic terminology

- *Failure inducing set* is a set of variable nodes that have to be initially in error for the decoder $\mathcal{D}$ to fail
- The *critical number* C 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
- *Strength* S of a trapping set with critical number C is the number of inducing sets of cardinality C (the number of weight-C error patterns on variable nodes in the trapping set)

S. K. Chilappagari, D. V. Nguyen, B. Vasić, and M. W. Marcellin, "Error correction capability of column-weight-three LDPC codes under the Gallager A algorithm - Part II," *IEEE Trans. Information Theory,* June 2010.

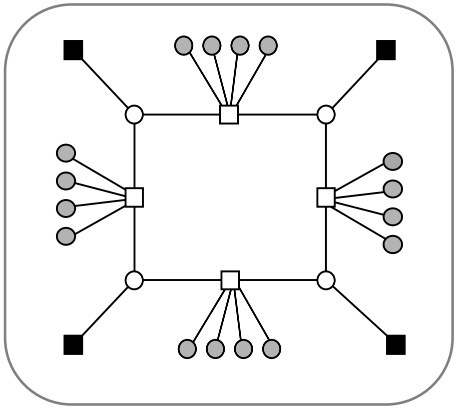
S. K. Chilappagari, D. V. Nguyen, B. Vasić, and M. W. Marcellin, "On Trapping Sets and Guaranteed Error Correction Capability of LDPC codes and GLDPC Codes," *IEEE Trans. Information Theory,* Apr. 2010.

B. Vasić, D.V. Nguyen, and S. K. Chilappagari, "Chapter 6 - Failures and Error Floors of Iterative Decoders ," *Academic Press Library in Mobile and Wireless Communications ,* 2014.

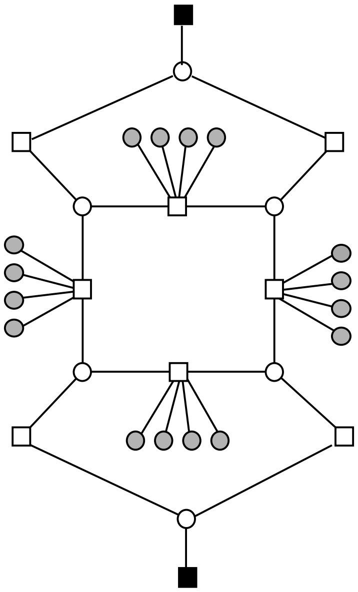
The expansion-contraction method

- In G find all **<u>dense</u>** subgraphs $\mathcal{G}$ up to a_{max} variable nodes that expand up to b_{max} check nodes
- The graphs $\mathcal{G}$ are not necessarily trapping sets
- Whether $\mathcal{G}$ contains a failure inducing set of variable nodes depends on its neighborhood in G
- Expand each $\mathcal{G}$ by including neighbors of degree-one check nodes up to certain depth – this creates a possibly large expanded graph $\mathcal{G}_{\mathrm{exp}}$
- Find the critical number C and *<u>all</u>* inducing sets $E_1, E_2, \ldots, E_s$ in $\mathcal{G}_{\mathrm{exp}}$ - the contracted graph induced by the variable nodes $\cup_i E_i$ is a true trapping set (with strength S)

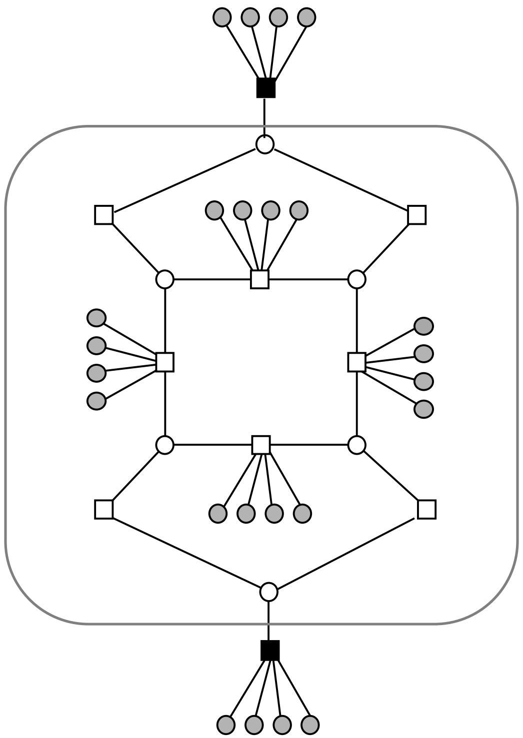
Depth-0



Expansion



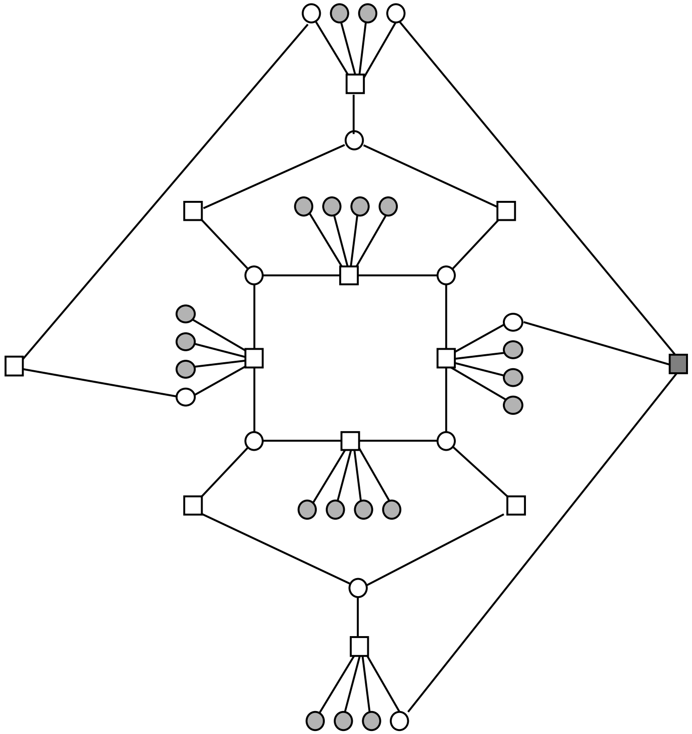
Depth-1



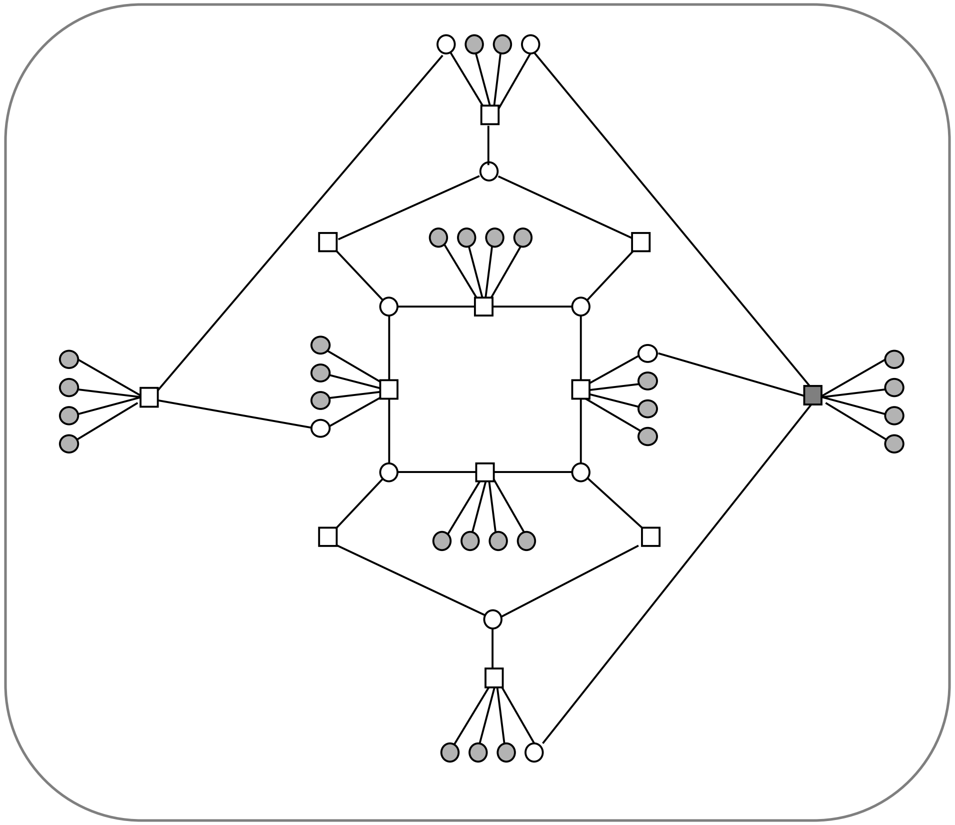
Codelucida

THE UNIVERSITY OF ARIZONA.
TUCSON ARIZONA

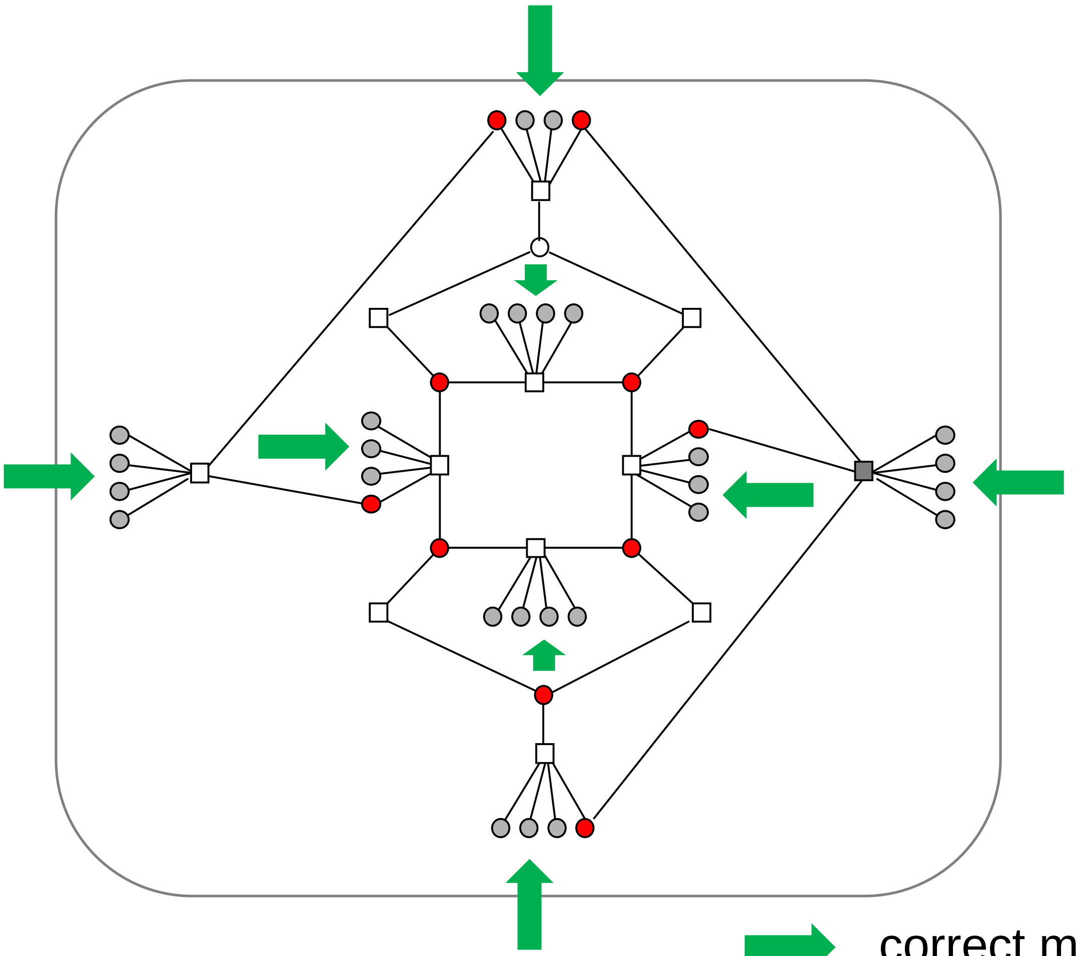
Expansion



Depth-2



Finding failure inducing sets • • • • •



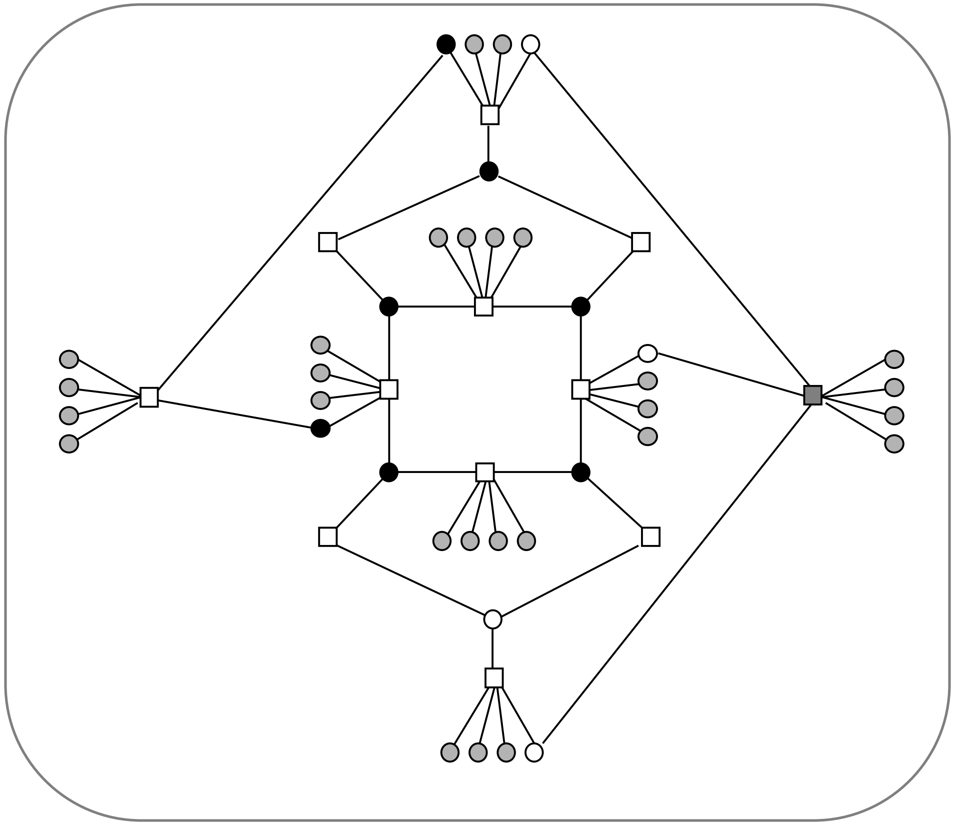
correct messages

possibly corrupt variables

Codelucida

THE UNIVERSITY OF ARIZONA.
TUCSON ARIZONA

Contraction

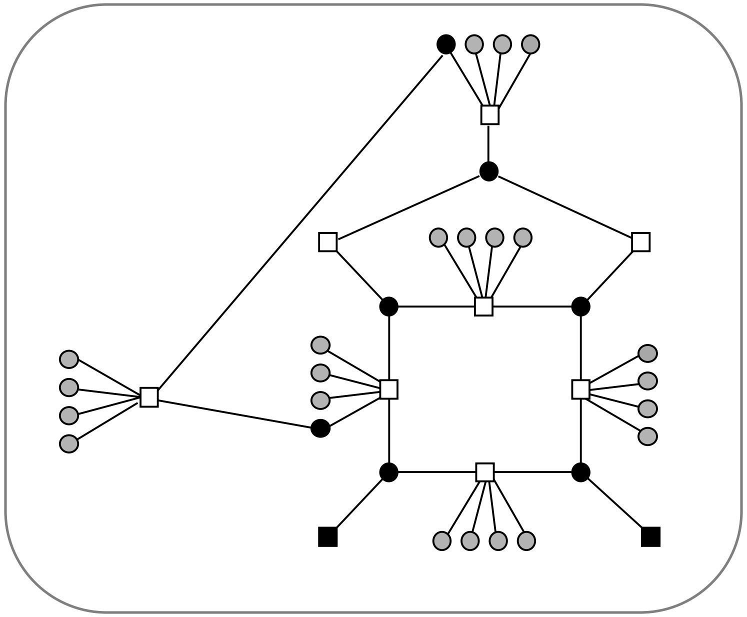


- variables appearing in at least one inducing set

Codelucida

THE UNIVERSITY OF ARIZONA. TUCSON ARIZONA

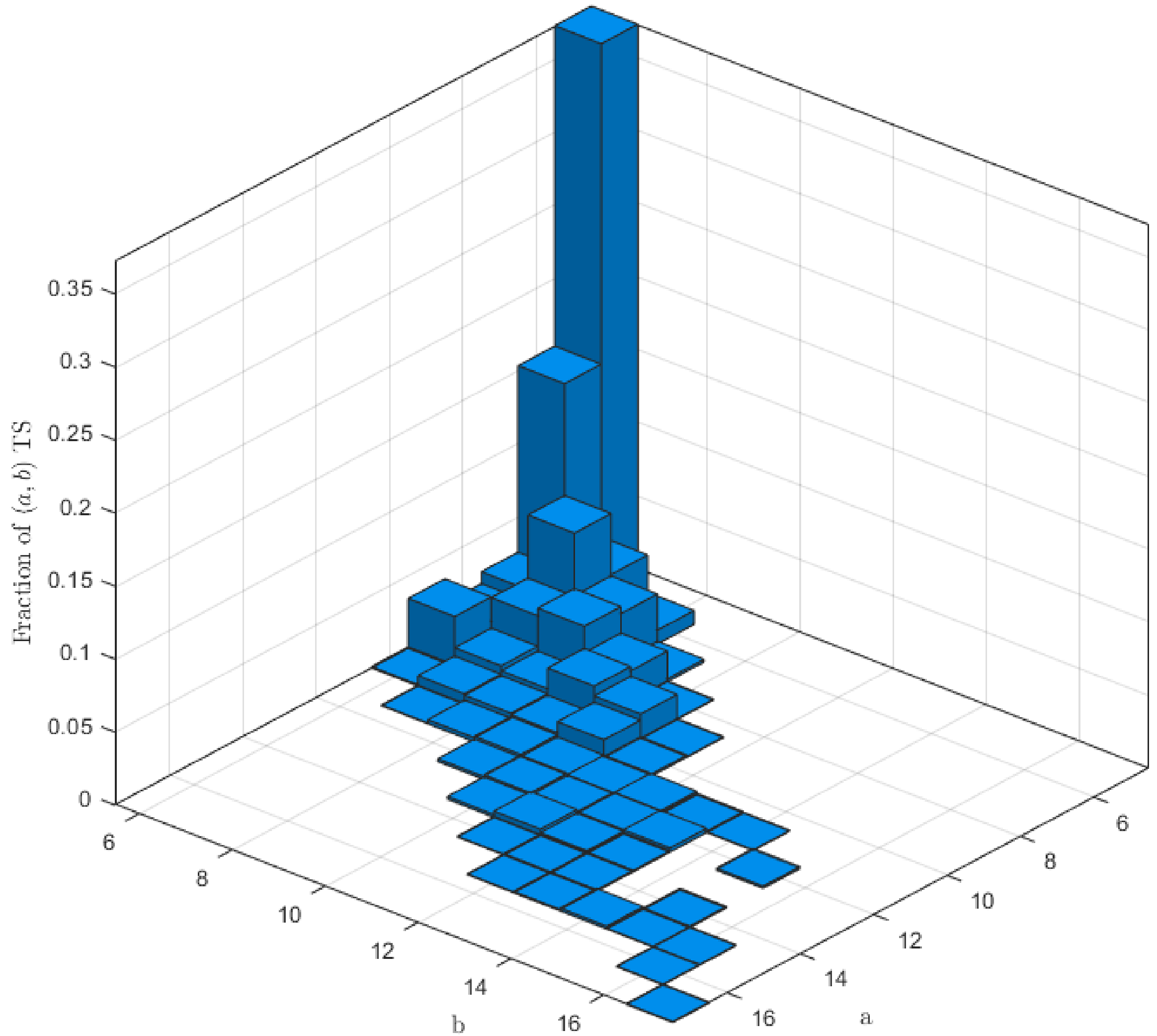
Contracted graph – the true trapping set



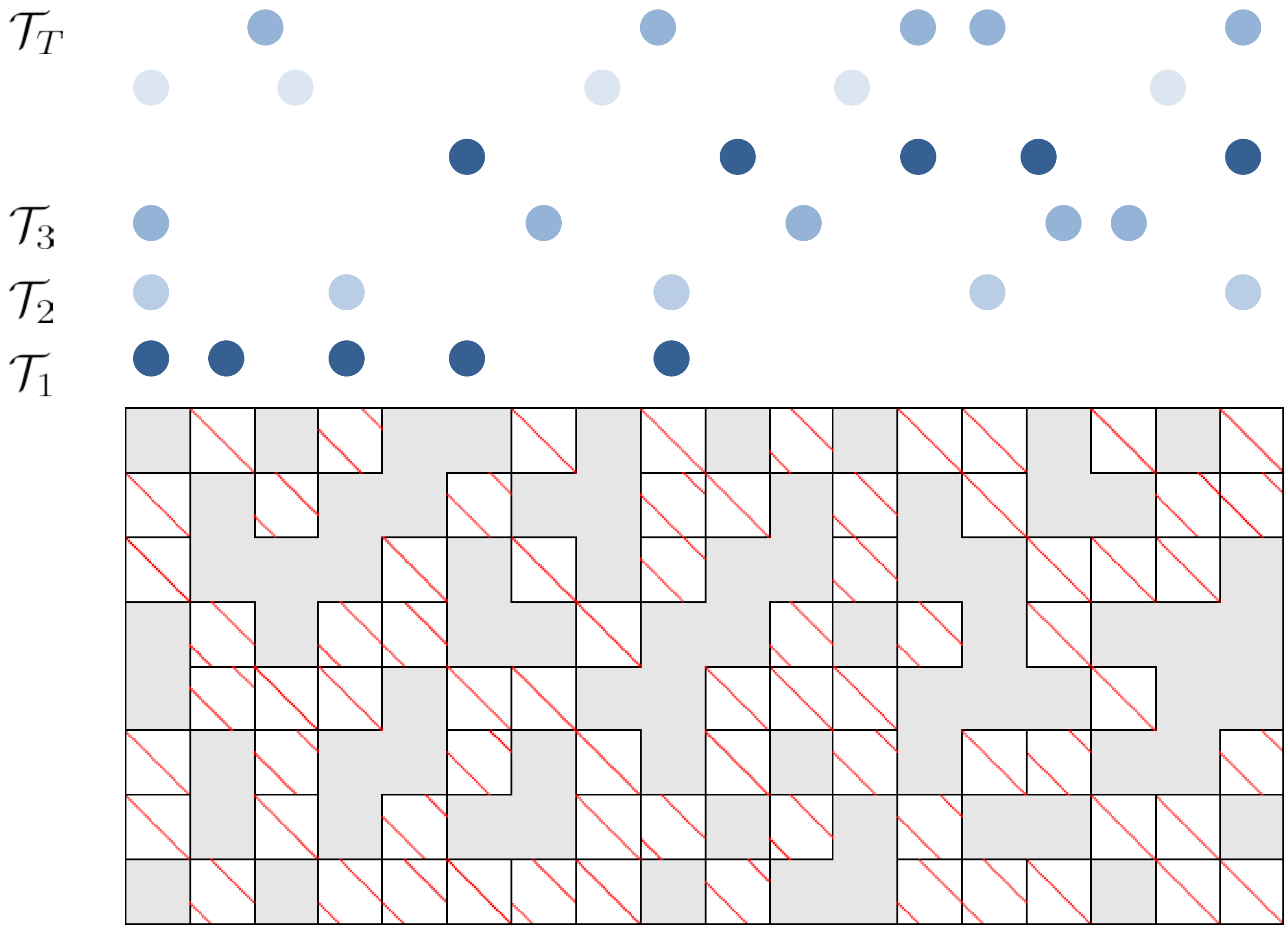
Codelucida

THE UNIVERSITY OF ARIZONA.
TUCSON ARIZONA

Distribution of trapping sets in a 2kB code



Identify harmful block columns and shorten



Codelucida

The University of Arizona
Tucson Arizona

Summary

- A computationally efficient method for estimating error floor of QC LDPC codes over the BSC channel
  - Arbitrary message update rule
  - Applicable to regular and irregular codes
  - Extendable to quantized output channels
- Graphs of small expansion in the Tanner graph are exhaustively expanded and contracted to obtain subgraphs that are true trapping sets
- Based on harmfulness of trapping sets code is shortened but in a way that it still contains most harmful trapping sets
- Allows fast optimization of decoders, and code optimization by removal of true trapping sets

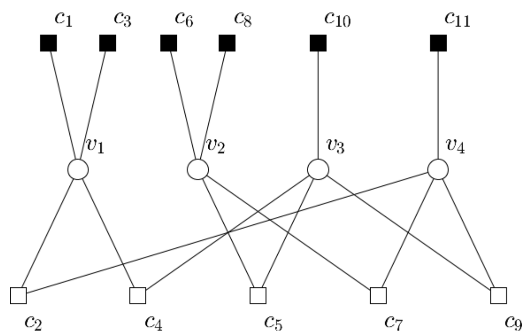
Thank you!

Codelucida

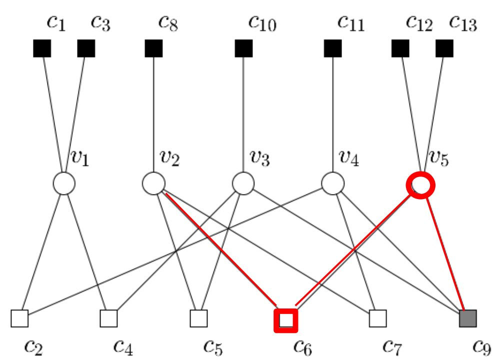
The University of Arizona
Tucson Arizona

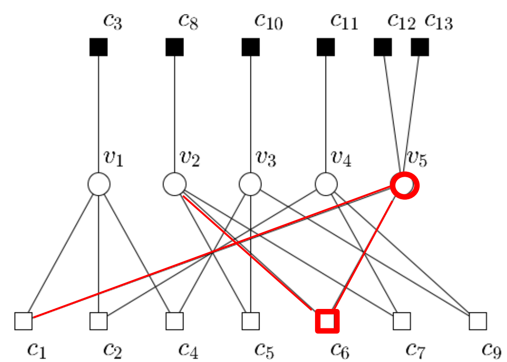
Expansion

(4,6) 0-2-1



(5,7) 0-4-1-2

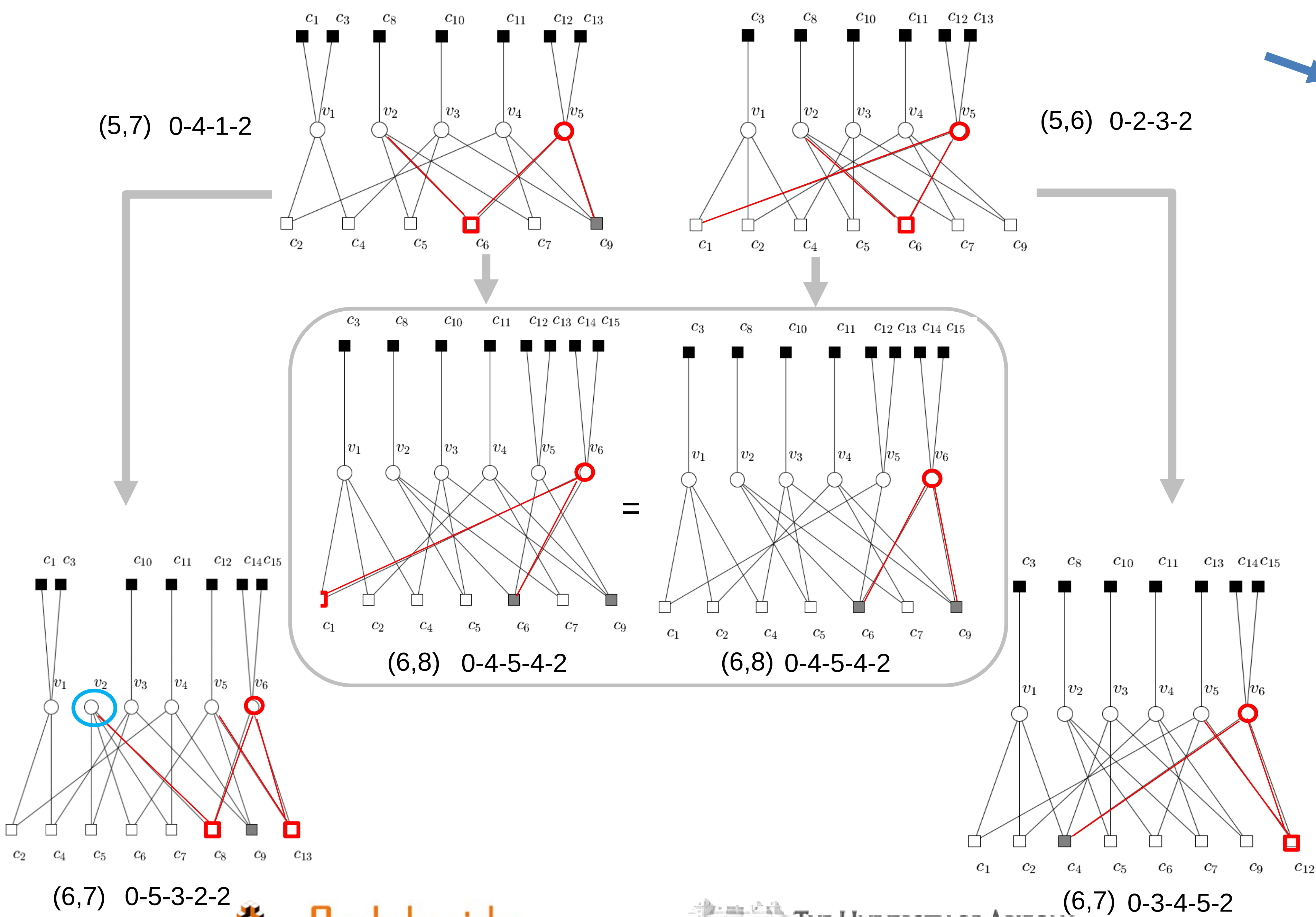




(5,6) 0-2-3-2

Codelucida

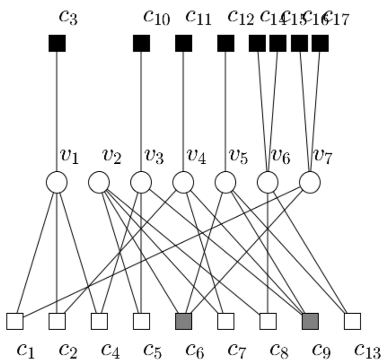
THE UNIVERSITY OF ARIZONA. TUCSON ARIZONA

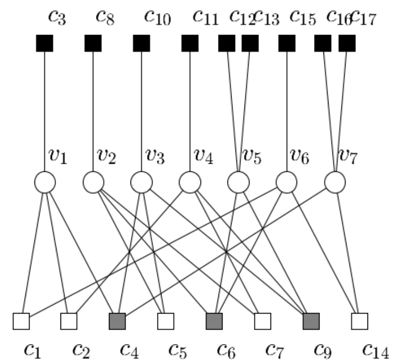


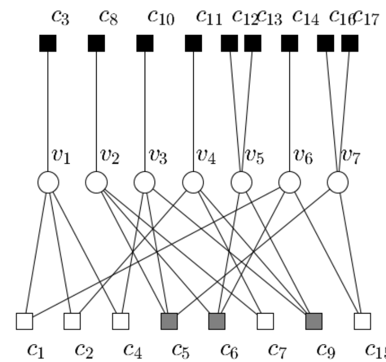
Codelucida

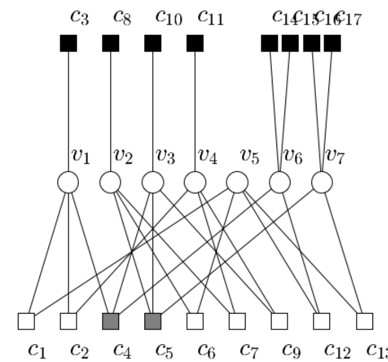
The University of Arizona
Tucson Arizona

Trapping sets with 7 variable nodes

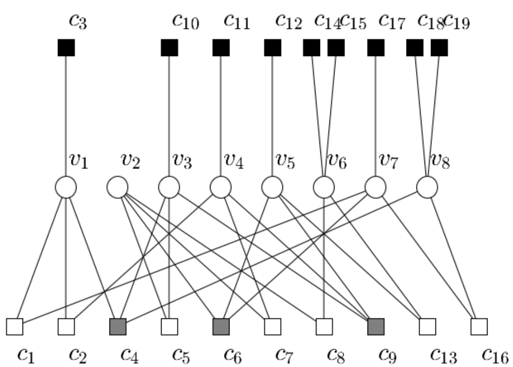


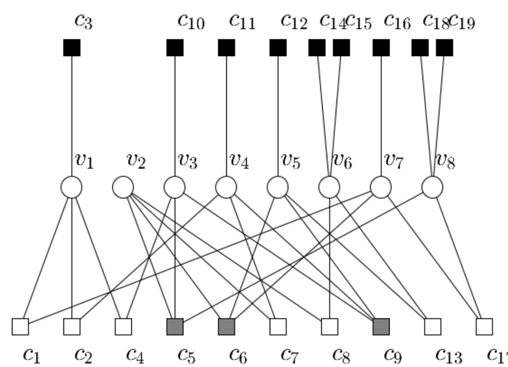


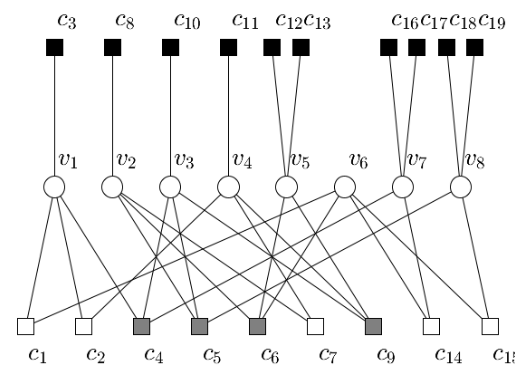




Trapping sets with 8 variable nodes







Trapping sets with 9 variable nodes