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
Trapping sets

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

![Diagram](image-url)

(5,3) trapping set (8,0) Trapping set
Some large trapping sets
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
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 $\text{FER}_{\mathcal{T}_1}, \text{FER}_{\mathcal{T}_2}, \ldots, \text{FER}_{\mathcal{T}_T}$ 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 $G$

\[
\text{FER} = \sum_{\mathcal{T}} w_\mathcal{T} \text{FER}_{\mathcal{T}}
\]
Harmfulness

- Major flaw of importance sampling – “subgraphs that are believed to be harmful”

  - 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 ($\gamma=3$)

- 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 $\gamma=4$, the number of nonisomorphic dense subgraphs is enormous.
- It is impractical to create a trapping set ontology as we did for $\gamma=3$, instead harmful subgraphs 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
Importance sampling through code shortening

• Create a shortened code $H_{\text{SHORT}}$ which contains the same most harmful trapping sets as the original code $H$, and run Monte Carlo simulations on $H_{\text{SHORT}}$ 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 $H_{\text{SHORT}}$ has lower rate, the error floor will appear at a higher FER, resulting in computational savings.

![Graph showing Frame Error Rate vs. RBER/Channel Error Probability]
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 $i$-th block harmfulness is equal to the sum of harmfulnesses of all trapping sets having a variable node in it.
Step 2

- Select the block-columns which have total harmfulness weight $H$ greater than a threshold $T$
Step 3

- Build a shortened version of the code, with $N_{\text{SHORT}}=9$ block-columns.
Step 4

- Correction factor - the ratio between remaining harmfulness weight and total harmfulness weight.
Direct simulation of the shortened codes
Results with the correction factor

N=2304 Bytes

- Black line: OMS Original Code R=0.903
- Blue line: OMS shortened R=0.80
- Red line: OMS shortened R=0.70
Selecting right block columns is critical.
Results for stronger decoders (FAID)

- Our prediction method is valid for ANY decoder
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).


The expansion-contraction method

- In $G$ find all **dense** subgraphs $G$ up to $a_{max}$ variable nodes that expand up to $b_{max}$ check nodes
- The graphs $G$ are not necessarily trapping sets
- Whether $G$ contains a failure inducing set of variable nodes depends on its neighborhood in $G$
- Expand each $G$ by including neighbors of degree-one check nodes up to certain depth – this creates a possibly large expanded graph $G_{exp}$
- Find the critical number $C$ and **all** inducing sets $E_1, E_2, \ldots, E_s$ in $G_{exp}$ - the contracted graph induced by the variable nodes $\bigcup_i E_i$ is a **true** trapping set (with strength $S$)
Depth-0
Expansion
Depth-1
Expansion
Depth-2
Finding failure inducing sets

- correct messages
- possibly corrupt variables
Contraction

- variables appearing in at least one inducing set
Contracted graph – the true trapping set
Distribution of trapping sets in a 2kB code
Identify harmful block columns and shorten
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!
Expansion

(4,6) 0-2-1

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

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