Bit Flipping Decoding Algorithms for LDPC codes

Author:
Miroslav Vanek
e-mail: miroslav.vanek@alcatel-lucent.com
Outline of presentation

- LDPC codes
  - Parity-check property
  - Characteristics
  - Description
  - Encoding
- LDPC codes decoding
  - “soft” decision decoding
  - “hard” decision decoding
Outline of presentation

- “Hard” decision decoding
  - Principles
  - Algorithms
    - Gallager algorithms
    - Weighted Bit Flipping algorithm
    - Parallel Weighted Bit Flipping algorithm
    - Fast Parallel Weighted Bit Flipping algorithm

- Results

- Conclusion
LDPC codes

- Introduced by R. Gallager (1964)
  - part of his PhD. Thesis
  - left unused for more than 30 years
- Subset of Linear Block Codes
  - linear property: \( c_1 + c_2 \in C \)
  - same length of each codeword
- Parity – Check codes
LDPC codes, parity-check property

Standard parity-check codes
- Checksum generated according to **Truth table**:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>X-OR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- Example of encoding:
  - Data = [1 1 1 1 1 1]
  - Encoded message = [1 1 1 1 1 1 0]
LDPC codes, parity-check property

- LDPC codes
  - using the same principal with *Truth table* as common parity-check codes
  - more parity-check bits added to the message
  - Also parity-check bits are involved in parity-check equation
- Example of encoding:
  - Data = [1 1 1 1 1 1]
  - Encoded message = [1 1 1 1 1 1 | 1]
LDPC codes, parity-check property

- LDPC codes
  - using the same principal with Truth table as common parity-check codes
  - more parity-check bits added to the message
  - Also parity-check bits are involved in parity-check equation

Example of encoding:
- Data = [1 1 1 1 1 1]
- Encoded message = [1 1 1 1 1 1 | 1 0]
LDPC codes, parity-check property

- LDPC codes
  - using the same principal with Truth table as common parity-check codes
  - more parity-check bits added to the message
  - Also parity-check bits are involved in parity-check equation

Example of encoding:
- Data = [1 1 1 1 1 1]
- Encoded message = [1 1 1 1 1 1 | 1 0 0]
LDPC codes

Code Description:
- sparse Parity – Check Matrix (matrix of huge dimensions)
- graphical representation (Tanner Graph)

Main characteristics:
- high performance (theoretically achieving Shannon limit)
- reliability
- increased need for computing resources in LDPC decoders
- successfully implemented in various standards (WiMAX)
Parity – Check Matrix

\[ H = \begin{bmatrix}
1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
\end{bmatrix} \]

- parity - check matrix
  - regular
  - irregular
- defines parity – check equations for each codeword
- if \( \mathbf{c} \) is a codeword, then: \( \mathbf{c} \times H^T = 0 \)
Parity – Check matrix: graphical representation

Tanner graph:
Encoding of LDPC codes

- Possibility of encoding using standard principles of linear codes encoding: $c = i \times G$

- Due to large dimensions of control matrix $H$, it is difficult to create generation matrix $G$

- Direct encoding algorithms – encoding based on control matrix $H$ instead of generation matrix $H$
  - encoder based on triangulation of control matrix
Decoding of LDPC codes

- Performance of ECC strongly dependent on decoding process

- Iterative decoding algorithms
  - sequential repair of erroneous bits instead of searching for closest codeword in code space

- Hard and Soft decision decoding algorithms
Decoding of LDPC codes

- **Hard decision decoding**
  - simple decoder construction
  - input values are not considering the channel information
  - bit flipping algorithms
  - faster convergence with significant impact on error correcting characteristics

- **Soft decision decoding**
  - complicated decoder construction
  - channel information is considered in decoding process
  - message passing algorithms
  - slow converging, but more powerful methods of decoding
Hard decision decoding – 
Bit Flipping decoding

- Principles of Bit Flipping decoding

  - Define **parity-check equations** from control matrix H
  - Based on parity-check equations define **evaluation metrics**
  - Results of this metrics defines a **set** of bits to be flipped
  - Flip the values of selected bits
  - Check the result and proceed to next iteration
Gallager algorithm – type A

- Basic hard decision bit flipping algorithm for LDPC decoding
- Simple construction of decoder – only comparator and XOR elements used
- Tanner graph used for making decisions
- Particular bit is considered to be correct – all check node influences are satisfied
Gallager algorithm, type A - algorithm

- Check all parity-check equations
  - If all of them are satisfied -> end decoding

- Find the number of all unsatisfied parity-check equations for all received bits
  - If this number for particular bit = overall number of its parity-check nodes -> flip this bit

- Check the stopping criteria and proceed to next iteration
Gallager algorithm, type A - example
Gallager algorithm – type B

- Using the same principle as type A algorithm

- Particular bit is flipped also in case, that not all of parity-check equations are unsatisfied (threshold approach)

- Ability to adapt the algorithm during the encoding process (threshold adaptation)
Gallager algorithm, type B - example

- Let threshold \( t=1 \)
Weighted Bit Flipping algorithm

- Description:
  - smoothing the difference between simple BF and soft decoding
  - Simple algorithm
  - Iterative approach
  - Flips only that bit of codeword, for which metric

\[
E_n(\alpha) = \sum_{m \in M(n)} (2s_m - 1)w_{n,m} - \alpha|y_n|
\]

achieves the maximum value

- Slow convergence due to flipping only one bit per iteration
WBF - algorithm

- Count syndrome components for all check nodes
  \[
  s_m = \sum_n h_{m,n} \cdot z_n
  \]

- Count evaluation metrics for each bit of received codeword
  \[
  E_n(\alpha) = \sum_{m \in M(n)} (2s_m - 1) \left| w_{n,m} \right| - \alpha \left| y_n \right|
  \]
  \[
  w_{n,m} = \min_{i \in N(m) \setminus n} \left| y_i \right|
  \]

- Flip the bit with highest value of \( E_n \)

- Check the final codeword and proceed to next iteration
Parallel Weighted Bit Flipping algorithms

Description:
- Improving WBF algorithms in the way of convergence time and number of iterations
- Introduces the ability of parallel flipping of several bits in one iteration
- Parallelization allows to improve speed of decoding and even improve overall performance of decoder
PWBF - algorithm

- Count syndrome component for each parity – check equation
- Count the metric from WBF algorithm for each bit of received codeword
- Each unsatisfied parity – check equation “votes” for one bit to be flipped
- If the number of votes for particular bit exceeds the threshold, this bit is flipped
- All bits are flipped in parallel
PWBF - example

Threshold = 3

18. March 2010
Fast PWBF algorithm – “Guaranteed” bits approach

- Analogy with Gallager algorithm – type A
- Each transmitted codeword contains bits transmitted without error (this bits are called “Guaranteed”)
- Set of guaranteed bits is identified using parity – check equations
- Bit is marked as “Guaranteed” only if all the parity – check equations, in which this bit participates are satisfied
- If bit is marked as “Guaranteed”, it is not necessary to decide, whether this bit needs to be flipped or not
“Guaranteed” bits – short example

\[ c = [0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0] \]

\[ H = \begin{bmatrix}
1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 
\end{bmatrix} \]
**“Guaranteed” bits – short example**

\[
c = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

\[
H = \begin{bmatrix}
1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0
0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1
0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0
0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0
\end{bmatrix}
\]
“Guaranteed” bits – short example

\[
\mathbf{c} = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

\[
\mathbf{H} = \begin{bmatrix}
1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0
0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0
0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0
\end{bmatrix}
\]
“Guaranteed” bits – short example

\[ c = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \]

\[ H = \begin{bmatrix}
1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
\end{bmatrix} \]
“Guaranteed” bits – short example

\[ c = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \]

\[ H = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \]
Average number of “Guaranteed” bits
Introduction to Fast PWBF algorithm

- Based on principles of standard PWBF algorithm
- Using “Guaranteed” bits approach speeds up the convergence
- Algorithm changes:
  - in each iteration, “Guaranteed” bits are found for received codeword
  - for “Guaranteed” bits, no evaluation metrics needs to be calculated and its value is set to $-\infty$ directly
- This approach leads to approximately 25% savings of computational time without any visible decrease of decoding performance
WBF vs. Fast WBF comparison
BF algorithms - comparison
BF vs. "soft" decoding - comparison

18. March 2010
Conclusion

- LDPC codes: “future of Coding Theory”
- Importance of decoding
- Hard vs. Soft decision decoding
- Hard decision decoding – space for improvement
References:


- Wu X., Zhao Ch., You X., “Parallel weighted Bit – Flipping decoding”, IEEE Communications letters, vol. 11, no. 8, August 2007, pp. 671 – 673

Thank you for your attention.