GSoC ’13
LDPC Codes and More FEC in GNU Radio

Manu T S
Mentor : Dr.-Ing. Jens Elsner

IIT Bombay

October 1, 2013
An Evaluation

The Original Proposal

- Generic Encoder/Decoder for LDPC codes
- Algorithms for generating LDPC codes
- Improve RS codes, add BCH etc

The Outcomes

- Generic Encoder/Decoder for LDPC codes
- 2 Algorithms for generating LDPC codes
- An attempt to incorporate LDPC into FECAPI
What is an LDPC Code

A linear error correcting code
All codewords should satisfy
\[ Hx = 0 \] (1)
Any vector that satisfy above equation is a codeword

A block code
If \( H \) is \( M \times N \) and has rank \( R \) then the code has dimension
\[ K = N - R \]
Block length of the code is \( N \).

Sparse \( H \)
Here we are dealing with binary LDPC codes, so \( H \) has few 1s rest all zeros
Tanner Graph

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

(a)
How we split up the work

Construction of LDPC codes.
- Regular LDPC matrix
- LDPC codes based on Reed-Solomon codes
- Progressive Edge Growth Algorithm

Encoding Techniques.
- Transform parity check matrix into approximate upper triangular matrix
- Obtain $G = [I \ P]$ followed by matrix multiplication

Decoding Techniques.
- Bit-Flip Decoding
- Message Passing decoder for BSC
- Message Passing decoder for AWGN Channel
If the parity check matrix is given as \([I P]\), where \(I\) is \(N - K \times N - K\) identity matrix, \(P\) is some \(N - K \times K\) matrix, a codeword \(x = [x_p^T \ x_d^T]^T\) can be formed by assigning

\[x_p = P \cdot x_d\]

We obtain \(G = [I P]\) from \(H\) by
- Column permutation
- Row additions
- Row permutations

Suppose \(f\) represents the permutation in obtaining \(G\) from \(H\), and \(G \cdot x = 0\), then \(H \cdot f(x) = 0\)
The Algorithms - Encoding

\[ f(x) \]

\[ x = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{bmatrix} \]

\[ x_p \]

\[ x_d \]

\[ f = \begin{bmatrix} 2 & 0 & 3 & 1 & 5 & 4 & 6 \end{bmatrix} \]

\[ f(x) = \begin{bmatrix} x_1 & x_3 & x_0 & x_2 & x_5 & x_4 & x_6 \end{bmatrix} \]
The Algorithms - Encoding

```cpp
std::vector<char> cldpc::encode(std::vector<char> dataword) {
    if (dataword.size() == K) {
        GF2Vec x(N);
        GF2Vec data(K);
        data.set_vec(dataword);
        for (int i = rank_H; i < N; i++) {
            x[i] = dataword[i - rank_H];
        }
        for (int i = 0; i < rank_H; i++) {
            x[i] = G[i].sub_vector(N-K, N)*data;
        }
        GF2Vec y(N);
        for (int i = 0; i < N; i++) {
            y[permute[i]] = x[i];
        }
        return y.get_vec();
    }
}
```
The Algorithms - Message Passing

check-to-variable

variable-to-check

source: Manu T S (IITB)
The Algorithms - Message Passing

\begin{align*}
\text{(0 0)} & \quad \text{(0 0)} \\
\text{(? ?)} & \quad \text{(? ?)} \\
\text{(0 0)} & \quad \text{(0 0)} \\
\text{(1 1)} & \quad \text{(0 ?)} \\
\text{(0 ?)} & \quad \text{(1 ?)} \\
\text{(0 0)} & \quad \text{(0 0)}
\end{align*}

\begin{align*}
\text{7} & \quad \text{3} \\
\text{6} & \quad \text{3} \\
\text{5} & \quad \text{2} \\
\text{4} & \quad \text{2} \\
\text{3} & \quad \text{2} \\
\text{2} & \quad \text{1} \\
\text{1} & \quad \text{1}
\end{align*}

\begin{align*}
\text{7} & \quad \text{3} \\
\text{6} & \quad \text{3} \\
\text{5} & \quad \text{2} \\
\text{4} & \quad \text{2} \\
\text{3} & \quad \text{2} \\
\text{2} & \quad \text{1} \\
\text{1} & \quad \text{1}
\end{align*}
std::vector<char> awgn_bp::decode(std::vector<float> rx_word,
    int *niteration) {
    *niteration = 0;
    compute_init_estimate(rx_word);
    if (is_codeword()) {
        return estimate;
    }
    else {
        rx_lr_calc(rx_word);
        spa_initialize();
        while (*niteration < max_iterations) {
            *niteration += 1;
            update_chks();
            update_vars();
            decision();
            if (is_codeword()) {
                break;
            }
        }
        return estimate;
    }
}
The Algorithms - Generation of LDPC codes

Regular and Irregular Progressive Edge Growth Tanner Graphs

- Suboptimal solution for a hard Combinatorial problem
- Initialized from $N$, $M$, and symbol node degree sequence
- The insertion of new edge has minimum effect on the girth of the graph
- Both regular and irregular LDPC codes can be constructed

LDPC Codes based on RS Codes with 2 information symbols

- Constructs a regular LDPC codes.
- Makes sure that there are no length 4 cycles
Introducing gr-ldpc OOT Module

The classes and blocks

- Class for GF(2) Vector arithmetic.
- Class for GF(2) Matrix arithmetic.
- Class for accessing LDPC codes in “alist-file” format.
- Class for holding an LDPC code.
- Class for message passing mechanisms for BSC and AWGN channels.
- LDPC Encoder block.
- LDPC Decoder blocks, for both float inputs and byte inputs.
- Two algorithms for construction of LDPC codes
- Codes available in github
Introducing gr-ldpc OOT Module
(3, 6) Regular Code of block length $N = 96$, dimension $K = 50$, Number of parity checks $M = 48$, $10^7$ Bytes Transferred

<table>
<thead>
<tr>
<th>sigma</th>
<th>No Code</th>
<th>With LDPC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Errors</td>
<td>BER</td>
</tr>
<tr>
<td>0.3</td>
<td>4338</td>
<td>0.0004338</td>
</tr>
<tr>
<td>0.4</td>
<td>61814</td>
<td>0.0061814</td>
</tr>
<tr>
<td>0.5</td>
<td>227228</td>
<td>0.0227228</td>
</tr>
<tr>
<td>0.6</td>
<td>477744</td>
<td>0.0477744</td>
</tr>
<tr>
<td>0.7</td>
<td>765326</td>
<td>0.0765326</td>
</tr>
<tr>
<td>0.8</td>
<td>1055848</td>
<td>0.1055848</td>
</tr>
<tr>
<td>0.9</td>
<td>1332039</td>
<td>0.1332039</td>
</tr>
<tr>
<td>1.0</td>
<td>1585653</td>
<td>0.1585653</td>
</tr>
</tbody>
</table>
LDPC generated using PEG Algorithm,
Block length $N = 1008$, Dimension $K = 504$
Number of Parity Checks $M = 504$

<table>
<thead>
<tr>
<th>sigma</th>
<th>No Code Errors</th>
<th>No Code BER</th>
<th>With LDPC Errors</th>
<th>With LDPC BER</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3</td>
<td>3898</td>
<td>0.196461</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0.4</td>
<td>18938</td>
<td>0.954488</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0.5</td>
<td>19840</td>
<td>0.999994</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0.6</td>
<td>19841</td>
<td>1.0</td>
<td>17</td>
<td>0.0008568</td>
</tr>
<tr>
<td>0.7</td>
<td>19841</td>
<td>1.0</td>
<td>19841</td>
<td>1.0</td>
</tr>
</tbody>
</table>
BER Plot

BER Comparison

No Code
With LDPC

Manu T S (IITB)
GSoC ’13
October 1, 2013
Current Issues & Future Work

Current Issues
- Have not finished porting them to fecapi
- Initialization process is very slow.
- LDPC codes saved as alist-file is only accepted.

Future Work
- Finish porting the codes to fecapi
- Optimization of message passing algorithms, initialization steps
- Benchmarking the algorithms
- VOLK for encoding, GF(2) matrix multiplication,
- Bring more of FEC into fecapi framework
Lessons Learned and Experiences

- How to use gr_modtool.
- fecapi framework.
- Importance of testing the code.
- Keep the mentor posted about what you do.
- Improved my coding skills.
I take this opportunity to thank

- Jens
- Nick, Tracie
- Prof. Sibi Raj B Pillai, IIT Bombay
- Prof. Radford Neal, University of Toronto
- Prof. Eric G. Larsson, Linköping University
- Sreeraj
- GNU Radio Developers and Users
- Ettus Research
- GNU Radio Mailing List, USRP Users Mailing List
- Google SoC Team


