Constructing error-correcting codes with huge distances

Florian Hug

Department of Electrical and Information Technology
Lund University, Sweden
Outline

1. Convolutional Codes
2. BEAST
3. Graphs & Hypergraphs
4. Conclusions & Outlook
Outline

1 Convolutional Codes

2 BEAST

3 Graphs & Hypergraphs

4 Conclusions & Outlook
General Model of a Communication System

Source

Source encoder

Channel encoder

Digital channel

Destination

Source decoder

Channel decoder

\[ \hat{u} \rightarrow u \rightarrow v \rightarrow \hat{r} \]
General Model of a Communication System

- Block codes
- Convolutional codes
Applications

Convolutional codes are used for
- Radio-Communications
- Mobile-Communications
- Satellite-Communications
- Space-Communications
“Famous” (7,5) rate $R = 1/2$ convolutional code with memory $m = 2$ and overall constraint length $\nu = 2$

Can be easily extended to general rate $R = b/c$ convolutional codes.
“Famous” (7,5) rate $R = 1/2$ convolutional code with memory $m = 2$ and overall constraint length $\nu = 2$

\[
v = uG \quad \quad G = \begin{pmatrix} g_1(D) & g_2(D) \\ \end{pmatrix} = \begin{pmatrix} 1 + D + D^2 & 1 + D^2 \\ \end{pmatrix} = \begin{pmatrix} 7 & 5 \end{pmatrix}
\]

Can be easily extended to general rate $R = b/c$ convolutional codes.
Trellis Representation

- $2^\nu$ different nodes $\xi$
- $2^b$ branches
Characterization

- Memory $m$
- Overall constraint length $\nu$
- Rate $R = b/c$
- Free distance
  \[ d_{\text{free}} = \min_{\nu \neq \nu'} \{ d_H(\nu, \nu') \} = \min_{\nu \neq 0} \{ w_H(\nu) \} \]
- Spectrum
Characterization

- Memory $m$
- Overall constraint length $\nu$
- Rate $R = b/c$
- Free distance
  \[ d_{\text{free}} = \min_{v \neq v'} \{ d_H(v, v') \} = \min_{\nu \neq 0} \{ w_H(\nu) \} \]
- Spectrum

Burst-Error Probability (BSC)

\[ P_B \leq \sum_{d = d_{\text{free}}}^{\infty} n_d \left( 2 \sqrt{\frac{e}{1 - e}} \right)^d \]
Outline

1. Convolutional Codes
2. BEAST
3. Graphs & Hypergraphs
4. Conclusions & Outlook
BEAST
Bidirectional Efficient Algorithm for Searching Trees

$R = b/c$ convolutional code

Find the number of codewords of weight $w = f_w + b_w$
BEAST
Bidirectional Efficient Algorithm for Searching Trees

- $R = b/c$ convolutional code
  Find the number of codewords of weight $w = f_w + b_w$

### Forward and Backward Sets

- $\mathcal{F}_{+j} = \{ \xi | w_F(\xi) = f_w + j, \ w_F(\xi^p) < f_w, \ \sigma(\xi) \neq 0 \}$
- $\mathcal{B}_{-j} = \{ \xi | w_B(\xi) = b_w - j, \ w_B(\xi^c) > b_w, \ \sigma(\xi) \neq 0 \}$

$\ j = 0, 1, \ldots, c$
BEAST
Bidirectional Efficient Algorithm for Searching Trees

- $R = b/c$ convolutional code
  Find the number of codewords of weight $w = f_w + b_w$

**Forward and Backward Sets**

- $\mathcal{F}_{+j} = \{ \xi | w_\mathcal{F}(\xi) = f_w + j, \ w_\mathcal{F}(\xi^p) < f_w, \ \sigma(\xi) \neq 0 \}$
- $\mathcal{B}_{-j} = \{ \xi | w_\mathcal{B}(\xi) = b_w - j, \ w_\mathcal{B}(\xi^c) > b_w, \ \sigma(\xi) \neq 0 \}$

- sort and match $\mathcal{F}_{+j}$ with $\mathcal{B}_{-j}$
- number of matches is equal to number of codewords $n$ of weight $w$
**Example**

\[ f_w = 3 \]
\[ w = f_w + b_w = 6 \]
\[ b_w = 3 \]

\[ \mathcal{F}_{+0} = \{(0 \ 1), \ (1 \ 1)\} \]
\[ \mathcal{F}_{+1} = \emptyset \]

\[ \mathcal{B}_{-0} = \{(1 \ 1), \ (1 \ 1), \ (1 \ 0)\} \]
\[ \mathcal{B}_{-1} = \emptyset \]
**Example**

\[ f_w = 3 \quad w = f_w + b_w = 6 \quad n = 2 \]

\[ b_w = 3 \]

**BEAST**

\[ \mathcal{F}_{+0} = \{(0 1), (1 1)\} \]

\[ \mathcal{F}_{+1} = \emptyset \]

\[ \mathcal{B}_{-0} = \{(1 1), (1 1), (1 0)\} \]

\[ \mathcal{B}_{-1} = \emptyset \]
Parallel Implementations

- Only a smaller degree of parallelization possible (recursion)
  - $c$ forward and $c$ backward sets
  - $2c$ individual sorts
  - $c$ mergers
- Fast and large growing sets (exceeding available memory)
- File I/O becomes a bottleneck
Outline

1. Convolutional Codes

2. BEAST

3. Graphs & Hypergraphs

4. Conclusions & Outlook
Graphs & Hypergraphs

2-uniform, 3-regular, 2-partite graph

Tannner graph representation
Encoding matrix of a rate $R = 5/20$ woven graph code

$$G_{wg}(D) = \begin{pmatrix} G_0(D) & G_1(D) & G_2(D) & G_3(D) & G_4(D) \\ G_4(D) & G_0(D) & G_1(D) & G_2(D) & G_3(D) \\ G_3(D) & G_4(D) & G_0(D) & G_1(D) & G_2(D) \\ G_2(D) & G_3(D) & G_4(D) & G_0(D) & G_1(D) \\ G_5(D) & G_5(D) & G_5(D) & G_5(D) & G_5(D) \end{pmatrix}$$

$G_0 = (1473, 40453, 16256, 62224)$

$G_1 = (44364, 50324, 36077, 30173)$

$G_2 = (53717, 4266, 30434, 32352)$

$G_3 = (37464, 14262, 6517, 71254)$

$G_4 = (47726, 14624, 31724, 5234)$

$G_5 = (4463, 7413, 6523, 6153)$. 
Encoding matrix of a rate $R = 5/20$ woven graph code

$$G_{wg}(D) = \begin{pmatrix} G_0(D) & G_1(D) & G_2(D) & G_3(D) & G_4(D) \\ G_4(D) & G_0(D) & G_1(D) & G_2(D) & G_3(D) \\ G_3(D) & G_4(D) & G_0(D) & G_1(D) & G_2(D) \\ G_2(D) & G_3(D) & G_4(D) & G_0(D) & G_1(D) \\ G_5(D) & G_5(D) & G_5(D) & G_5(D) & G_5(D) \end{pmatrix}$$

Free Distance

Using BEAST leads to

$$d_{\text{free}} = 120$$

Size of Forward and Backward Sets was 1.4 TB
Outline

1. Convolutional Codes
2. BEAST
3. Graphs & Hypergraphs
4. Conclusions & Outlook
Conclusions & Outlook

So far...

- Huge free distances can be verified with BEAST
- Iterative implementation was derived
- Algorithm was ported to Cell Broadband Engine (PS3)

Maybe...

- Further speed-ups by using Solid-State-Drives
- Higher parallelization degree possible
The End

Thanks a lot for your attention
Constructing error-correcting codes with huge distances

Florian Hug

Department of Electrical and Information Technology
Lund University, Sweden