Constructing error-correcting codes with huge distances

Florian Hug

Department of Electrical and Information Technology

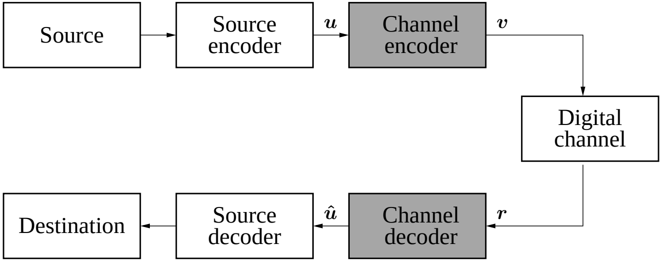
Lund University, Sweden

Outline

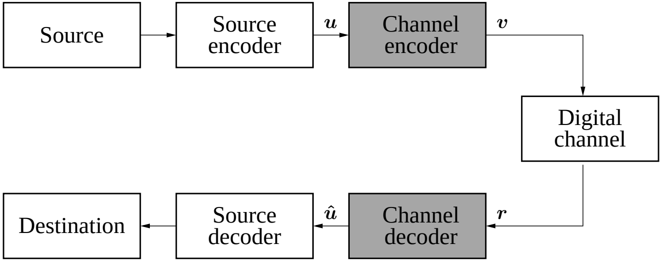
Outline

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

General Model of a Communication System



General Model of a Communication System



- Block codes
- Convolutional codes

Applications

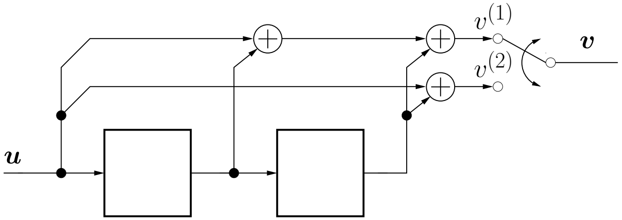
Convolutional codes are used for

- Radio-Communications
- Mobile-Communications
- Satellite-Communications
- Space-Communications

Convolutional Encoder

"Famous" (7,5) rate $R = 1/2$ convolutional code with

memory $m = 2$ and overall constraint length $\nu = 2$

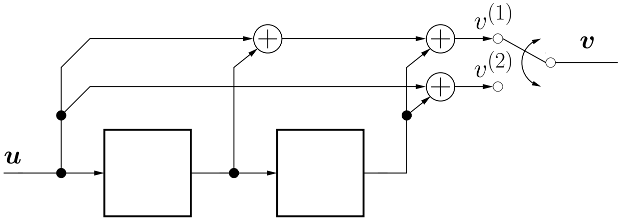


Can be easily extended to general rate $R = b/c$ convolutional codes.

Convolutional Encoder

"Famous" (7,5) rate $R = 1/2$ convolutional code with

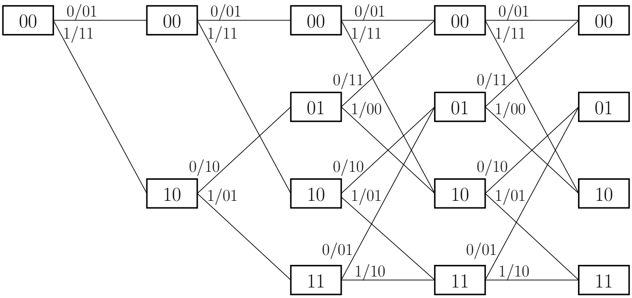
memory $m = 2$ and overall constraint length $\nu = 2$



$$
\begin{aligned}
\boldsymbol{v} &= \boldsymbol{u}G \\
G &= \left(g_1(D) \quad g_2(D)\right) \\
&= \left(1 + D + D^2 \quad 1 + D^2\right) \\
&= \left(7 \quad 5\right)
\end{aligned}
$$

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

Trellis Repesentation



- 2^{ν} different nodes ξ
- 2^{b} branches

Characterization

- Memory m
- Overall constraint length ν
- Rate $R = b/c$
- Free distance

$$d_{\mathsf{free}} = \min_{\boldsymbol{v} \neq \boldsymbol{v'}} \left\{ d_{\mathsf{H}}(\boldsymbol{v}, \boldsymbol{v'}) \right\} = \min_{\boldsymbol{v} \neq \boldsymbol{0}} \left\{ w_{\mathsf{H}}(\boldsymbol{v}) \right\}$$

- Spectrum

Characterization

- Memory m
- Overall constraint length ν
- Rate $R = b/c$
- Free distance

$$d_{\mathsf{free}} = \min_{\boldsymbol{v} \neq \boldsymbol{v}'} \left\{ d_{\mathsf{H}}(\boldsymbol{v}, \boldsymbol{v}') \right\} = \min_{\boldsymbol{v} \neq \boldsymbol{0}} \left\{ w_{\mathsf{H}}(\boldsymbol{v}) \right\}$$

- Spectrum

Burst-Error Probability (BSC)

$$P_{\mathsf{B}} \leq \sum_{d=d_{\mathsf{free}}}^{\infty} n_d \left(2\sqrt{\varepsilon(1-\varepsilon)} \right)^d$$

Outline

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

$$\mathcal{F}_{+j} = \{\xi \,|w_{\mathcal{F}}(\xi) = f_w + j,\; w_{\mathcal{F}}(\xi^{\mathsf{P}}) < f_w,\; \boldsymbol{\sigma}(\xi) \neq \mathbf{0}\}$$
$$\mathcal{B}_{-j} = \{\xi \,|w_{\mathcal{B}}(\xi) = b_w - j,\; w_{\mathcal{B}}(\xi^{\mathsf{C}}) > b_w,\; \boldsymbol{\sigma}(\xi) \neq \mathbf{0}\}$$
$$j = 0, 1, \ldots, c$$

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

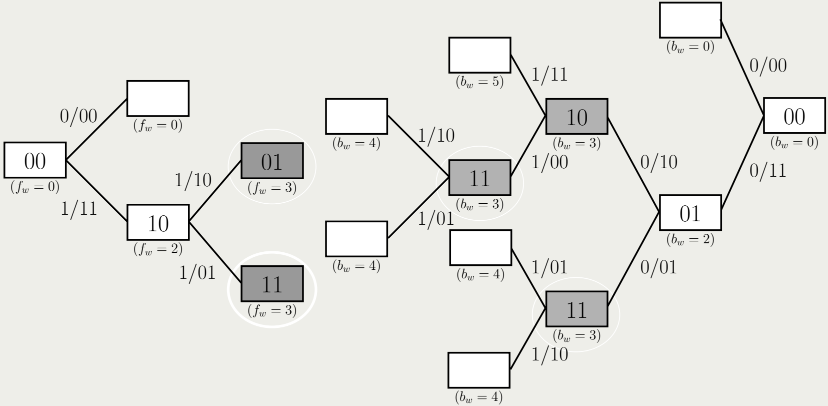
$$\mathcal{F}_{+j} = \{\xi \,|w_{\mathcal{F}}(\xi) = f_w + j,\ w_{\mathcal{F}}(\xi^{\mathsf{P}}) < f_w,\ \boldsymbol{\sigma}(\xi) \neq \mathbf{0}\}$$
$$\mathcal{B}_{-j} = \{\xi \,|w_{\mathcal{B}}(\xi) = b_w - j,\ w_{\mathcal{B}}(\xi^{\mathsf{C}}) > b_w,\ \boldsymbol{\sigma}(\xi) \neq \mathbf{0}\}$$

$j = 0, 1, \ldots, c$

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

Example

$f_w = 3$ $\qquad$ $w = f_w + b_w = 6$ $\qquad$ $b_w = 3$



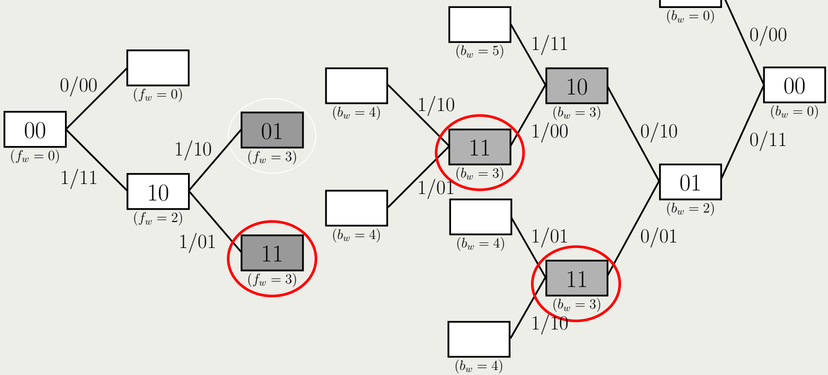
$$\mathcal{F}_{+0} = \{(0\ 1),\ \ (1\ 1)\} \qquad \mathcal{B}_{-0} = \{(1\ 1),\ \ (1\ 1),\ \ (1\ 0)\}$$
$$\mathcal{F}_{+1} = \emptyset \qquad \mathcal{B}_{-1} = \emptyset$$

BEAST

Example

$f_w = 3$ $\qquad w = f_w + b_w = 6$ $\qquad b_w = 3$

$n = 2$



$\mathcal{F}_{+0} = \{(0\ 1),\ \ (1\ 1)\}$ $\qquad \mathcal{B}_{-0} = \{(1\ 1),\ \ (1\ 1),\ \ (1\ 0)\}$

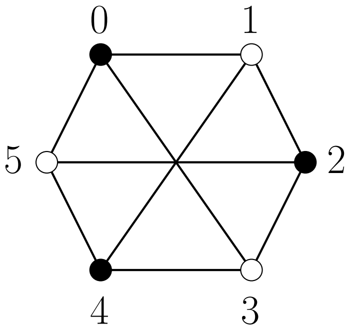
$\mathcal{F}_{+1} = \emptyset$ $\qquad \mathcal{B}_{-1} = \emptyset$

Parallel Implementations

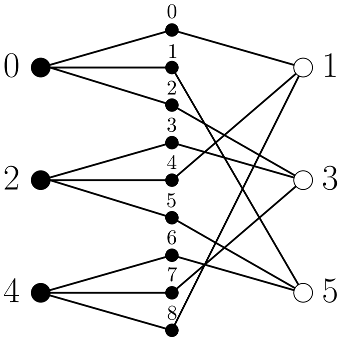
- Only a smaller degree of parallelization possible (recursion)
  - c forward and c backward sets
  - $2c$ individual sorts
  - c mergers
- Fast and large growing sets (exceeding available memory)
- File I/O becomes a bottleneck

Outline

Graphs & Hypergraphs



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



Tannner graph representation

Example

Example

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

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

$$\begin{aligned} G_0 &= \begin{pmatrix} 1473 & 40453 & 16256 & 62224 \end{pmatrix} \\ G_1 &= \begin{pmatrix} 44364 & 50324 & 36077 & 30173 \end{pmatrix} \\ G_2 &= \begin{pmatrix} 53717 & 4266 & 30434 & 32352 \end{pmatrix} \\ G_3 &= \begin{pmatrix} 37464 & 14262 & 6517 & 71254 \end{pmatrix} \\ G_4 &= \begin{pmatrix} 47726 & 14624 & 31724 & 5234 \end{pmatrix} \\ G_5 &= \begin{pmatrix} 4463 & 7413 & 6523 & 6153 \end{pmatrix}. \end{aligned}$$

Example

Example

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

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

Free Distance

Using BEAST leads to

$$\mathbf{d}_{\text{free}} = \mathbf{120}$$

Size of Forward and Backward Sets was 1.4 TB

Outline

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