Decoding Woven Convolutional LDPC Codes

Kondrashov. K
Zyablov. V

Institute for Information Transmission Problems
Russian Academy of Science

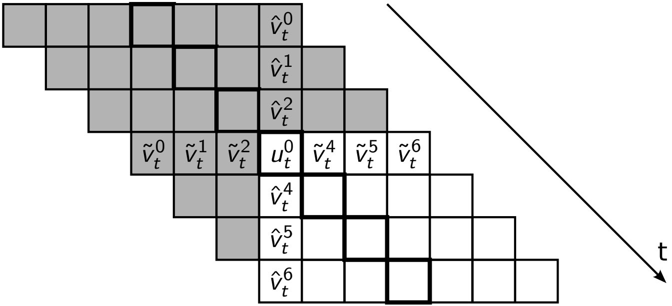
Contents

2-woven code construction

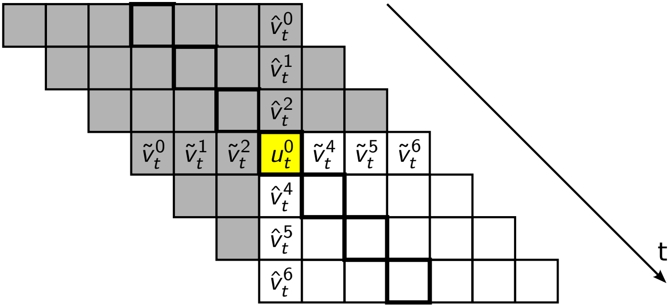
Braided Block Code (BBC).
A. J. Felström, M. Lentmaier, D. V. Truhachev and K. Sh. Zigangirov, *Braided block codes, IEEE Proc. Inf. Th*, 1999

Each symbol is covered by 2 constituent codes.

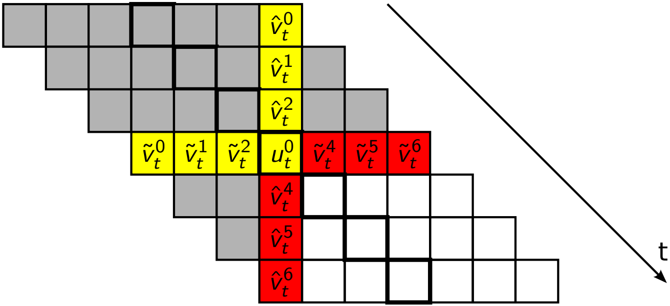
Braided Block Code



Braided Block Code



Braided Block Code

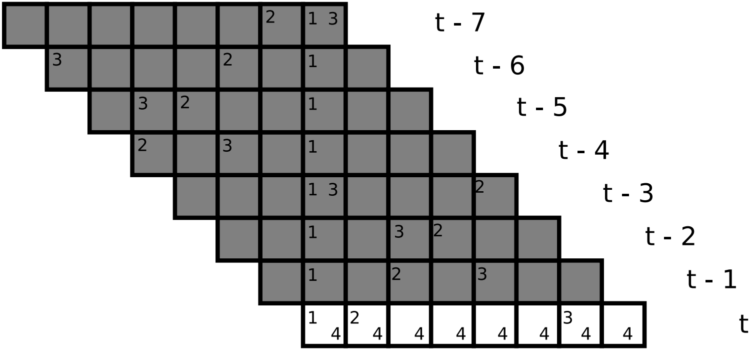


4-woven code construction

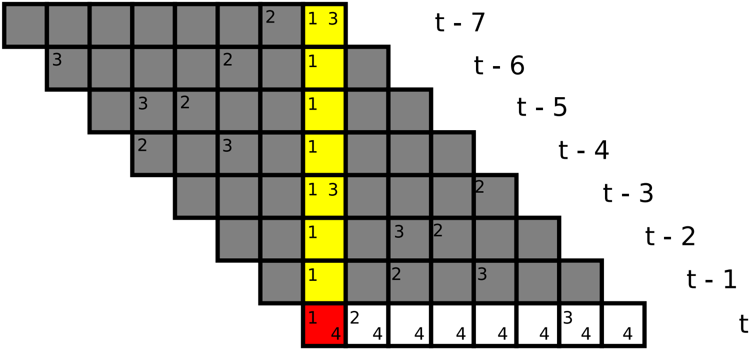
4-woven convolutional LDCP code.
V. V. Zyablov, K. A. Kondrashov, *Two LDPCC constructions, Information Technologies and Systems Workshop, Becasovo, Russia*, 2009

Each symbol is covered by 4 constituent codes.
Constituent codes are simple parity-check codes.

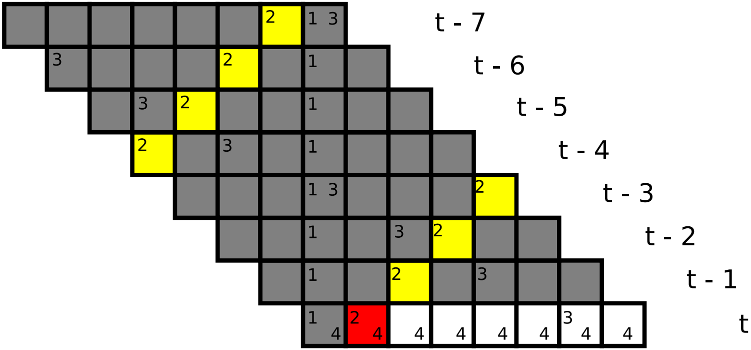
4-woven (4,8)-LDPC convolutional code



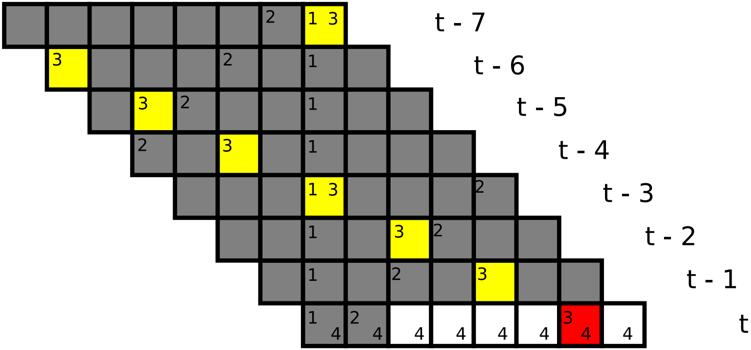
4-woven (4,8)-LDPC convolutional code



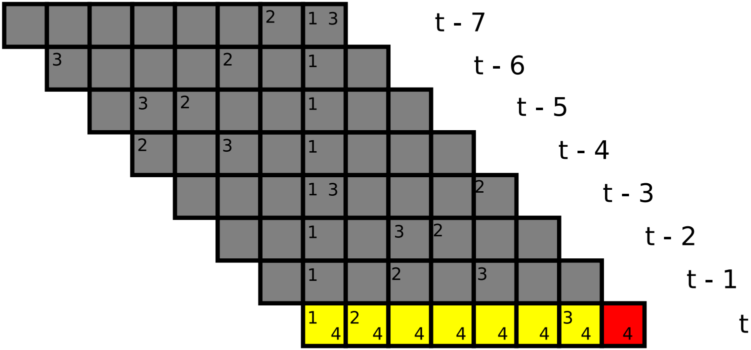
4-woven (4,8)-LDPC convolutional code



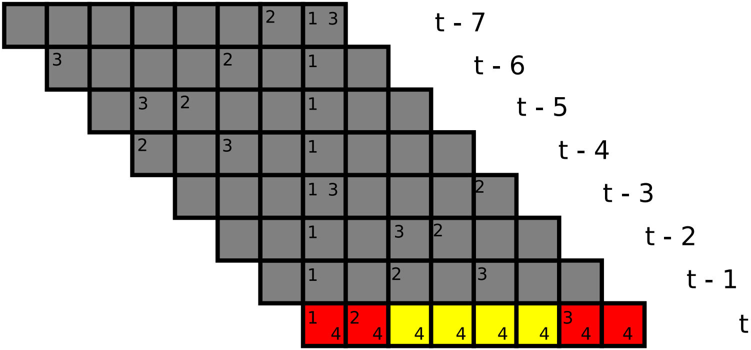
4-woven (4,8)-LDPC convolutional code



4-woven (4,8)-LDPC convolutional code



4-woven (4,8)-LDPC convolutional code



Convolutional LDPC code parity-check matrix

$$H^{\mathrm{T}} = \begin{pmatrix} H_0^{\mathrm{T}}(0) & \dots & H_{m_s}^{\mathrm{T}}(m_s) & & \\ & \ddots & & \ddots & \\ & & H_0^{\mathrm{T}}(t) & \dots & H_{m_s}^{\mathrm{T}}(t+m_s) \\ & & \ddots & & \ddots \end{pmatrix}$$

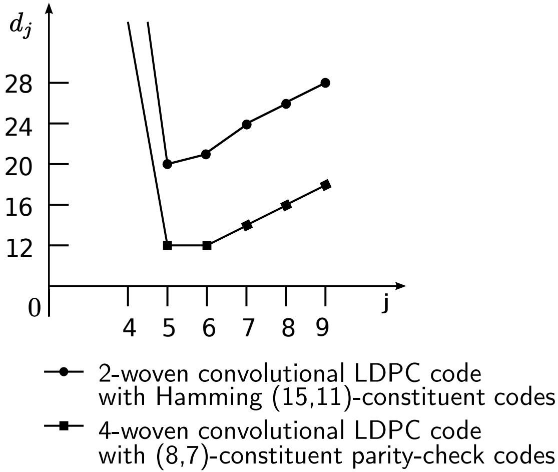
$H_i^{\mathrm{T}}(t)$ is of dimension $c \times (c-b)$.

b is information block length.

c is encoded block length.

m_s is convolutional LDPC code memory.

$$\begin{aligned} d_{free} &= \min_{v \neq v'} \{d_{\mathrm{H}}(v, v')\} \\ d_j &= \min \left\{\omega_{\mathrm{H}}(v_{[1,j]})\right\} : \; v_{[1,j]} H^{\mathrm{T}}_{[1,j+m_s-1]} = \mathbf{0}. \end{aligned}$$


d_j
28
24
20
16
12
0
4
5
6
7
8
9
j
2-woven convolutional LDPC code with Hamming (15,11)-constituent codes
4-woven convolutional LDPC code with (8,7)-constituent parity-check codes

Decoding algorithms

Notation:

$\boldsymbol{r}$ – received erroneous word.
$\boldsymbol{r}^{(i)}$ – i-th iteration input, $\boldsymbol{r}^{(1)} = \boldsymbol{r}$.
$\boldsymbol{r}^{(i+1)}$ – i-th iteration output.
$\left\{\mathcal{D}^{(k)}\right\}_{k=1}^{J}$ – set of parallel constituent codes error correcting decoders.
$\left\{\mathcal{E}^{(k)}\right\}_{k=1}^{J}$ – set of parallel constituent codes erasure correcting decoders.

At each iteration two stages of decoding are carried out: inner decoding and outer decoding. At inner decoding no changes are made to symbols of input word.

Iterative majority-voting algorithm $\mathcal{A}_1$

1. For each constituent code k its decoder $\mathcal{D}^k$ decodes corresponding inner words from $\boldsymbol{r}^{(i)}$. Results are stored in $\boldsymbol{r}_k^{(i+1)}$;
2. $\boldsymbol{r}^{(i+1)}$ is generated. Its symbols $r_j^{(i+1)}$ are obtained from the majority voting function over $r_{k,j}^{(i+1)}$:

   if more than a half of decoders gave up with the same value for $r_{k,j}^{(i+1)}$, say α, then $r_j^{(i+1)}$ is set to α. Otherwise $r_j^{(i+1)}$ gets old value $r_j^{(i)}$.

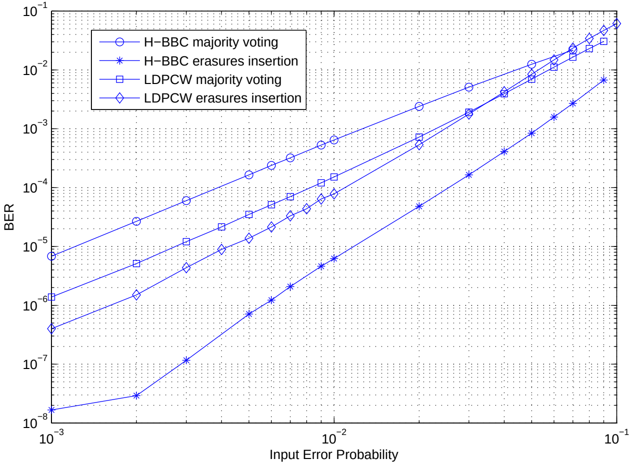
Decoding continues while syndrome is not all zero and input and output differs.

Iterative majority-voting algorithm $\mathcal{A}_2$ with erasure insertion

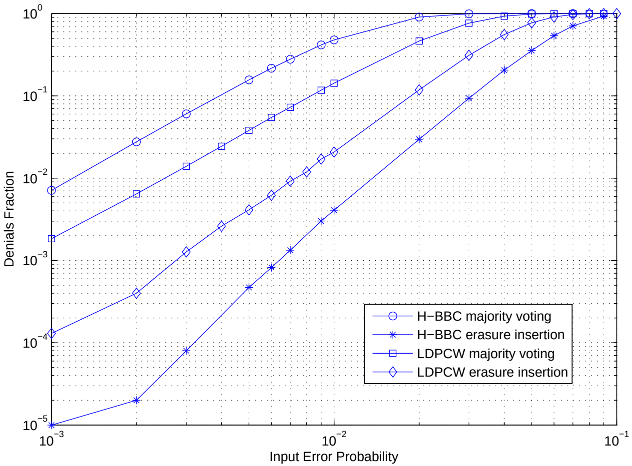
1. For each constituent code k its decoder $\mathcal{D}^k$ decodes corresponding inner words from $\boldsymbol{r}^{(i)}$. Results are stored in $\boldsymbol{r}_k^{(i+1)}$;
2. $\boldsymbol{r}^{(i+1)}$ is generated.
   $r_j^{(i+1)}$ are set to α if more than a half of $r_{k,j}^{(i+1)}$ are equal to α. Otherwise $r_j^{(i+1)}$ is erased and inner decoders are switched from $\mathcal{D}^k$ to $\mathcal{E}^k$. Decoding continues with $\mathcal{A}_1$ algorithm unless all erasures are corrected.

Decoding stops when syndrome is all zero, input and output do not differ or no erasures were corrected at last iteration.

Decoding simulation results



Decoding simulation results



Thank you for your attention!